



DI-Guy SDK Version 12.5.2

Copyright (C) 1992–2012 Boston Dynamics

ALL RIGHTS RESERVED.

These coded instructions, statements, and computer programs contain unpublished proprietary information of Boston Dynamics and are protected by Copyright Laws of the United States. They may not be used, duplicated, or disclosed in any form, in whole or in part, without the prior written consent from Boston Dynamics.

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the government is subject to restrictions as set forth in FAR 52.227.19(c)(2) or subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227–7013 and/or in similar or successor clauses in the FAR, or the DOD or NASA FAR Supplement, or to subparagraphs (c)(1) and (c)(2) of the Commercial Computer Software--Restricted Rights at 48 CFR 52.227–19, as applicable. Unpublished—rights reserved under the Copyright Laws of the United States.

Contractor/Manufacturer is:

Boston Dynamics/78 Fourth Avenue/Waltham MA 02451.

Table of Contents

1 Types List.....	1
2 Files List.....	5
3 Functions List.....	8
4 Constants List.....	80
5 Types Reference.....	94
5.1 diguyAgentParams.....	95
5.2 diguyApp.....	168
5.3 diguyAuthorInterface.....	234
5.4 diguyAuthorVisualMaterial.....	264
5.5 diguyAuthorVisualMultiline.....	271
5.6 diguyAuthorVisualObjectHandle.....	279
5.7 diguyAuthorVisualRegionMesh.....	291
5.8 diguyAuthorVisualRegionPaintbrush.....	302
5.9 diguyAuthorVisualWaypoint.....	312
5.10 diguyChainSettings.....	319
5.11 diguyChainSimulation.....	365
5.12 diguyCharacter.....	375
5.13 diguyCharacterFaceExpression.....	911
5.14 diguyCharacterGesture.....	920
5.15 diguyCharacterGestureType Type Declaration.....	970
5.16 diguyCharacterGroup.....	972
5.17 diguyCharacterGuide.....	1000
5.18 diguyCharacterPath.....	1013
5.19 diguyCharacterPathActionBead.....	1051
5.20 diguyCharacterPathAimBead.....	1074
5.21 diguyCharacterPathDecisionBead.....	1087
5.22 diguyCharacterPathEvent.....	1100
5.23 diguyCharacterPathGazeBead.....	1103
5.24 diguyCharacterPathScriptBead.....	1116
5.25 diguyCharacterPoseOverride.....	1131
5.26 diguyCharacterTypeMap.....	1142
5.27 diguyCrowd.....	1156
5.28 diguyCrowdProfile.....	1221
5.29 diguyExerciseInterface.....	1237
5.30 diguyFormation.....	1257
5.31 DI-Guy Graphics API Programming Overview.....	1272
5.32 diguyGraphicsFile.....	1282
5.33 diguyGraphicsLink.....	1288
5.34 diguyGraphicsMaterial.....	1340
5.35 diguyGraphicsMesh.....	1350
5.36 diguyGraphicsShaderInstance.....	1377
5.37 diguyGraphicsShaderProgram.....	1395
5.38 diguyGraphicsShaderTechnique.....	1429
5.39 diguyGraphicsShape.....	1443

Table of Contents

5 Types Reference

5.40 diguyGraphicsState.....	1475
5.41 diguyGraphicsTexture.....	1484
5.42 diguyGraphicsVertexFormats.....	1504
5.43 diguyIGuyController.....	1507
5.44 diguyImpact.....	1544
5.45 diguyInfoPopup.....	1572
5.46 diguyInteractionMachine.....	1578
5.47 diguyLoadManager.....	1626
5.48 diguyMotionDirection Type Declaration.....	1647
5.49 diguyMotionPosture Type Declaration.....	1651
5.50 diguyMotionSpeedType Type Declaration.....	1653
5.51 diguyMotionVariant Type Declaration.....	1656
5.52 DI-Guy Graphics API, diguyOglGraphicsLink.....	1658
5.53 DI-Guy Graphics API, diguyOglGraphicsMaterial.....	1666
5.54 DI-Guy Graphics API, diguyOglGraphicsMesh.....	1670
5.55 DI-Guy Graphics API, diguyOglGraphicsShaderInstance.....	1677
5.56 DI-Guy Graphics API, diguyOglGraphicsShaderProgram.....	1680
5.57 DI-Guy Graphics API, diguyOglGraphicsShape.....	1701
5.58 DI-Guy Graphics API, diguyOglGraphicsTexture.....	1706
5.59 DI-Guy Graphics API, diguyOsgExtGraphicsFile.....	1712
5.60 DI-Guy Graphics API, diguyOsgExtGraphicsLink.....	1717
5.61 DI-Guy Graphics API, diguyOsgExtGraphicsShape.....	1730
5.62 DI-Guy Graphics API, diguyOsgGraphicsLink.....	1737
5.63 DI-Guy Graphics API, diguyOsgGraphicsMaterial.....	1757
5.64 DI-Guy Graphics API, diguyOsgGraphicsMesh.....	1759
5.65 DI-Guy Graphics API, diguyOsgGraphicsShaderInstance.....	1767
5.66 DI-Guy Graphics API, diguyOsgGraphicsShaderProgram.....	1770
5.67 DI-Guy Graphics API, diguyOsgGraphicsShape.....	1778
5.68 DI-Guy Graphics API, diguyOsgGraphicsState.....	1789
5.69 DI-Guy Graphics API, diguyOsgGraphicsTexture.....	1792
5.70 diguyPathShape.....	1815
5.71 diguyPluginKeyboardEvent.....	1831
5.72 diguyPluginMouseEvent.....	1834
5.73 diguyRegion.....	1838
5.74 diguyScenario.....	1852
5.75 diguyScenarioMergeSettings.....	2220
5.76 diguySceneObject.....	2266
5.77 diguySensorRegion.....	2306
5.78 diguySignal.....	2329
5.79 diguySound.....	2345
5.80 diguySoundInstance.....	2351
5.81 diguyVariable.....	2356
5.82 diguyVehicleController.....	2383
5.83 diguyView.....	2394
5.84 diguyViewButtonPanel.....	2410
5.85 diguyViewCamera.....	2435
5.86 diguyViewCameraSettings.....	2494

Table of Contents

5 Types Reference

5.87 diguyViewFog.....	2533
5.88 diguyViewFogSettings.....	2548
5.89 diguyViewLabel.....	2555
5.90 diguyViewLight.....	2599
5.91 diguyViewLightSettings.....	2627
5.92 diguyViewPainter.....	2633
5.93 diguyWaypoint.....	2649
5.94 DI-Guy AI, Main Header File.....	2689
5.95 DI-Guy API, Main Header File.....	2691
5.96 DI-Guy AI, Main Header File.....	2701
5.97 diguy_author_ui.h.....	2704
5.98 DI-Guy Changes and Version History.....	2706
5.99 diguy_changes_dox.h.....	2803
5.100 DI-Guy API, Constants and Enumerations.....	2830
5.101 DI-Guy API, DIS Constants and Enumerations.....	2879
5.102 DI-Guy API, Info Graphics Module.....	2893
5.103 DI-Guy API, OpenGL Graphics Module.....	2905
5.104 DI-Guy API, VR-Link DIS Module.....	2925
5.105 DI-Guy API, FaceFx Expressive Module.....	2964
5.106 DI-Guy API, VR-Link HLA Module.....	2967
5.107 DI-Guy API, VR-Link.....	3005
5.108 DI-Guy API, Particles Module.....	3060
5.109 DI-Guy API, Lua Script Module.....	3062
5.110 DI-Guy API, PhysX Simulation Module.....	3064
5.111 DI-Guy API, OpenAL Sound Module.....	3069
5.112 DI-Guy Release Notes.....	3073
5.113 diguy_typedefs.h.....	3085
5.114 DI-Guy Vector Classes.....	3096
5.115 Boston Dynamics Expressive Face Init Structure.....	3153
5.116 Boston Dynamics Sound Init Structure, OpenAL.....	3154
5.117 DI-Guy Graphics Init Structure, Graphics API.....	3156
5.118 DI-Guy Graphics Init Structure, OpenGL.....	3164
5.119 Boston Dynamics Log Library.....	3175

1 Types List

Listed here are the C++ classes and types that make up the DI-Guy API.

Most of the time DI-Guy classes will be in a header file with the same name, with the file being in the `include` subdirectory of the DI-Guy installation.

- `bdiCallbackHandle`
- `bdiFaceFxInit`
- `bdiGraphicsInitGraphicsAPI`
- `bdiGraphicsInitOgl`
- `bdiSoundInitOpenAL`
- `diguyAgentBehavior`
- `diguyAgentParams`
- `diguyAgentRepulsionZone`
- `diguyAgentUntrapMethod`
- `diguyAimAlgorithm`
- `diguyApp`
- `diguyAuthorInterface`
- `diguyAuthorVisualMaterial`
- `diguyAuthorVisualMultiline`
- `diguyAuthorVisualObjectHandle`
- `diguyAuthorVisualRegionMesh`
- `diguyAuthorVisualRegionPaintbrush`
- `diguyAuthorVisualUpdateFlag`
- `diguyAuthorVisualWaypoint`
- `diguyCallbackReturn`
- `diguyChainCrossSection`
- `diguyChainSettings`
- `diguyChainSimulation`
- `diguyCharacter`
- `diguyCharacterClass`
- `diguyCharacterFaceExpression`
- `diguyCharacterFeelerNumber`
- `diguyCharacterFeelerStatus`
- `diguyCharacterFeelerTurnDirection`
- `diguyCharacterGesture`
- `diguyCharacterGestureType`
- `diguyCharacterGroup`
- `diguyCharacterGuide`
- `diguyCharacterMode`
- `diguyCharacterPath`
- `diguyCharacterPathActionBead`
- `diguyCharacterPathAimBead`
- `diguyCharacterPathDecisionBead`
- `diguyCharacterPathEvent`
- `diguyCharacterPathGazeBead`
- `diguyCharacterPathScriptBead`
- `diguyCharacterPoseOverride`
- `diguyCharacterSimulator`

- diguyCharacterTypeMap
- diguyCharacterTypeMapField
- diguyCoordinateSystemType
- diguyCrowd
- diguyCrowdProfile
- diguyCrowdProfileSelectionMethod
- diguyDatetimeAdvanceMethod
- diguyDatetimeNetworkMode
- diguyDynamicAvoidanceMethod
- diguyEulerAngleOrder
- diguyExerciseInterface
- diguyFormation
- diguyGeometryTag
- diguyGraphicsFile
- diguyGraphicsIndexType
- diguyGraphicsJointType
- diguyGraphicsLink
- diguyGraphicsMaterial
- diguyGraphicsMesh
- diguyGraphicsShaderInstance
- diguyGraphicsShaderLanguage
- diguyGraphicsShaderProgram
- diguyGraphicsShaderProgramType
- diguyGraphicsShaderQualityLevel
- diguyGraphicsShaderTechnique
- diguyGraphicsShape
- diguyGraphicsState
- diguyGraphicsTexture
- diguyGraphicsTextureCompressionType
- diguyGraphicsTextureLoadingBehavior
- diguyGraphicsVertexFormat
- diguyHistoryType
- diguyIGuyController
- diguyImpact
- diguyImpactType
- diguyInfoPopup
- diguyInteractionMachine
- diguyLoadManager
- diguyModuleSimPhysXInit
- diguyMotionDirection
- diguyMotionPosture
- diguyMotionSpeedType
- diguyMotionVariant
- diguyMuzzle
- diguyNavMeshPathPlannerCostMask
- diguyNavPathAtEnd
- diguyNavPathDirection
- diguyNavPathSearchResults
- diguyNetInterface
- diguyParameterComparison
- diguyPathShape

- diguyPhysicsCollisionGroup
- diguyPhysicsSimulationMode
- diguyPluginKeyboardEvent
- diguyPluginMouseCursor
- diguyPluginMouseEvent
- diguyRegion
- diguyRegionPaintbrushMode
- diguyScenario
- diguyScenarioEditAction
- diguyScenarioInputMode
- diguyScenarioMergeFlag
- diguyScenarioMergeInitialSettings
- diguyScenarioMergeResetFlag
- diguyScenarioMergeSettings
- diguyScenarioPlaybackMode
- diguyScenarioWindowId
- diguySceneObject
- diguySensorRegion
- diguySignal
- diguySound
- diguySoundInstance
- diguySpeedUnits
- diguyStaticAvoidanceMethod
- diguySubregionIndex
- diguySubregionMask
- diguyTacticsUnitType
- diguyTextureMapMasks
- diguyTextureMapType
- diguyVariable
- diguyVec2f
- diguyVec3d
- diguyVec3f
- diguyVec3i
- diguyVec4d
- diguyVec4f
- diguyVec4i
- diguyVehicleController
- diguyVehicleSimulationMode
- diguyView
- diguyViewButtonPanel
- diguyViewCamera
- diguyViewCameraMoveMode
- diguyViewCameraProjectionMode
- diguyViewCameraSettings
- diguyViewFog
- diguyViewFogMode
- diguyViewFogSettings
- diguyViewLabel
- diguyViewLabelJustification
- diguyViewLight
- diguyViewLightSettings

- [diguyViewPainter](#)
- [diguyVisibleFlag](#)
- [diguyWaypoint](#)

2 Files List

Listed here are the C++ source and header files that make up the DI-Guy API.

Header (.h) files are files that will be included in C++ programs that use the DI-Guy SDK. Most are found in the `include` subdirectory of the DI-Guy installation.

C++ source (.cpp) files show example code from programming examples. The programming example files are found in the `programming_examples` subdirectory.

- [diguyAgentParams.h](#) (diguyAgentParams)
- [diguyApp.h](#) (diguyApp)
- [diguyAuthorInterface.h](#) (diguyAuthorInterface)
- [diguyAuthorVisualMaterial.h](#) (diguyAuthorVisualMaterial)
- [diguyAuthorVisualMultiline.h](#) (diguyAuthorVisualMultiline)
- [diguyAuthorVisualObjectHandle.h](#) (diguyAuthorVisualObjectHandle)
- [diguyAuthorVisualRegionMesh.h](#) (diguyAuthorVisualRegionMesh)
- [diguyAuthorVisualRegionPaintbrush.h](#) (diguyAuthorVisualRegionPaintbrush)
- [diguyAuthorVisualWaypoint.h](#) (diguyAuthorVisualWaypoint)
- [diguyChainSettings.h](#) (diguyChainSettings)
- [diguyChainSimulation.h](#) (diguyChainSimulation)
- [diguyCharacter.h](#) (diguyCharacter)
- [diguyCharacterFaceExpression.h](#) (diguyCharacterFaceExpression)
- [diguyCharacterGesture.h](#) (diguyCharacterGesture)
- [diguyCharacterGestureType.h](#) (diguyCharacterGestureType Type Declaration)
- [diguyCharacterGroup.h](#) (diguyCharacterGroup)
- [diguyCharacterGuide.h](#) (diguyCharacterGuide)
- [diguyCharacterPath.h](#) (diguyCharacterPath)
- [diguyCharacterPathActionBead.h](#) (diguyCharacterPathActionBead)
- [diguyCharacterPathAimBead.h](#) (diguyCharacterPathAimBead)
- [diguyCharacterPathDecisionBead.h](#) (diguyCharacterPathDecisionBead)
- [diguyCharacterPathEvent.h](#) (diguyCharacterPathEvent)
- [diguyCharacterPathGazeBead.h](#) (diguyCharacterPathGazeBead)
- [diguyCharacterPathScriptBead.h](#) (diguyCharacterPathScriptBead)
- [diguyCharacterPoseOverride.h](#) (diguyCharacterPoseOverride)
- [diguyCharacterTypeMap.h](#) (diguyCharacterTypeMap)
- [diguyCrowd.h](#) (diguyCrowd)
- [diguyCrowdProfile.h](#) (diguyCrowdProfile)
- [diguyExerciseInterface.h](#) (diguyExerciseInterface)
- [diguyFormation.h](#) (diguyFormation)
- [diguyGraphicsAPIOverview.h](#) (DI-Guy Graphics API Programming Overview)
- [diguyGraphicsFile.h](#) (diguyGraphicsFile)
- [diguyGraphicsLink.h](#) (diguyGraphicsLink)
- [diguyGraphicsMaterial.h](#) (diguyGraphicsMaterial)
- [diguyGraphicsMesh.h](#) (diguyGraphicsMesh)
- [diguyGraphicsShaderInstance.h](#) (diguyGraphicsShaderInstance)
- [diguyGraphicsShaderProgram.h](#) (diguyGraphicsShaderProgram)
- [diguyGraphicsShaderTechnique.h](#) (diguyGraphicsShaderTechnique)
- [diguyGraphicsShape.h](#) (diguyGraphicsShape)
- [diguyGraphicsState.h](#) (diguyGraphicsState)

- [diguyGraphicsTexture.h](#) (diguyGraphicsTexture)
- [diguyGraphicsVertexFormats.h](#) (diguyGraphicsVertexFormats)
- [diguyIGuyController.h](#) (diguyIGuyController)
- [diguyImpact.h](#) (diguyImpact)
- [diguyInfoPopup.h](#) (diguyInfoPopup)
- [diguyInteractionMachine.h](#) (diguyInteractionMachine)
- [diguyLoadManager.h](#) (diguyLoadManager)
- [diguyMotionDirection.h](#) (diguyMotionDirection Type Declaration)
- [diguyMotionPosture.h](#) (diguyMotionPosture Type Declaration)
- [diguyMotionSpeedType.h](#) (diguyMotionSpeedType Type Declaration)
- [diguyMotionVariant.h](#) (diguyMotionVariant Type Declaration)
- [diguyPathShape.h](#) (diguyPathShape)
- [diguyPluginKeyboardEvent.h](#) (diguyPluginKeyboardEvent)
- [diguyPluginMouseEvent.h](#) (diguyPluginMouseEvent)
- [diguyRegion.h](#) (diguyRegion)
- [diguyScenario.h](#) (diguyScenario)
- [diguyScenarioMergeSettings.h](#) (diguyScenarioMergeSettings)
- [diguySceneObject.h](#) (diguySceneObject)
- [diguySensorRegion.h](#) (diguySensorRegion)
- [diguySignal.h](#) (diguySignal)
- [diguySound.h](#) (diguySound)
- [diguySoundInstance.h](#) (diguySoundInstance)
- [diguyVariable.h](#) (diguyVariable)
- [diguyVehicleController.h](#) (diguyVehicleController)
- [diguyView.h](#) (diguyView)
- [diguyViewButtonPanel.h](#) (diguyViewButtonPanel)
- [diguyViewCamera.h](#) (diguyViewCamera)
- [diguyViewCameraSettings.h](#) (diguyViewCameraSettings)
- [diguyViewFog.h](#) (diguyViewFog)
- [diguyViewFogSettings.h](#) (diguyViewFogSettings)
- [diguyViewLabel.h](#) (diguyViewLabel)
- [diguyViewLight.h](#) (diguyViewLight)
- [diguyViewLightSettings.h](#) (diguyViewLightSettings)
- [diguyViewPainter.h](#) (diguyViewPainter)
- [diguyWaypoint.h](#) (diguyWaypoint)
- [diguy_ai.h](#) (DI-Guy AI, Main Header File)
- [diguy_api.h](#) (DI-Guy API, Main Header File)
- [diguy_author.h](#) (DI-Guy AI, Main Header File)
- [diguy_author_ui.h](#)
- [diguy_changes.h](#) (DI-Guy Changes and Version History)
- [diguy_changes_dox.h](#)
- [diguy_constants.h](#) (DI-Guy API, Constants and Enumerations)
- [diguy_dis_constants.h](#) (DI-Guy API, DIS Constants and Enumerations)
- [diguy_graphics_api.h](#) (DI-Guy API, Info Graphics Module)
- [diguy_graphics_ogl.h](#) (DI-Guy API, OpenGL Graphics Module)
- [diguy_module_dis.h](#) (DI-Guy API, VR-Link DIS Module)
- [diguy_module_facefx.h](#) (DI-Guy API, FaceFx Expressive Module)
- [diguy_module_hla.h](#) (DI-Guy API, VR-Link HLA Module)
- [diguy_module_net.h](#) (DI-Guy API, VR-Link)
- [diguy_module_particles.h](#) (DI-Guy API, Particles Module)
- [diguy_module_script_lua.h](#) (DI-Guy API, Lua Script Module)

DI-Guy SDK Version 12.5.2 Index

- [diguy_module_sim_physx.h](#) (DI-Guy API, PhysX Simulation Module)
- [diguy_module_sound_openal.h](#) (DI-Guy API, OpenAL Sound Module)
- [diguy_release_notes.h](#) (DI-Guy Release Notes)
- [diguy_typedefs.h](#)
- [diguy_vector_classes.h](#) (DI-Guy Vector Classes)
- [bdiFaceFxInit.h](#) (Boston Dynamics Expressive Face Init Structure)
- [bdiSoundInitOpenAL.h](#) (Boston Dynamics Sound Init Structure, OpenAL)
- [libbdilog.h](#) (Boston Dynamics Log Library)
- [bdiGraphicsInitGraphicsAPI.h](#) (DI-Guy Graphics Init Structure, Graphics API)
- [graphics_init_ogl.h](#) (DI-Guy Graphics Init Structure, OpenGL)
- [simple_playback_ogl.cpp](#) (DI-Guy Graphics API, Immediate Mode with DI-Guy Loader Example)
- [simple_playback_osg_2.cpp](#) (DI-Guy Graphics API, Scene Graph with DI-Guy Loader Example)
- [simple_playback_osg_2_ext.cpp](#) (DI-Guy Graphics API, Scene Graph with External Loader Example)
- [diguyOglGraphicsLink.cpp](#) (DI-Guy Graphics API, diguyOglGraphicsLink)
- [diguyOglGraphicsMaterial.cpp](#) (DI-Guy Graphics API, diguyOglGraphicsMaterial)
- [diguyOglGraphicsMesh.cpp](#) (DI-Guy Graphics API, diguyOglGraphicsMesh)
- [diguyOglGraphicsShaderInstance.cpp](#) (DI-Guy Graphics API, diguyOglGraphicsShaderInstance)
- [diguyOglGraphicsShaderProgram.cpp](#) (DI-Guy Graphics API, diguyOglGraphicsShaderProgram)
- [diguyOglGraphicsShaderTechnique.cpp](#)
- [diguyOglGraphicsShape.cpp](#) (DI-Guy Graphics API, diguyOglGraphicsShape)
- [diguyOglGraphicsTexture.cpp](#) (DI-Guy Graphics API, diguyOglGraphicsTexture)
- [diguyOsgExtGraphicsFile.cpp](#) (DI-Guy Graphics API, diguyOsgExtGraphicsFile)
- [diguyOsgExtGraphicsLink.cpp](#) (DI-Guy Graphics API, diguyOsgExtGraphicsLink)
- [diguyOsgExtGraphicsShape.cpp](#) (DI-Guy Graphics API, diguyOsgExtGraphicsShape)
- [diguyOsgGraphicsLink.cpp](#) (DI-Guy Graphics API, diguyOsgGraphicsLink)
- [diguyOsgGraphicsMaterial.cpp](#) (DI-Guy Graphics API, diguyOsgGraphicsMaterial)
- [diguyOsgGraphicsMesh.cpp](#) (DI-Guy Graphics API, diguyOsgGraphicsMesh)
- [diguyOsgGraphicsShaderInstance.cpp](#) (DI-Guy Graphics API, diguyOsgGraphicsShaderInstance)
- [diguyOsgGraphicsShaderProgram.cpp](#) (DI-Guy Graphics API, diguyOsgGraphicsShaderProgram)
- [diguyOsgGraphicsShape.cpp](#) (DI-Guy Graphics API, diguyOsgGraphicsShape)
- [diguyOsgGraphicsState.cpp](#) (DI-Guy Graphics API, diguyOsgGraphicsState)
- [diguyOsgGraphicsTexture.cpp](#) (DI-Guy Graphics API, diguyOsgGraphicsTexture)
- [diguyOsgUtil.cpp](#)

3 Functions List

Listed here alphabetically are all of the functions in the DI-Guy API.

- `abort_all_dof_targets` (diguySceneObject)
- `abort_all_gestures` (diguyCharacter)
- `abort_all_switch_targets` (diguySceneObject)
- `abort_executing_gesture` (diguyCharacter)
- `abort_now` (diguyCharacterGesture)
- `add_additional_shader` (diguyGraphicsShaderTechnique)
- `add_all_characters_as_members` (diguyCharacterGroup)
- `add_all_characters_in_group` (diguyCrowd)
- `add_appearance` (diguyScenario)
- `add_appearance_based_on_existing_appearance` (diguyScenario)
- `add_button` (diguyViewButtonPanel)
- `add_button_with_script` (diguyViewButtonPanel)
- `add_callback` (diguyCharacter)
- `add_callback` (diguyCharacterGesture)
- `add_callback` (diguyCharacterPath)
- `add_callback` (diguyCrowd)
- `add_callback` (diguyScenario)
- `add_callback` (diguySensorRegion)
- `add_callback` (diguySignal)
- `add_callback` (diguyVariable)
- `add_callback` (diguyView)
- `add_callback` (diguyViewCamera)
- `add_callback` (diguyViewFog)
- `add_callback` (diguyViewLabel)
- `add_callback` (diguyViewLight)
- `add_callback_script` (diguyCharacter)
- `add_callback_script` (diguyCharacterGesture)
- `add_callback_script` (diguyCrowd)
- `add_callback_script` (diguyScenario)
- `add_callback_script` (diguySensorRegion)
- `add_callback_script` (diguySignal)
- `add_callback_script` (diguyVariable)
- `add_callback_script` (diguyView)
- `add_callback_script` (diguyViewCamera)
- `add_callback_script` (diguyViewFog)
- `add_callback_script` (diguyViewLabel)
- `add_callback_script` (diguyViewLight)
- `add_candidate_characters_in_region_to_group` (diguySensorRegion)
- `add_channel_weight_target` (diguyCharacterGesture)
- `add_character_to_octtree` (diguyScenario)
- `add_characters_in_region_to_group` (diguySensorRegion)
- `add_companion_crowd` (diguyCrowd)
- `add_default_callback` (diguyScenario)
- `add_default_callback_script` (diguyScenario)
- `add_default_camera_callback` (diguyScenario)
- `add_default_camera_callback_script` (diguyScenario)

- `add_default_character_callback` (diguyScenario)
- `add_default_character_callback_script` (diguyScenario)
- `add_default_fog_callback` (diguyScenario)
- `add_default_fog_callback_script` (diguyScenario)
- `add_default_light_callback` (diguyScenario)
- `add_default_light_callback_script` (diguyScenario)
- `add_default_path_callback` (diguyScenario)
- `add_default_scenario_callback` (diguyApp)
- `add_default_scenario_callback_script` (diguyApp)
- `add_default_sensor_region_callback` (diguyScenario)
- `add_default_sensor_region_callback_script` (diguyScenario)
- `add_default_signal_callback` (diguyScenario)
- `add_default_signal_callback_script` (diguyScenario)
- `add_default_variable_callback` (diguyScenario)
- `add_default_variable_callback_script` (diguyScenario)
- `add_default_view_callback` (diguyScenario)
- `add_default_view_callback_script` (diguyScenario)
- `add_dof_target` (diguySceneObject)
- `add_dof_targetflt` (diguySceneObject)
- `add_entry` (diguyCrowdProfile)
- `add_float_slider` (diguyViewButtonPanel)
- `add_guide` (diguyCharacter)
- `add_input` (diguyInteractionMachine)
- `add_int_slider` (diguyViewButtonPanel)
- `add_member` (diguyCharacterGroup)
- `add_member` (diguyCrowd)
- `add_motion_texture` (diguyCharacter)
- `add_pending_desired_action` (diguyCharacter)
- `add_pending_force_action` (diguyCharacter)
- `add_secondary_heartbeat` (diguyCharacter)
- `add_spacer` (diguyViewButtonPanel)
- `add_state` (diguyInteractionMachine)
- `add_switch_target` (diguySceneObject)
- `add_triangle_array_to_user_defined` (diguySceneObject)
- `add_triangle_to_user_defined` (diguySceneObject)
- `add_vertex_array_to_user_defined` (diguySceneObject)
- `add_vertex_to_user_defined` (diguySceneObject)
- `add_wakeup_callback` (diguyCharacter)
- `add_waypoint` (diguyCharacterPath)
- `add_waypoint` (diguyPathShape)
- `agent_accept_message` (diguyCharacter)
- `agent_attack_character` (diguyCharacter)
- `agent_attack_group` (diguyCharacter)
- `agent_begin_behavior` (diguyCharacter)
- `agent_broadcast_message` (diguyCharacter)
- `agent_broadcast_message_to_group` (diguyCharacter)
- `agent_call_formation` (diguyCharacter)
- `agent_flee_character` (diguyCharacter)
- `agent_flee_crowd` (diguyCharacter)
- `agent_flee_group` (diguyCharacter)
- `agent_flee_location` (diguyCharacter)

- [agent_get_attack_zone](#) (diguyCharacter)
- [agent_get_attack_zone_enabled](#) (diguyCharacter)
- [agent_get_auto_variant_selection](#) (diguyCharacter)
- [agent_get_crowd](#) (diguyCharacter)
- [agent_get_current_focus_character](#) (diguyCharacter)
- [agent_get_current_params](#) (diguyCharacter)
- [agent_get_current_params_profile_name](#) (diguyCharacter)
- [agent_get_feeler_function](#) (diguyCharacter)
- [agent_get_initial_params](#) (diguyCharacter)
- [agent_get_path_follow_path](#) (diguyCharacter)
- [agent_get_path_follow_path_t](#) (diguyCharacter)
- [agent_get_path_follow_path_t_paused](#) (diguyCharacter)
- [agent_mingle_in_region](#) (diguyCharacter)
- [agent_move_on](#) (diguyCharacter)
- [agent_move_to_point](#) (diguyCharacter)
- [agent_move_to_point_bg](#) (diguyCharacter)
- [agent_move_to_point_via_subregions](#) (diguyCharacter)
- [agent_move_to_point_via_subregions_bg](#) (diguyCharacter)
- [agent_move_to_region](#) (diguyCharacter)
- [agent_move_to_region_via_subregions](#) (diguyCharacter)
- [agent_path_follow](#) (diguyCharacter)
- [agent_pursue_character](#) (diguyCharacter)
- [agent_pursue_character_with_offset](#) (diguyCharacter)
- [agent_pursue_group](#) (diguyCharacter)
- [agent_remove_all_flee_objects](#) (diguyCharacter)
- [agent_resume_path_follow](#) (diguyCharacter)
- [agent_set_attack_zone](#) (diguyCharacter)
- [agent_set_attack_zone_enabled](#) (diguyCharacter)
- [agent_set_auto_variant_selection](#) (diguyCharacter)
- [agent_set_current_behavior_region_border_is_solid](#) (diguyCharacter)
- [agent_set_current_focus_character](#) (diguyCharacter)
- [agent_set_current_focus_group](#) (diguyCharacter)
- [agent_set_current_params_from_initial_params](#) (diguyCharacter)
- [agent_set_current_params_from_profile](#) (diguyCharacter)
- [agent_set_feeler_function](#) (diguyCharacter)
- [agent_set_initial_params_from_profile](#) (diguyCharacter)
- [agent_stop_behavior](#) (diguyCharacter)
- [agent_travel_behavior_path](#) (diguyCharacter)
- [agent_travel_path_shape](#) (diguyCharacter)
- [agent_travel_region_border](#) (diguyCharacter)
- [agent_wander_region](#) (diguyCharacter)
- [agents_attack_character](#) (diguyCrowd)
- [agents_attack_group](#) (diguyCrowd)
- [agents_chain_pursue](#) (diguyCrowd)
- [agents_flee_character](#) (diguyCrowd)
- [agents_flee_crowd](#) (diguyCrowd)
- [agents_flee_group](#) (diguyCrowd)
- [agents_flee_location](#) (diguyCrowd)
- [agents_mingle_in_region](#) (diguyCrowd)
- [agents_pursue_character](#) (diguyCrowd)
- [agents_pursue_group](#) (diguyCrowd)

- `agents_remove_all_flee_objects` (diguyCrowd)
- `agents_stop_behavior` (diguyCrowd)
- `agents_travel_behavior_path` (diguyCrowd)
- `agents_travel_path_shape` (diguyCrowd)
- `agents_travel_region_border` (diguyCrowd)
- `agents_wander_region` (diguyCrowd)
- `aim_at_angle` (diguyCharacter)
- `aim_at_angle_local` (diguyCharacter)
- `aim_at_character` (diguyCharacter)
- `aim_at_point` (diguyCharacter)
- `aim_at_point_local` (diguyCharacter)
- `aim_converge` (diguyCharacter)
- `all_members_add_character_callback` (diguyCrowd)
- `all_members_add_character_callback_script` (diguyCrowd)
- `all_members_remove_character_callback` (diguyCrowd)
- `all_members_remove_character_callback_script` (diguyCrowd)
- `all_members_remove_character_callback_with_user_data` (diguyCrowd)
- `allocate_pose_array` (diguyCharacterPoseOverride)
- `allocate_var_names` (diguyCharacterPoseOverride)
- `appearance_effect_is_active` (diguyCharacter)
- `append_file` (diguyInfoPopup)
- `append_string` (diguyInfoPopup)
- `apply` (diguyViewLight)
- `apply_aim_bead` (diguyCharacter)
- `apply_decision_bead` (diguyCharacter)
- `apply_formation_to_group` (diguyCharacterGroup)
- `apply_formation_to_group_with_dir` (diguyCharacterGroup)
- `apply_formation_to_group_with_goal` (diguyCharacterGroup)
- `apply_gaze_bead` (diguyCharacter)
- `apply_impulse_to_physics_sim` (diguyCharacter)
- `apply_impulse_to_physics_sim_from_world_position` (diguyCharacter)
- `apply_mouse_inputs` (diguyAuthorInterface)
- `apply_script_bead` (diguyCharacter)
- `apply_settings` (diguyViewCamera)
- `attach_to_link` (diguyGraphicsLink)
- `attach_to_scene` (diguyGraphicsLink)
- `author_ui_window_visibility_changed` (diguyAuthorInterface)
- `bdi_log_file_disable`
- `bdi_log_file_enable`
- `bdi_log_get_global_notify_level`
- `bdi_log_get_notify_level`
- `bdi_log_get_num_logs_open`
- `bdi_log_get_string_from_notify_level`
- `bdi_log_lock`
- `bdi_log_print`
- `bdi_log_print_to_output_window`
- `bdi_log_register_callback`
- `bdi_log_register_fatal_exit_function`
- `bdi_log_set_global_notify_level`
- `bdi_log_set_notify_level`
- `bdi_log_stderr_disable`

- `bdi_log_stderr_enable`
- `bdi_log_stdout_disable`
- `bdi_log_stdout_enable`
- `bdi_log_unlock`
- `bdi_log_unregister_callback`
- `begin_2d_text_mode` (diguyViewPainter)
- `begin_appearance_effect` (diguyCharacter)
- `begin_character_draw` (diguyGraphicsLink)
- `begin_interaction` (diguyInteractionMachine)
- `begin_legacy_appearance_effect` (diguyCharacter)
- `begin_now` (diguyCharacterGesture)
- `bind` (diguyGraphicsShaderInstance)
- `bind` (diguyGraphicsShaderProgram)
- `bind_color_array` (diguyGraphicsShaderProgram)
- `bind_link_matrices` (diguyGraphicsShaderProgram)
- `bind_now` (diguyGraphicsMaterial)
- `bind_now` (diguyGraphicsState)
- `bind_now` (diguyGraphicsTexture)
- `bind_uniform_with_location` (diguyGraphicsShaderProgram)
- `blink` (diguyCharacter)
- `break_formation` (diguyCharacter)
- `broadcast_message` (diguyExerciseInterface)
- `broadcast_message_to_group` (diguyExerciseInterface)
- `build` (diguyAuthorVisualMultiline)
- `build` (diguyAuthorVisualObjectHandle)
- `build` (diguyAuthorVisualRegionMesh)
- `build` (diguyAuthorVisualRegionPaintbrush)
- `build` (diguyAuthorVisualWaypoint)
- `build` (diguyGraphicsLink)
- `build` (diguyGraphicsMaterial)
- `build` (diguyGraphicsMesh)
- `build` (diguyGraphicsShaderTechnique)
- `build` (diguyGraphicsShape)
- `build` (diguyGraphicsState)
- `build` (diguyGraphicsTexture)
- `build_user_defined` (diguySceneObject)
- `call_formation` (diguyCharacter)
- `check_character_visibility` (diguyCharacter)
- `check_point_visibility` (diguyCharacter)
- `check_visibility` (diguyScenario)
- `check_z_clearance` (diguyAuthorInterface)
- `clear` (diguyCharacterFaceExpression)
- `clear` (diguyInfoPopup)
- `clear_inputs` (diguyInteractionMachine)
- `clear_last_bound_material` (diguyGraphicsMaterial)
- `clear_last_bound_texture` (diguyGraphicsTexture)
- `clear_region` (diguyRegion)
- `clear_states` (diguyInteractionMachine)
- `clear_subregion` (diguyRegion)
- `clear_transient_script_events` (diguyScenario)
- `contains_character` (diguySensorRegion)

- contains_character_by_type (diguySensorRegion)
- contains_character_by_type_and_appearance (diguySensorRegion)
- contains_character_from_group (diguySensorRegion)
- contains_character_link (diguySensorRegion)
- contains_point (diguySensorRegion)
- convert_lat_long_to_local (diguyExerciseInterface)
- convert_lat_long_to_local (diguyNetInterface)
- convert_local_to_lat (diguyExerciseInterface)
- convert_local_to_lat (diguyNetInterface)
- convert_local_to_lat_long_string (diguyExerciseInterface)
- convert_local_to_long (diguyExerciseInterface)
- convert_local_to_long (diguyNetInterface)
- convert_local_to_mgrs (diguyExerciseInterface)
- convert_mgrs_to_local (diguyExerciseInterface)
- convert_mgrs_to_local (diguyNetInterface)
- convert_screen_to_world (diguyAuthorInterface)
- convert_world_to_screen (diguyAuthorInterface)
- copy (diguyVariable)
- copy_path_shape (diguyScenario)
- create_action_bead (diguyCharacterPath)
- create_and_force_bridge_path (diguyCharacter)
- create_and_force_bridge_path_to_point (diguyCharacter)
- create_chain_settings (diguyScenario)
- create_character (diguyCrowdProfile)
- create_character (diguyScenario)
- create_complete_pose_override (diguyCharacter)
- create_crowd (diguyScenario)
- create_crowd_profile (diguyScenario)
- create_entry (diguyCrowdProfile)
- create_face_expression (diguyScenario)
- create_formation (diguyScenario)
- create_gesture (diguyCharacter)
- create_group (diguyScenario)
- create_guide (diguyCharacter)
- create_interaction_machine (diguyScenario)
- create_log_channel (diguyApp)
- create_lua_mind (diguyCharacter)
- create_merge_settings (diguyScenario)
- create_or_update_companion_waypoint (diguyCharacterPathActionBead)
- create_particle_system (diguyScenario)
- create_particle_system_with_orientation (diguyScenario)
- create_path (diguyCharacter)
- create_path_from_path_shape (diguyCharacter)
- create_path_shape (diguyScenario)
- create_pending_reflected_character (diguyScenario)
- create_pose_override (diguyCharacter)
- create_region (diguyScenario)
- create_scene_object (diguyScenario)
- create_script_bead (diguyCharacterPath)
- create_signal (diguyScenario)
- create_simple_path (diguyCharacter)

- `create_sound` (diguyScenario)
- `create_sound_instance` (diguyCharacter)
- `create_temporary_character` (diguyScenario)
- `create_user_defined_scene_object` (diguyScenario)
- `create_waypoint` (diguyCharacterPath)
- `create_waypoint` (diguyPathShape)
- `create_waypoint` (diguyScenario)
- `decrease_speed` (diguyViewCamera)
- `decrement_value_as_int` (diguyVariable)
- `delete_generated_vertices` (diguyAuthorInterface)
- `derive_diguy_shader_source_filename` (diguyGraphicsShaderProgram)
- `derive_playback_mode_dt` (diguyScenario)
- `derive_render_mode_from_light_settings` (diguyScenario)
- `destroy_action_bead` (diguyCharacterPath)
- `destroy_all` (diguyViewButtonPanel)
- `destroy_chain_settings` (diguyScenario)
- `destroy_character` (diguyScenario)
- `destroy_complete_pose_override` (diguyCharacter)
- `destroy_crowd` (diguyScenario)
- `destroy_crowd_profile` (diguyScenario)
- `destroy_entry` (diguyCrowdProfile)
- `destroy_face_expression` (diguyScenario)
- `destroy_formation` (diguyScenario)
- `destroy_gesture` (diguyCharacter)
- `destroy_group` (diguyScenario)
- `destroy_instance` (diguyViewButtonPanel)
- `destroy_interaction_machine` (diguyScenario)
- `destroy_label` (diguyScenario)
- `destroy_log_channel` (diguyApp)
- `destroy_merge_settings` (diguyScenario)
- `destroy_panel` (diguyScenario)
- `destroy_path` (diguyCharacter)
- `destroy_path_shape` (diguyScenario)
- `destroy_pose_override` (diguyCharacter)
- `destroy_region` (diguyScenario)
- `destroy_scene_object` (diguyScenario)
- `destroy_script_bead` (diguyCharacterPath)
- `destroy_signal` (diguyScenario)
- `destroy_sound` (diguyScenario)
- `destroy_sound_instance` (diguyCharacter)
- `destroy_variable` (diguyCharacter)
- `destroy_variable` (diguyCharacterGroup)
- `destroy_variable` (diguyScenario)
- `destroy_waypoint` (diguyScenario)
- `detach_from_link` (diguyGraphicsLink)
- `detach_from_scene` (diguyGraphicsLink)
- `die_now` (diguyCharacter)
- `diguyVec2f` (diguyVec2f)
- `diguyVec3d` (diguyVec3d)
- `diguyVec3f` (diguyVec3f)
- `diguyVec3i` (diguyVec3i)

- diguyVec4d (diguyVec4d)
- diguyVec4f (diguyVec4f)
- diguyVec4i (diguyVec4i)
- diguy_ai_deinitialize
- diguy_ai_initialize
- diguy_author_get_author_enabled
- diguy_author_get_author_interface
- diguy_author_initialize
- diguy_author_ui_deinitialize
- diguy_author_ui_initialize
- diguy_author_ui_process_ui_events
- diguy_character_type_map_field_from_string
- diguy_character_type_map_field_to_string
- diguy_convert_euler_angle_order
- diguy_convert_euler_angles_to_quaternion
- diguy_create_matrix_from_translation_and_euler_angles
- diguy_create_scenario
- diguy_default_fire_weapon_intersection_function
- diguy_deinitialize
- diguy_destroy_scenario
- diguy_dis_deinitialize
- diguy_dis_enter_vpnet_mode
- diguy_dis_establish_new_offsets
- diguy_dis_get_character_from_id
- diguy_dis_get_display_remote_chars
- diguy_dis_get_earth_coord
- diguy_dis_get_exercise
- diguy_dis_get_host
- diguy_dis_get_ipaddr
- diguy_dis_get_lat_lon
- diguy_dis_get_model_from_septet
- diguy_dis_get_name_from_entity
- diguy_dis_get_next_participant_name
- diguy_dis_get_online
- diguy_dis_get_port
- diguy_dis_get_position_offset
- diguy_dis_get_publish_local_chars
- diguy_dis_get_send_custom_pdus
- diguy_dis_get_septet_from_model
- diguy_dis_get_site
- diguy_dis_get_sync_mode_camera_offset
- diguy_dis_get_sync_mode_camera_slaved
- diguy_dis_get_sync_mode_enabled
- diguy_dis_get_sync_mode_master
- diguy_dis_get_sync_mode_slave_cam
- diguy_dis_get_use_user_defined_coords
- diguy_dis_get_utm
- diguy_dis_go_offline
- diguy_dis_go_online
- diguy_dis_initialize
- diguy_dis_set_camera_details_out_of_date

- diguy_dis_set_cfg_file
- diguy_dis_set_client_mode
- diguy_dis_set_display_remote_chars
- diguy_dis_set_exercise
- diguy_dis_set_host
- diguy_dis_set_ipaddr
- diguy_dis_set_lat_lon
- diguy_dis_set_port
- diguy_dis_set_position_offset
- diguy_dis_set_publish_local_chars
- diguy_dis_set_send_custom_pdus
- diguy_dis_set_server_mode
- diguy_dis_set_site
- diguy_dis_set_sync_mode_camera_offset
- diguy_dis_set_sync_mode_camera_slaved
- diguy_dis_set_sync_mode_enabled
- diguy_dis_set_sync_mode_master
- diguy_dis_set_sync_mode_slave_cam
- diguy_dis_set_use_database_coords
- diguy_dis_set_use_user_defined_coords
- diguy_dis_set_utm
- diguy_dis_set_utm_zone
- diguy_dis_toggle_earth_coord_format
- diguy_dis_update
- diguy_euler_angles_to_quaternion (diguyVec3f)
- diguy_extract_translation_and_euler_angles_from_matrix
- diguy_facefx_deinitialize
- diguy_facefx_get_defaults
- diguy_facefx_initialize
- diguy_format_lat_or_lon
- diguy_gesture_type_from_string
- diguy_gesture_type_to_string
- diguy_get_app
- diguy_graphics_get_defaults
- diguy_graphics_get_dx9_defaults
- diguy_graphics_get_osg_defaults
- diguy_graphics_get_use_far_positions
- diguy_graphics_initialize
- diguy_graphics_set_file_create_func
- diguy_graphics_set_link_create_func
- diguy_graphics_set_material_create_func
- diguy_graphics_set_mesh_create_func
- diguy_graphics_setPainter_create_func
- diguy_graphics_set_shader_instance_create_func
- diguy_graphics_set_shader_program_create_func
- diguy_graphics_set_shader_technique_create_func
- diguy_graphics_set_shape_create_func
- diguy_graphics_set_state_create_func
- diguy_graphics_set_texture_create_func
- diguy_graphics_set_use_far_positions
- diguy_hla_deinitialize

- diguy_hla_enter_vpnet_mode
- diguy_hla_establish_new_offsets
- diguy_hla_get_character_from_id
- diguy_hla_get_display_remote_chars
- diguy_hla_get_earth_coord
- diguy_hla_get_exercise
- diguy_hla_get_host
- diguy_hla_get_ipaddr
- diguy_hla_get_lat_lon
- diguy_hla_get_model_from_septet
- diguy_hla_get_name_from_entity
- diguy_hla_get_next_participant_name
- diguy_hla_get_online
- diguy_hla_get_port
- diguy_hla_get_position_offset
- diguy_hla_get_publish_local_chars
- diguy_hla_get_send_custom_pdus
- diguy_hla_get_septet_from_model
- diguy_hla_get_site
- diguy_hla_get_sync_mode_camera_offset
- diguy_hla_get_sync_mode_camera_slaved
- diguy_hla_get_sync_mode_enabled
- diguy_hla_get_sync_mode_master
- diguy_hla_get_sync_mode_slave_cam
- diguy_hla_get_use_user_defined_coords
- diguy_hla_get_utm
- diguy_hla_go_offline
- diguy_hla_go_online
- diguy_hla_initialize
- diguy_hla_set_camera_details_out_of_date
- diguy_hla_set_cfg_file
- diguy_hla_set_display_remote_chars
- diguy_hla_set_exercise
- diguy_hla_set_host
- diguy_hla_set_ipaddr
- diguy_hla_set_lat_lon
- diguy_hla_set_port
- diguy_hla_set_position_offset
- diguy_hla_set_publish_local_chars
- diguy_hla_set_send_custom_pdus
- diguy_hla_set_site
- diguy_hla_set_sync_mode_camera_offset
- diguy_hla_set_sync_mode_camera_slaved
- diguy_hla_set_sync_mode_enabled
- diguy_hla_set_sync_mode_master
- diguy_hla_set_sync_mode_slave_cam
- diguy_hla_set_use_database_coords
- diguy_hla_set_use_user_defined_coords
- diguy_hla_set_utm
- diguy_hla_set_utm_zone
- diguy_hla_toggle_earth_coord_format

- diguy_hla_update
- diguy_motion_direction_from_angle
- diguy_motion_direction_from_string
- diguy_motion_direction_to_angle
- diguy_motion_direction_to_string
- diguy_motion_posture_from_string
- diguy_motion_posture_to_string
- diguy_motion_speed_type_from_string
- diguy_motion_speed_type_to_string
- diguy_motion_variant_from_string
- diguy_motion_variant_to_string
- diguy_ogl_2d_draw_filled_rect
- diguy_ogl_2d_draw_text
- diguy_ogl_2d_get_text_height
- diguy_ogl_2d_get_text_width
- diguy_ogl_2d_set_color
- diguy_ogl_apply_active_lights
- diguy_ogl_clear_all_frame_stats
- diguy_ogl_draw_frame_stats
- diguy_ogl_frame_stat_name
- diguy_ogl_generate_character_texture_list
- diguy_ogl_get_character_texture_list_num_textures
- diguy_ogl_get_character_texture_list_texture_filename_at_index
- diguy_ogl_get_character_texture_list_texture_object_at_index
- diguy_ogl_get_defaults
- diguy_ogl_get_fbo_texture_id
- diguy_ogl_get_frame_stat
- diguy_ogl_get_num_frame_stats
- diguy_ogl_get_texture_unit
- diguy_ogl_get_use_far_positions
- diguy_ogl_initialize
- diguy_ogl_release_character_texture_list
- diguy_ogl_reload_changed_textures
- diguy_ogl_render_shadow_map
- diguy_ogl_set_character_texture_objects_modified
- diguy_ogl_set_default_shader_source_files
- diguy_ogl_set_texture_unit
- diguy_ogl_set_use_far_positions
- diguy_parse_dms_to_radians
- diguy_particles_deinitialize
- diguy_particles_initialize
- diguy_script_lua_deinitialize
- diguy_script_lua_initialize
- diguy_set_cfg_file
- diguy_set_diguy_base_dir_path
- diguy_sim_physx_deinitialize
- diguy_sim_physx_initialize
- diguy_sound_openal_deinitialize
- diguy_sound_openal_get_defaults
- diguy_sound_openal_initialize
- diguy_to_y_forward_z_up_coords (diguyVec3f)

- [diguy_to_z_forward_y_up_coords \(diguyVec3f\)](#)
- [disable_bar_graph \(diguyViewLabel\)](#)
- [do_mind_current_state_changed_callback \(diguyCharacter\)](#)
- [do_stabilization_time_runoff \(diguyChainSimulation\)](#)
- [draw \(diguyAuthorVisualMultiline\)](#)
- [draw \(diguyAuthorVisualObjectHandle\)](#)
- [draw \(diguyAuthorVisualRegionMesh\)](#)
- [draw \(diguyAuthorVisualRegionPaintbrush\)](#)
- [draw \(diguyAuthorVisualWaypoint\)](#)
- [draw \(diguyCharacter\)](#)
- [draw \(diguyGraphicsLink\)](#)
- [draw \(diguyGraphicsMesh\)](#)
- [draw \(diguyGraphicsShape\)](#)
- [draw \(diguyScenario\)](#)
- [draw_2d_rect \(diguyViewPainter\)](#)
- [draw_all_members \(diguyCharacterGroup\)](#)
- [draw_arrow \(diguyViewPainter\)](#)
- [draw_box \(diguyViewPainter\)](#)
- [draw_circle \(diguyViewPainter\)](#)
- [draw_cylinder \(diguyViewPainter\)](#)
- [draw_disc \(diguyViewPainter\)](#)
- [draw_line \(diguyViewPainter\)](#)
- [draw_line_2c \(diguyViewPainter\)](#)
- [draw_pass1 \(diguyCharacter\)](#)
- [draw_pass1 \(diguyScenario\)](#)
- [draw_pass1_all_members \(diguyCharacterGroup\)](#)
- [draw_pass2 \(diguyCharacter\)](#)
- [draw_pass2 \(diguyScenario\)](#)
- [draw_pass2_all_members \(diguyCharacterGroup\)](#)
- [draw_private_path_shape \(diguyViewPainter\)](#)
- [draw_rotated_box \(diguyViewPainter\)](#)
- [draw_sphere \(diguyViewPainter\)](#)
- [draw_text \(diguyViewPainter\)](#)
- [draw_torus \(diguyViewPainter\)](#)
- [draw_visual \(diguySceneObject\)](#)
- [draw_wedge \(diguyViewPainter\)](#)
- [draw_wedge_with_normal \(diguyViewPainter\)](#)
- [dump_package_info \(diguyScenario\)](#)
- [enable_bar_graph \(diguyViewLabel\)](#)
- [end_2d_text_mode \(diguyViewPainter\)](#)
- [end_aim \(diguyCharacter\)](#)
- [end_all_gestures \(diguyCharacter\)](#)
- [end_appearance_effect \(diguyCharacter\)](#)
- [end_character_draw \(diguyGraphicsLink\)](#)
- [end_executing_gesture \(diguyCharacter\)](#)
- [end_gaze \(diguyCharacter\)](#)
- [end_gesture \(diguyCharacterGesture\)](#)
- [end_interaction \(diguyInteractionMachine\)](#)
- [end_legacy_appearance_effect \(diguyCharacter\)](#)
- [end_lpoint \(diguyCharacter\)](#)
- [end_motion_texture \(diguyCharacter\)](#)

- [enter_vpnet_mode \(diguyNetInterface\)](#)
- [equal_to_float \(diguyVariable\)](#)
- [equal_to_int \(diguyVariable\)](#)
- [equal_to_string \(diguyVariable\)](#)
- [erase \(diguyRegion\)](#)
- [erase_rectangle \(diguyRegion\)](#)
- [eval_script \(diguyScenario\)](#)
- [eval_script_file \(diguyScenario\)](#)
- [evaluate_mind_function \(diguyCharacter\)](#)
- [execute_1stage_gesture \(diguyCharacter\)](#)
- [execute_2stage_gesture \(diguyCharacter\)](#)
- [execute_3stage_gesture \(diguyCharacter\)](#)
- [execute_gesture \(diguyCharacter\)](#)
- [execute_table_gesture \(diguyCharacter\)](#)
- [exp_intersect_geometry \(diguyScenario\)](#)
- [face_character \(diguyIGuyController\)](#)
- [facefx_mount_animset \(diguyCharacter\)](#)
- [facefx_mount_animset \(diguyScenario\)](#)
- [find_action_bead \(diguyCharacterPath\)](#)
- [find_aim_bead \(diguyCharacterPath\)](#)
- [find_camera \(diguyScenario\)](#)
- [find_camera_settings \(diguyScenario\)](#)
- [find_chain_settings \(diguyScenario\)](#)
- [find_character \(diguyScenario\)](#)
- [find_character_at_screen_coords \(diguyScenario\)](#)
- [find_crowd \(diguyScenario\)](#)
- [find_crowd_profile \(diguyScenario\)](#)
- [find_decision_bead \(diguyCharacterPath\)](#)
- [find_face_expression \(diguyScenario\)](#)
- [find_facefx_node_index \(diguyCharacter\)](#)
- [find_fog \(diguyScenario\)](#)
- [find_fog_settings \(diguyScenario\)](#)
- [find_formation \(diguyScenario\)](#)
- [find_formation_of_size \(diguyScenario\)](#)
- [find_gaze_bead \(diguyCharacterPath\)](#)
- [find_group \(diguyScenario\)](#)
- [find_guide \(diguyCharacter\)](#)
- [find_info_popup \(diguyScenario\)](#)
- [find_interaction_machine \(diguyScenario\)](#)
- [find_label \(diguyScenario\)](#)
- [find_light \(diguyScenario\)](#)
- [find_light_settings \(diguyScenario\)](#)
- [find_link \(diguyCharacter\)](#)
- [find_navigation_path \(diguyScenario\)](#)
- [find_or_create_crowd \(diguyScenario\)](#)
- [find_or_create_fog \(diguyScenario\)](#)
- [find_or_create_formation \(diguyScenario\)](#)
- [find_or_create_group \(diguyScenario\)](#)
- [find_or_create_guide \(diguyCharacter\)](#)
- [find_or_create_instance \(diguyViewButtonPanel\)](#)
- [find_or_create_label \(diguyScenario\)](#)

- [find_or_create_panel](#) (diguyScenario)
- [find_or_create_region](#) (diguyCrowd)
- [find_or_create_region](#) (diguyScenario)
- [find_or_create_shader_variation](#) (diguyGraphicsShaderTechnique)
- [find_or_create_variable](#) (diguyCharacter)
- [find_or_create_variable](#) (diguyCharacterGroup)
- [find_or_create_variable](#) (diguyScenario)
- [find_panel](#) (diguyScenario)
- [find_part](#) (diguyGraphicsFile)
- [find_path](#) (diguyCharacter)
- [find_path_shape](#) (diguyScenario)
- [find_region](#) (diguyScenario)
- [find_scene_object](#) (diguyScenario)
- [find_script_bead](#) (diguyCharacterPath)
- [find_sensor_region](#) (diguyScenario)
- [find_shader_program](#) (diguyApp)
- [find_shader_technique](#) (diguyApp)
- [find_shape](#) (diguyGraphicsLink)
- [find_shape_by_class_type](#) (diguyGraphicsLink)
- [find_signal](#) (diguyScenario)
- [find_sound](#) (diguyScenario)
- [find_variable](#) (diguyCharacter)
- [find_variable](#) (diguyCharacterGroup)
- [find_variable](#) (diguyScenario)
- [find_view](#) (diguyScenario)
- [find_waypoint](#) (diguyCharacterPath)
- [fire_screen_space_bullet](#) (diguyCharacter)
- [fire_weapon](#) (diguyCharacter)
- [fire_weapon_n_times](#) (diguyCharacter)
- [force_action](#) (diguyCharacter)
- [force_action_and_path_shape](#) (diguyCharacter)
- [force_action_with_duration](#) (diguyCharacter)
- [force_local_path](#) (diguyCharacter)
- [force_octtree_generation](#) (diguyScenario)
- [force_partial_path](#) (diguyCharacter)
- [force_path](#) (diguyCharacter)
- [force_table_action](#) (diguyCharacterGesture)
- [format_lat_or_lon](#) (diguyNetInterface)
- [free_loader_memory](#) (diguyGraphicsMaterial)
- [free_loader_memory](#) (diguyGraphicsMesh)
- [free_loader_memory](#) (diguyGraphicsState)
- [free_loader_memory](#) (diguyGraphicsTexture)
- [free_pose_array](#) (diguyCharacterPoseOverride)
- [free_var_names](#) (diguyCharacterPoseOverride)
- [gaze_at_angle](#) (diguyCharacter)
- [gaze_at_angle_local](#) (diguyCharacter)
- [gaze_at_character](#) (diguyCharacter)
- [gaze_at_point](#) (diguyCharacter)
- [gaze_at_point_local](#) (diguyCharacter)
- [generate_box_vertices](#) (diguyAuthorInterface)
- [generate_cone_vertices](#) (diguyAuthorInterface)

- generate_cylinder_vertices (diguyAuthorInterface)
- generate_navmesh (diguyRegion)
- generate_sphere_vertices (diguyAuthorInterface)
- get_DIS_burst_fuze (diguyImpact)
- get_DIS_burst_quantity (diguyImpact)
- get_DIS_burst_rate (diguyImpact)
- get_DIS_burst_warhead_type (diguyImpact)
- get_DIS_damage_state (diguyCharacter)
- get_DIS_detonation_result (diguyImpact)
- get_DIS_event_number (diguyImpact)
- get_DIS_flames_state (diguyCharacter)
- get_DIS_host (diguyImpact)
- get_DIS_lifeform_state (diguyCharacter)
- get_DIS_munition_number (diguyImpact)
- get_DIS_munition_septet_value (diguyImpact)
- get_DIS_power_plant_state (diguyCharacter)
- get_DIS_primary_weapon_position (diguyCharacter)
- get_DIS_secondary_weapon_position (diguyCharacter)
- get_DIS_site (diguyImpact)
- get_DIS_smoke_state (diguyCharacter)
- get_DIS_smoke_trail_state (diguyCharacter)
- get_DIS_trailing_effect_state (diguyCharacter)
- get_accessory1_state (diguyCharacter)
- get_action (diguyCharacterPathActionBead)
- get_action_bead_at_index (diguyCharacterPath)
- get_action_bead_labels_visible (diguyScenario)
- get_action_channel_weight (diguyCharacter)
- get_action_from_description (diguyCharacter)
- get_action_from_entity_state (diguyCharacter)
- get_action_mode (diguyCharacter)
- get_action_num_channels (diguyCharacter)
- get_active (diguyViewLight)
- get_active_interaction_machine (diguyScenario)
- get_additional_feature_name (diguyGraphicsShaderProgram)
- get_additional_shader_at_index (diguyGraphicsShaderTechnique)
- get_ai_state_string (diguyCharacter)
- get_aim_algorithm (diguyCharacter)
- get_aim_at_link (diguyCharacter)
- get_aim_at_link_offset (diguyCharacter)
- get_aim_azimuth (diguyCharacter)
- get_aim_bead_at_index (diguyCharacterPath)
- get_aim_converge_every_frame (diguyCharacter)
- get_aim_elevation (diguyCharacter)
- get_aim_is_acquired (diguyCharacter)
- get_aim_is_possible (diguyCharacter)
- get_aim_is_steady (diguyCharacter)
- get_aim_param_acquire_factor (diguyCharacter)
- get_aim_param_azimuth_acquired_range (diguyCharacter)
- get_aim_param_elevation_acquired_range (diguyCharacter)
- get_aim_param_max_per_frame_adj_factor (diguyCharacter)
- get_aim_param_new_aim_duration (diguyCharacter)

- [get_aim_param_new_aim_factor](#) (diguyCharacter)
- [get_aim_trajectories_visible](#) (diguyScenario)
- [get_aim_trajectory](#) (diguyCharacter)
- [get_air_b](#) (diguyChainSettings)
- [get_all_blends_disabled](#) (diguyCharacter)
- [get_alternate_filename](#) (diguySceneObject)
- [get_altitude](#) (diguyAuthorInterface)
- [get_altitude_function](#) (diguyCharacter)
- [get_altitude_function](#) (diguyScenario)
- [get_altitude_function_enabled_in_free_position_mode](#) (diguyCharacter)
- [get_altitude_function_enabled_in_path_position_mode](#) (diguyCharacter)
- [get_altitude_interpolation_rate](#) (diguyCharacter)
- [get_altitude_max_climb](#) (diguyScenario)
- [get_altitude_max_drop](#) (diguyScenario)
- [get_altitude_update_rate](#) (diguyCharacter)
- [get_always_run_flag](#) (diguyIGuyController)
- [get_ambient](#) (diguyGraphicsMaterial)
- [get_ambient_color](#) (diguyChainSettings)
- [get_ambient_color](#) (diguyViewLight)
- [get_ambient_color](#) (diguyViewLightSettings)
- [get_angle_to_path](#) (diguyCharacter)
- [get_angular_velocity](#) (diguyCharacter)
- [get_app](#) (diguyApp)
- [get_appearance](#) (diguyCharacter)
- [get_appearance](#) (diguyCrowdProfile)
- [get_appearance_begin_cap](#) (diguyChainSettings)
- [get_appearance_end_cap](#) (diguyChainSettings)
- [get_appearance_name_at_index](#) (diguyCharacter)
- [get_apply_actor_scale_to_action_bead_travel](#) (diguyCharacter)
- [get_apply_actor_scale_to_action_bead_xy_travel](#) (diguyCharacter)
- [get_aspect](#) (diguyViewCamera)
- [get_aspect](#) (diguyViewCameraSettings)
- [get_associated_group](#) (diguyCrowd)
- [get_attached_to_link](#) (diguyGraphicsLink)
- [get_attack_distance](#) (diguyAgentParams)
- [get_attack_fire_interval](#) (diguyAgentParams)
- [get_attack_stands_ground](#) (diguyAgentParams)
- [get_attack_zone](#) (diguyAgentParams)
- [get_attack_zone_enabled](#) (diguyAgentParams)
- [get_attacker](#) (diguyImpact)
- [get_attacker_name](#) (diguyImpact)
- [get_attrib_bind_location](#) (diguyGraphicsShaderProgram)
- [get_author_interface](#) (diguyApp)
- [get_author_selection_handles_visible](#) (diguyScenario)
- [get_author_ui_window_visible](#) (diguyAuthorInterface)
- [get_automatic_blink_duration_in](#) (diguyCharacter)
- [get_automatic_blink_duration_out](#) (diguyCharacter)
- [get_automatic_blink_target_in](#) (diguyCharacter)
- [get_automatic_blink_target_out](#) (diguyCharacter)
- [get_automatic_blinks_enabled](#) (diguyCharacter)
- [get_automatic_cpl_switching_enabled](#) (diguyCharacter)

- [get_automatic_graphics_lod_switching](#) (diguyScenario)
- [get_automatic_motion_lod_switching](#) (diguyScenario)
- [get_autosense_callback_candidate_entered](#) (diguySensorRegion)
- [get_autosense_callback_candidate_left](#) (diguySensorRegion)
- [get_autosense_candidates_group](#) (diguySensorRegion)
- [get_autosense_members_group](#) (diguySensorRegion)
- [get_awareness_radius](#) (diguyCrowd)
- [get_azimuth_eye_left](#) (diguyCharacter)
- [get_azimuth_eye_right](#) (diguyCharacter)
- [get_bar_graph_lines_are_vertical](#) (diguyViewLabel)
- [get_bar_graph_num_lines](#) (diguyViewLabel)
- [get_base_datetime](#) (diguyScenario)
- [get_base_datetime_default_to_now](#) (diguyScenario)
- [get_base_datetime_default_to_today](#) (diguyScenario)
- [get_base_input_mode](#) (diguyApp)
- [get_base_link](#) (diguyCharacter)
- [get_bead_arm_thickness](#) (diguyAuthorVisualObjectHandle)
- [get_bead_array_visible](#) (diguyScenario)
- [get_bead_depth](#) (diguyAuthorVisualObjectHandle)
- [get_begin_at_object_name](#) (diguyCharacterPathAimBead)
- [get_begin_at_object_name](#) (diguyCharacterPathDecisionBead)
- [get_begin_at_object_name](#) (diguyCharacterPathGazeBead)
- [get_begin_at_object_name](#) (diguyCharacterPathScriptBead)
- [get_begin_at_object_offset](#) (diguyCharacterPathAimBead)
- [get_begin_at_object_offset](#) (diguyCharacterPathDecisionBead)
- [get_begin_at_object_offset](#) (diguyCharacterPathGazeBead)
- [get_begin_at_object_offset](#) (diguyCharacterPathScriptBead)
- [get_behavior](#) (diguyAgentParams)
- [get_behavior_path_shape](#) (diguyAgentParams)
- [get_behavior_region](#) (diguyAgentParams)
- [get_behavior_region_border_is_solid](#) (diguyAgentParams)
- [get_behavior_subregion](#) (diguyAgentParams)
- [get_bind_color_array](#) (diguyGraphicsShape)
- [get_bit_depth](#) (diguySound)
- [get_blend_function_name_at_index](#) (diguyApp)
- [get_bounding_radius](#) (diguyCharacter)
- [get_brightness_multiplier](#) (diguyViewLight)
- [get_brush_color](#) (diguyAuthorVisualMaterial)
- [get_brush_is_eraser](#) (diguyAuthorVisualRegionPaintbrush)
- [get_brush_size](#) (diguyAuthorVisualRegionPaintbrush)
- [get_button_state](#) (diguyPluginMouseEvent)
- [get_cache_optimized_motion_files](#) (diguyApp)
- [get_callback_character](#) (diguyCrowd)
- [get_callback_impact](#) (diguyCrowd)
- [get_camera](#) (diguyView)
- [get_camera_3rd_person_look_at_offset](#) (diguyIGuyController)
- [get_camera_3rd_person_look_from_offset](#) (diguyIGuyController)
- [get_camera_attachment_link](#) (diguyIGuyController)
- [get_camera_change_enabled](#) (diguyIGuyController)
- [get_camera_settings_at_index](#) (diguyScenario)
- [get_camera_settings_merge_flag](#) (diguyScenarioMergeSettings)

- `get_camera_use_1st_person_flag` (diguyIGuyController)
- `get_careful_interpolation_threshold` (diguyCharacter)
- `get_casts_shadows` (diguyViewLight)
- `get_chain_mass_position_is_fixed` (diguyChainSimulation)
- `get_chain_settings_at_index` (diguyScenario)
- `get_chain_settings_merge_flag` (diguyScenarioMergeSettings)
- `get_chain_simulation` (diguyCharacter)
- `get_channel_weight` (diguyCharacterGesture)
- `get_character` (diguyCharacterPath)
- `get_character` (diguyCharacterPathEvent)
- `get_character` (diguyGraphicsLink)
- `get_character` (diguyGraphicsShaderInstance)
- `get_character` (diguyGraphicsShape)
- `get_character` (diguyVariable)
- `get_character_at_index` (diguyScenario)
- `get_character_class` (diguyCharacter)
- `get_character_culling_distance` (diguyScenario)
- `get_character_culling_enabled` (diguyScenario)
- `get_character_from_id` (diguyNetInterface)
- `get_character_group` (diguyScenarioMergeSettings)
- `get_character_labels_visible` (diguyScenario)
- `get_character_load_manager` (diguyScenario)
- `get_character_los_function` (diguyCharacter)
- `get_character_merge_flag` (diguyScenarioMergeSettings)
- `get_character_type` (diguyCrowdProfile)
- `get_character_type_abbreviation` (diguyScenario)
- `get_character_type_action_at_index` (diguyScenario)
- `get_character_type_action_direction` (diguyScenario)
- `get_character_type_action_distance` (diguyScenario)
- `get_character_type_action_duration` (diguyScenario)
- `get_character_type_action_facing_angle_in` (diguyScenario)
- `get_character_type_action_facing_angle_out` (diguyScenario)
- `get_character_type_action_num_variants` (diguyScenario)
- `get_character_type_action_posture` (diguyScenario)
- `get_character_type_action_primary_variant` (diguyScenario)
- `get_character_type_action_speed` (diguyScenario)
- `get_character_type_action_variant_at_index` (diguyScenario)
- `get_character_type_appearance_at_index` (diguyScenario)
- `get_character_type_at_index` (diguyScenario)
- `get_character_type_class` (diguyScenario)
- `get_character_type_default_appearance` (diguyScenario)
- `get_character_type_default_skinned_appearance` (diguyScenario)
- `get_character_type_deprecated_at_version` (diguyScenario)
- `get_character_type_deprecated_suggested_alternatives` (diguyScenario)
- `get_character_type_direction_at_index` (diguyScenario)
- `get_character_type_from_appearance` (diguyScenario)
- `get_character_type_gesture_at_index` (diguyScenario)
- `get_character_type_head_appearance_at_index` (diguyScenario)
- `get_character_type_map` (diguyScenario)
- `get_character_type_map_entry_at_index` (diguyScenario)
- `get_character_type_num_actions` (diguyScenario)

- [get_character_type_num_appearances \(diguyScenario\)](#)
- [get_character_type_num_directions \(diguyScenario\)](#)
- [get_character_type_num_gestures \(diguyScenario\)](#)
- [get_character_type_num_head_appearances \(diguyScenario\)](#)
- [get_character_type_num_postures \(diguyScenario\)](#)
- [get_character_type_num_variants \(diguyScenario\)](#)
- [get_character_type_posture_at_index \(diguyScenario\)](#)
- [get_character_type_quality_bias \(diguyScenario\)](#)
- [get_character_type_string \(diguyCharacter\)](#)
- [get_character_type_variant_at_index \(diguyScenario\)](#)
- [get_character_visible \(diguyScenario\)](#)
- [get_character_visualize_bounds_enabled \(diguyScenario\)](#)
- [get_checkpoint_frequency \(diguyScenario\)](#)
- [get_checkpointing_enabled \(diguyScenario\)](#)
- [get_child_at_index \(diguyCharacter\)](#)
- [get_child_link_at_index \(diguyGraphicsLink\)](#)
- [get_clamp_at_end_flag \(diguyPathShape\)](#)
- [get_clamp_s \(diguyGraphicsTexture\)](#)
- [get_clamp_t \(diguyGraphicsTexture\)](#)
- [get_coil_taper_begin_mass_count \(diguyChainSettings\)](#)
- [get_coil_taper_end_mass_count \(diguyChainSettings\)](#)
- [get_coil_thickness \(diguyChainSettings\)](#)
- [get_collision_group_detection_flag \(diguyApp\)](#)
- [get_color \(diguyViewFog\)](#)
- [get_color \(diguyViewFogSettings\)](#)
- [get_companion_waypoint_name \(diguyCharacterPathActionBead\)](#)
- [get_contains_alpha_component \(diguyGraphicsMaterial\)](#)
- [get_contains_alpha_component \(diguyGraphicsState\)](#)
- [get_contains_alpha_component \(diguyGraphicsTexture\)](#)
- [get_contains_skinned_file \(diguyGraphicsShape\)](#)
- [get_control_point_in_position \(diguyWaypoint\)](#)
- [get_control_point_out_position \(diguyWaypoint\)](#)
- [get_coordinate_system_type \(diguyNetInterface\)](#)
- [get_create_network_crowds \(diguyScenario\)](#)
- [get_cross_section \(diguyChainSettings\)](#)
- [get_crowd_at_index \(diguyScenario\)](#)
- [get_crowd_behavior_visible \(diguyScenario\)](#)
- [get_crowd_feelers_visible \(diguyScenario\)](#)
- [get_crowd_influence_visible \(diguyScenario\)](#)
- [get_crowd_merge_flag \(diguyScenarioMergeSettings\)](#)
- [get_crowd_profile_at_index \(diguyScenario\)](#)
- [get_crowd_profile_merge_flag \(diguyScenarioMergeSettings\)](#)
- [get_crowd_regions_visible \(diguyScenario\)](#)
- [get_culling_enabled \(diguyNetInterface\)](#)
- [get_current_action \(diguyCharacter\)](#)
- [get_current_appearance \(diguyCharacter\)](#)
- [get_current_appearance_is \(diguyCharacter\)](#)
- [get_current_behavior \(diguyCrowd\)](#)
- [get_current_bound_shader_program \(diguyApp\)](#)
- [get_current_character \(diguyScenario\)](#)
- [get_current_character_simulator \(diguyCharacter\)](#)

- [get_current_cpl](#) (diguyCharacter)
- [get_current_crowd](#) (diguyScenario)
- [get_current_crowd_profile](#) (diguyScenario)
- [get_current_datetime](#) (diguyScenario)
- [get_current_datetime_advance_factor](#) (diguyScenario)
- [get_current_datetime_advance_method](#) (diguyScenario)
- [get_current_datetime_string](#) (diguyScenario)
- [get_current_facefx_animation_group](#) (diguyCharacter)
- [get_current_formation](#) (diguyCharacterGroup)
- [get_current_graphics_state](#) (diguyGraphicsMesh)
- [get_current_head_appearance](#) (diguyCharacter)
- [get_current_head_appearance_is](#) (diguyCharacter)
- [get_current_motion_lod](#) (diguyCharacter)
- [get_current_path_index](#) (diguyCharacter)
- [get_current_path_shape](#) (diguyScenario)
- [get_current_physics_simulation_mode](#) (diguyCharacter)
- [get_current_region](#) (diguyScenario)
- [get_current_render_mode](#) (diguyScenario)
- [get_current_scenario](#) (diguyApp)
- [get_current_scenario](#) (diguyScenario)
- [get_current_state](#) (diguyInteractionMachine)
- [get_current_tin](#) (diguyCharacter)
- [get_current_tout](#) (diguyCharacter)
- [get_current_traveled_path](#) (diguyCharacter)
- [get_current_traveled_path_is](#) (diguyCharacter)
- [get_current_voice_actor](#) (diguyCharacter)
- [get_data_ptr](#) (diguyGraphicsLink)
- [get_datetimes_are_utc](#) (diguyScenario)
- [get_dds_block_size](#) (diguyGraphicsTexture)
- [get_dead](#) (diguyCharacter)
- [get_decision_bead_at_index](#) (diguyCharacterPath)
- [get_decision_merge_flag](#) (diguyScenarioMergeSettings)
- [get_default_acceleration](#) (diguyVehicleController)
- [get_default_agent_params](#) (diguyCrowdProfile)
- [get_default_blend_duration](#) (diguyCharacterGesture)
- [get_default_bounding_radius](#) (diguyCharacter)
- [get_default_character_shader](#) (diguyScenario)
- [get_default_crowd_size](#) (diguyCrowdProfile)
- [get_default_deceleration](#) (diguyVehicleController)
- [get_default_muzzle_munition_type](#) (diguyCharacter)
- [get_default_particle_system_shader](#) (diguyScenario)
- [get_default_scene_object_shader](#) (diguyScenario)
- [get_default_weapon_muzzle](#) (diguyCharacter)
- [get_default_weapon_sound](#) (diguyCharacter)
- [get_default_weapon_sound_gain](#) (diguyCharacter)
- [get_default_weight](#) (diguyCharacterPoseOverride)
- [get_delta_to_desired_orientation](#) (diguyCharacter)
- [get_delta_to_desired_position](#) (diguyCharacter)
- [get_density](#) (diguyViewFog)
- [get_density](#) (diguyViewFogSettings)
- [get_derive_desired_duration_flag](#) (diguyCharacterPathActionBead)

- `get_derive_duration_flag` (diguyCharacterPathActionBead)
- `get_derive_length_flag` (diguyCharacterPathActionBead)
- `get_derive_motion_reps_flag` (diguyCharacterPathActionBead)
- `get_description` (diguyScenario)
- `get_desired_action` (diguyCharacter)
- `get_desired_duration` (diguyCharacterPathActionBead)
- `get_desired_front_wheels_steer_angle` (diguyVehicleController)
- `get_desired_navpath_distance_threshold` (diguyAgentParams)
- `get_desired_orientation` (diguyCharacter)
- `get_desired_position` (diguyCharacter)
- `get_desired_posture` (diguyCharacter)
- `get_desired_speed` (diguyVehicleController)
- `get_desired_t` (diguyScenario)
- `get_desired_variant` (diguyCharacter)
- `get_diffuse` (diguyGraphicsMaterial)
- `get_diffuse_color` (diguyChainSettings)
- `get_diffuse_color` (diguyViewLight)
- `get_diffuse_color` (diguyViewLightSettings)
- `get_diguy_graphics_file` (diguyGraphicsShape)
- `get_diguy_graphics_file_part` (diguyGraphicsShape)
- `get_direction` (diguyViewLight)
- `get_direction` (diguyViewLightSettings)
- `get_display_remote_chars` (diguyNetInterface)
- `get_distance` (diguyViewCamera)
- `get_distance` (diguyViewCameraSettings)
- `get_distance_along_path` (diguyCharacter)
- `get_distance_into_path` (diguyCharacterPathActionBead)
- `get_distance_into_path` (diguyCharacterPathAimBead)
- `get_distance_into_path` (diguyCharacterPathDecisionBead)
- `get_distance_into_path` (diguyCharacterPathGazeBead)
- `get_distance_into_path` (diguyCharacterPathScriptBead)
- `get_distance_smooth` (diguyWaypoint)
- `get_distance_to_character` (diguyCharacter)
- `get_distance_to_impact` (diguyCharacter)
- `get_distance_to_path` (diguyCharacter)
- `get_distance_xy_to_character` (diguyCharacter)
- `get_dof_state` (diguySceneObject)
- `get_dof_state_flt` (diguySceneObject)
- `get_dolly_link_name` (diguyViewCamera)
- `get_dolly_link_name` (diguyViewCameraSettings)
- `get_dolly_name` (diguyViewCamera)
- `get_dolly_name` (diguyViewCameraSettings)
- `get_dolly_offset_is_in_world_coordinates` (diguyViewCamera)
- `get_dolly_offset_is_in_world_coordinates` (diguyViewCameraSettings)
- `get_dolly_offset_x` (diguyViewCamera)
- `get_dolly_offset_x` (diguyViewCameraSettings)
- `get_dolly_offset_y` (diguyViewCamera)
- `get_dolly_offset_y` (diguyViewCameraSettings)
- `get_dolly_offset_z` (diguyViewCamera)
- `get_dolly_offset_z` (diguyViewCameraSettings)
- `get_draw_character_shadow_disks` (diguyScenario)

- [get_draw_limit \(diguyViewLabel\)](#)
- [get_drawn_by_scenario_flag \(diguyCharacter\)](#)
- [get_duration \(diguyCharacterPath\)](#)
- [get_duration \(diguyCharacterPathActionBead\)](#)
- [get_duration \(diguyCharacterPathAimBead\)](#)
- [get_duration \(diguyCharacterPathDecisionBead\)](#)
- [get_duration \(diguyCharacterPathGazeBead\)](#)
- [get_duration \(diguyCharacterPathScriptBead\)](#)
- [get_duration \(diguySound\)](#)
- [get_dynamic_object_avoidance_method \(diguyCrowd\)](#)
- [get_earth_coord \(diguyNetInterface\)](#)
- [get_edge_faces \(diguyAuthorVisualMaterial\)](#)
- [get_edge_point \(diguyAuthorVisualRegionMesh\)](#)
- [get_edges_data_ptr \(diguyAuthorVisualRegionMesh\)](#)
- [get_elevation_eye_left \(diguyCharacter\)](#)
- [get_elevation_eye_right \(diguyCharacter\)](#)
- [get_emission \(diguyGraphicsMaterial\)](#)
- [get_emit_light \(diguyCharacter\)](#)
- [get_enabled \(diguyCharacter\)](#)
- [get_enabled \(diguyCharacterGuide\)](#)
- [get_enabled \(diguyCrowd\)](#)
- [get_enabled \(diguyLoadManager\)](#)
- [get_enabled \(diguySceneObject\)](#)
- [get_end \(diguyViewFog\)](#)
- [get_end \(diguyViewFogSettings\)](#)
- [get_end_at_object_name \(diguyCharacterPathAimBead\)](#)
- [get_end_at_object_name \(diguyCharacterPathDecisionBead\)](#)
- [get_end_at_object_name \(diguyCharacterPathGazeBead\)](#)
- [get_end_at_object_name \(diguyCharacterPathScriptBead\)](#)
- [get_end_at_object_offset \(diguyCharacterPathAimBead\)](#)
- [get_end_at_object_offset \(diguyCharacterPathDecisionBead\)](#)
- [get_end_at_object_offset \(diguyCharacterPathGazeBead\)](#)
- [get_end_at_object_offset \(diguyCharacterPathScriptBead\)](#)
- [get_eval_decisions_as_scripts \(diguyScenario\)](#)
- [get_exercise \(diguyNetInterface\)](#)
- [get_exercise_interface \(diguyApp\)](#)
- [get_eye_droop \(diguyCharacter\)](#)
- [get_face_exp_merge_flag \(diguyScenarioMergeSettings\)](#)
- [get_face_expression_at_index \(diguyScenario\)](#)
- [get_face_ry \(diguyIGuyController\)](#)
- [get_face_ry_desired \(diguyIGuyController\)](#)
- [get_face_rz \(diguyIGuyController\)](#)
- [get_face_rz_desired \(diguyIGuyController\)](#)
- [get_facefx_actor \(diguyCharacter\)](#)
- [get_facefx_node_name \(diguyCharacter\)](#)
- [get_facefx_node_value \(diguyCharacter\)](#)
- [get_far_plane \(diguyViewCamera\)](#)
- [get_far_plane \(diguyViewCameraSettings\)](#)
- [get_far_plane \(diguyViewLight\)](#)
- [get_farthest_character_to_point \(diguyCharacterGroup\)](#)
- [get_feeler_collision_distance \(diguyAgentParams\)](#)

- `get_feeler_enabled` (diguyAgentParams)
- `get_feeler_height` (diguyAgentParams)
- `get_feeler_rz_offset` (diguyAgentParams)
- `get_feeler_turn_direction` (diguyAgentParams)
- `get_feeler_turn_distance` (diguyAgentParams)
- `get_field_value` (diguyCharacterTypeMap)
- `get_field_value_using_field_string` (diguyCharacterTypeMap)
- `get_filename` (diguyGraphicsFile)
- `get_filename` (diguyGraphicsTexture)
- `get_filename` (diguyScenario)
- `get_filename` (diguySceneObject)
- `get_filename_without_directory` (diguyScenario)
- `get_fill_faces` (diguyAuthorVisualMaterial)
- `get_final_action_bead` (diguyCharacterPath)
- `get_first_link_index` (diguyGraphicsMesh)
- `get_first_link_weight` (diguyGraphicsMesh)
- `get_first_no_progress_trap_time_threshold` (diguyAgentParams)
- `get_first_no_progress_untrap_method` (diguyAgentParams)
- `get_first_normal` (diguyGraphicsMesh)
- `get_first_subgroup_index` (diguyFormation)
- `get_first_tangent` (diguyGraphicsMesh)
- `get_first_texture_indices` (diguyGraphicsMesh)
- `get_first_vertex` (diguyGraphicsMesh)
- `get_fix_segment_offset` (diguyViewCamera)
- `get_fix_x` (diguyViewCamera)
- `get_fix_x` (diguyViewCameraSettings)
- `get_fix_y` (diguyViewCamera)
- `get_fix_y` (diguyViewCameraSettings)
- `get_fix_z` (diguyViewCamera)
- `get_fix_z` (diguyViewCameraSettings)
- `get_focus_affects_gaze` (diguyAgentParams)
- `get_focus_character` (diguyAgentParams)
- `get_focus_group` (diguyAgentParams)
- `get_fog` (diguyView)
- `get_fog_mode` (diguyViewFog)
- `get_fog_mode` (diguyViewFogSettings)
- `get_fog_settings_at_index` (diguyScenario)
- `get_fog_settings_merge_flag` (diguyScenarioMergeSettings)
- `get_follower_instance` (diguyFormation)
- `get_follower_rz_offset` (diguyFormation)
- `get_follower_x_offset` (diguyFormation)
- `get_follower_y_offset` (diguyFormation)
- `get_formation_at_index` (diguyScenario)
- `get_formation_merge_flag` (diguyScenarioMergeSettings)
- `get_forward_normal_in` (diguyWaypoint)
- `get_forward_normal_out` (diguyWaypoint)
- `get_fov` (diguyViewCamera)
- `get_fov` (diguyViewCameraSettings)
- `get_front_wheels_steer_angle` (diguyVehicleController)
- `get_full_window` (diguyViewLabel)
- `get_fully_resolved_filename` (diguySceneObject)

- `get_fvar` (diguyCharacter)
- `get_gaze_at_link` (diguyCharacter)
- `get_gaze_at_link_offset` (diguyCharacter)
- `get_gaze_bead_at_index` (diguyCharacterPath)
- `get_gaze_can_change_posture` (diguyCharacter)
- `get_gaze_can_change_variant` (diguyCharacter)
- `get_gaze_change_enabled` (diguyIGuyController)
- `get_gaze_is_acquired` (diguyCharacter)
- `get_gaze_is_active` (diguyCharacter)
- `get_gaze_is_steady` (diguyCharacter)
- `get_gaze_link_azimuth_limits` (diguyCharacter)
- `get_gaze_link_azimuth_tolerance` (diguyCharacter)
- `get_gaze_link_elevation_limits` (diguyCharacter)
- `get_gaze_link_elevation_tolerance` (diguyCharacter)
- `get_gaze_link_max_angular_vel` (diguyCharacter)
- `get_gaze_link_scale_factor` (diguyCharacter)
- `get_gaze_link_uses_desired_locate_time` (diguyCharacter)
- `get_gaze_param_desired_locate_time` (diguyCharacter)
- `get_gaze_point` (diguyCharacter)
- `get_geometry_tag` (diguyCharacter)
- `get_gesture_type` (diguyCharacterGesture)
- `get_gl_format` (diguyGraphicsTexture)
- `get_gl_format_as_string` (diguyGraphicsTexture)
- `get_graphics_api_node_ptr` (diguyCharacter)
- `get_graphics_api_node_ptr` (diguyScenario)
- `get_graphics_api_shape_update_disabled` (diguyCharacter)
- `get_graphics_lod` (diguyCharacter)
- `get_graphics_lod` (diguyGraphicsShape)
- `get_graphics_ptr` (diguyCharacter)
- `get_graphics_state_switch_index` (diguyGraphicsShape)
- `get_gravity` (diguyChainSettings)
- `get_ground_b` (diguyChainSettings)
- `get_ground_clamp_max_climb` (diguyNetInterface)
- `get_ground_clamp_max_drop` (diguyNetInterface)
- `get_ground_clamp_reflected_land_entities_enabled` (diguyNetInterface)
- `get_ground_clamp_reflected_land_entities_from_host` (diguyNetInterface)
- `get_ground_clamp_when_on_path` (diguyCharacterPath)
- `get_ground_clamp_when_on_path` (diguyPathShape)
- `get_ground_height` (diguyChainSettings)
- `get_ground_k` (diguyChainSettings)
- `get_ground_lateral_b` (diguyChainSettings)
- `get_ground_stiction_engage_vel` (diguyChainSettings)
- `get_ground_stiction_release_force` (diguyChainSettings)
- `get_group_at_index` (diguyScenario)
- `get_group_centroid` (diguyCharacterGroup)
- `get_group_membership_at_index` (diguyCharacter)
- `get_group_merge_flag` (diguyScenarioMergeSettings)
- `get_group_min_circle` (diguyCharacterGroup)
- `get_guide_algorithm_float_parameter` (diguyCharacterGuide)
- `get_guide_algorithm_string_parameter` (diguyCharacterGuide)
- `get_guide_merge_flag` (diguyScenarioMergeSettings)

- `get_guide_name` (diguyFormation)
- `get_handle_shape` (diguyAuthorVisualObjectHandle)
- `get_has_begun_flag` (diguyCharacterGesture)
- `get_has_bump_maps` (diguyCharacter)
- `get_has_completed_flag` (diguyCharacterGesture)
- `get_has_offset_rotation` (diguyGraphicsShape)
- `get_has_offset_scale` (diguyGraphicsShape)
- `get_has_offset_translation` (diguyGraphicsShape)
- `get_heading` (diguyInteractionMachine)
- `get_heading_visible` (diguyInteractionMachine)
- `get_height` (diguyAuthorVisualObjectHandle)
- `get_height` (diguyGraphicsTexture)
- `get_height` (diguyView)
- `get_height` (diguyViewButtonPanel)
- `get_height` (diguyViewLabel)
- `get_hide_author_selection_handles_during_play` (diguyScenario)
- `get_highlight_level` (diguyAuthorVisualRegionMesh)
- `get_history_size_increment_t` (diguyCharacter)
- `get_history_size_t` (diguyCharacter)
- `get_history_type` (diguyCharacter)
- `get_history_type` (diguyScenario)
- `get_hit_link_index` (diguyImpact)
- `get_hit_link_name` (diguyImpact)
- `get_hit_scene_object` (diguyImpact)
- `get_hit_shape_name` (diguyImpact)
- `get_host` (diguyNetInterface)
- `get_how_far` (diguyCharacterPathEvent)
- `get_ideal_num_threads` (diguyApp)
- `get_ignore_mouse drags` (diguyViewCamera)
- `get_ignore_mouse drags` (diguyViewCameraSettings)
- `get_iguy_character` (diguyIGuyController)
- `get_iguy_controller` (diguyScenario)
- `get_impact_at_screen_coords` (diguyScenario)
- `get_impact_been_processed` (diguyImpact)
- `get_impact_normal` (diguyImpact)
- `get_impact_number` (diguyImpact)
- `get_impact_on_line` (diguyScenario)
- `get_impact_radius` (diguyImpact)
- `get_impact_speed` (diguyImpact)
- `get_impact_type` (diguyImpact)
- `get_impact_type_string` (diguyImpact)
- `get_impact_velocity` (diguyImpact)
- `get_include_in_octtree` (diguySceneObject)
- `get_include_in_physics_sim_scene` (diguySceneObject)
- `get_index` (diguyCharacter)
- `get_index` (diguyGraphicsLink)
- `get_index_buffer_data` (diguyGraphicsMesh)
- `get_index_buffer_size` (diguyGraphicsMesh)
- `get_index_count` (diguyGraphicsMesh)
- `get_index_in_subgroup` (diguyCharacterGroup)
- `get_index_of_chain_mass_connected_to_character` (diguyChainSettings)

- [get_index_of_member](#) (diguyCharacterGroup)
- [get_index_of_member](#) (diguyCrowd)
- [get_index_type](#) (diguyGraphicsMesh)
- [get_influence_box_enabled](#) (diguyAgentParams)
- [get_info](#) (diguyInteractionMachine)
- [get_info_popup_at_index](#) (diguyScenario)
- [get_info_popup_default_encoding](#) (diguyScenario)
- [get_info_popup_merge_flag](#) (diguyScenarioMergeSettings)
- [get_info_text](#) (diguySignal)
- [get_info_visible](#) (diguyInteractionMachine)
- [get_initial_character_simulator](#) (diguyCharacter)
- [get_initial_heading](#) (diguyInteractionMachine)
- [get_initial_info](#) (diguyInteractionMachine)
- [get_initial_motion_texture](#) (diguyCharacter)
- [get_initial_motion_texture_gain](#) (diguyCharacter)
- [get_initial_orientation](#) (diguyCharacter)
- [get_initial_path](#) (diguyCharacter)
- [get_initial_physics_simulation_mode](#) (diguyCharacter)
- [get_initial_position](#) (diguyCharacter)
- [get_initial_render_mode](#) (diguyScenario)
- [get_initial_ui_appearance](#) (diguyInteractionMachine)
- [get_initial_up_vector](#) (diguyCharacter)
- [get_initial_value_as_float](#) (diguyVariable)
- [get_initial_value_as_int](#) (diguyVariable)
- [get_initial_value_as_string](#) (diguyVariable)
- [get_input_capture_enabled](#) (diguyIGuyController)
- [get_input_id](#) (diguyInteractionMachine)
- [get_input_int_param_0](#) (diguyInteractionMachine)
- [get_input_resulting_state](#) (diguyInteractionMachine)
- [get_input_shortcuts_visible](#) (diguyInteractionMachine)
- [get_input_string](#) (diguyInteractionMachine)
- [get_inputs_visible](#) (diguyInteractionMachine)
- [get_instance](#) (diguyViewButtonPanel)
- [get_interaction_is_active](#) (diguyInteractionMachine)
- [get_interaction_machine_at_index](#) (diguyScenario)
- [get_interaction_subject](#) (diguyIGuyController)
- [get_interaction_subject](#) (diguyInteractionMachine)
- [get_interpolation_flag](#) (diguyCharacterGesture)
- [get_intersection_at_screen_coords](#) (diguyScenario)
- [get_intersection_shader](#) (diguyGraphicsShaderTechnique)
- [get_inverse_rotation_matrix](#) (diguyGraphicsLink)
- [get_inverse_rotation_matrix_4x4_ptr](#) (diguyGraphicsLink)
- [get_inverted](#) (diguyAuthorVisualObjectHandle)
- [get_invisible_flag](#) (diguyCharacter)
- [get_ipaddr](#) (diguyNetInterface)
- [get_is_3d_sound_flag](#) (diguySoundInstance)
- [get_is_active](#) (diguyCharacter)
- [get_is_aiming](#) (diguyCharacter)
- [get_is_base_link](#) (diguyGraphicsLink)
- [get_is_clickable](#) (diguyViewLabel)
- [get_is_deformable](#) (diguyGraphicsMesh)

- [get_is_drawn \(diguyCharacter\)](#)
- [get_is_from_net_detonation \(diguyImpact\)](#)
- [get_is_frozen \(diguyChainSimulation\)](#)
- [get_is_hidden \(diguyGraphicsShape\)](#)
- [get_is_immune_to_impacts \(diguyCharacter\)](#)
- [get_is_load_managed \(diguyCharacter\)](#)
- [get_is_network_paused \(diguyCharacter\)](#)
- [get_is_network_published \(diguyCharacter\)](#)
- [get_is_network_reflected \(diguyCharacter\)](#)
- [get_is_noise \(diguyViewLabel\)](#)
- [get_is_path_planning \(diguyCharacter\)](#)
- [get_is_playing \(diguySoundInstance\)](#)
- [get_is_position_link \(diguyGraphicsLink\)](#)
- [get_is_recycled \(diguyCharacter\)](#)
- [get_is_scene_object \(diguyCharacter\)](#)
- [get_is_skinned_mesh \(diguyGraphicsMesh\)](#)
- [get_is_sky_box \(diguySceneObject\)](#)
- [get_is_stopping \(diguySoundInstance\)](#)
- [get_is_temporary \(diguyCharacter\)](#)
- [get_is_temporary \(diguyViewLabel\)](#)
- [get_is_unlit \(diguyGraphicsState\)](#)
- [get_is_visible_in_signal_palette \(diguySignal\)](#)
- [get_ivar \(diguyCharacter\)](#)
- [get_joint_type \(diguyGraphicsLink\)](#)
- [get_joystick_input_enabled \(diguyIGuyController\)](#)
- [get_key \(diguyPluginKeyboardEvent\)](#)
- [get_keyboard_input_enabled \(diguyIGuyController\)](#)
- [get_label \(diguyCharacterGroup\)](#)
- [get_label_at_index \(diguyScenario\)](#)
- [get_label_background_color \(diguyCharacter\)](#)
- [get_label_background_color_vec4 \(diguyCharacter\)](#)
- [get_label_is_visible \(diguyCharacter\)](#)
- [get_label_link_offset_override \(diguyCharacter\)](#)
- [get_label_offset_above_centroid \(diguyCharacterGroup\)](#)
- [get_label_shows_character_state \(diguyCharacter\)](#)
- [get_label_text \(diguyCharacter\)](#)
- [get_label_text_color \(diguyCharacter\)](#)
- [get_label_text_color_vec4 \(diguyCharacter\)](#)
- [get_label_text_shadow_color \(diguyCharacter\)](#)
- [get_label_text_shadow_color_vec4 \(diguyCharacter\)](#)
- [get_label_text_shadow_is_visible \(diguyCharacter\)](#)
- [get_last_bound_material \(diguyGraphicsMaterial\)](#)
- [get_last_bound_texture \(diguyGraphicsTexture\)](#)
- [get_last_clicked_character \(diguyApp\)](#)
- [get_last_clicked_xyz \(diguyApp\)](#)
- [get_last_env_impact_info \(diguyScenario\)](#)
- [get_last_impact_been_processed \(diguyCharacter\)](#)
- [get_last_impact_is_from_vehicle \(diguyCharacter\)](#)
- [get_last_impact_link \(diguyCharacter\)](#)
- [get_last_impact_link_is \(diguyCharacter\)](#)
- [get_last_impact_number \(diguyCharacter\)](#)

- `get_last_impact_record` (diguyCharacter)
- `get_last_impact_speed_is_greater_than` (diguyCharacter)
- `get_last_input_variable` (diguyInteractionMachine)
- `get_last_selected_input` (diguyInteractionMachine)
- `get_lat_lon` (diguyNetInterface)
- `get_left_normal_in` (diguyWaypoint)
- `get_left_normal_out` (diguyWaypoint)
- `get_length` (diguyChainSimulation)
- `get_length` (diguyCharacterPath)
- `get_length` (diguyCharacterPathActionBead)
- `get_length` (diguyCharacterPathAimBead)
- `get_length` (diguyCharacterPathDecisionBead)
- `get_length` (diguyCharacterPathGazeBead)
- `get_length` (diguyCharacterPathScriptBead)
- `get_length` (diguyPathShape)
- `get_library_function_merge_flag` (diguyScenarioMergeSettings)
- `get_light` (diguyView)
- `get_light_array_visible` (diguyScenario)
- `get_light_settings_at_index` (diguyScenario)
- `get_light_settings_merge_flag` (diguyScenarioMergeSettings)
- `get_line_width` (diguyAuthorVisualMaterial)
- `get_link` (diguyGraphicsShape)
- `get_link_at_index` (diguyCharacter)
- `get_link_enabled_for_gaze` (diguyCharacter)
- `get_link_enabled_for_lpoint` (diguyCharacter)
- `get_link_index_offset` (diguyGraphicsMesh)
- `get_link_index_stride` (diguyGraphicsMesh)
- `get_link_name_at_index` (diguyCharacter)
- `get_link_position` (diguyCharacter)
- `get_link_position_double` (diguyCharacter)
- `get_link_position_with_offset` (diguyCharacter)
- `get_link_position_with_offset_double` (diguyCharacter)
- `get_link_relative_position` (diguyCharacter)
- `get_link_relative_position_with_offset` (diguyCharacter)
- `get_link_shape_name_at_index` (diguyCharacter)
- `get_link_transformation_matrix` (diguyGraphicsLink)
- `get_link_transformation_matrix_4x4_ptr` (diguyGraphicsLink)
- `get_link_weight_offset` (diguyGraphicsMesh)
- `get_link_weight_stride` (diguyGraphicsMesh)
- `get_linked` (diguyGraphicsShaderProgram)
- `get_loaded` (diguyGraphicsShaderProgram)
- `get_loaded_from_review_data` (diguyScenario)
- `get_local_space_bounding_box` (diguyCharacter)
- `get_local_transformation_matrix` (diguyGraphicsLink)
- `get_local_transformation_matrix_4x4_ptr` (diguyGraphicsLink)
- `get_location` (diguyImpact)
- `get_location_in_link_coordinates` (diguyImpact)
- `get_lod_ranges` (diguyGraphicsShape)
- `get_log_channel_at_index` (diguyApp)
- `get_lpoint_base_gesture_name` (diguyCharacter)
- `get_lpoint_is_acquired` (diguyCharacter)

- [get_lpoint_is_active](#) (diguyCharacter)
- [get_lpoint_is_steady](#) (diguyCharacter)
- [get_lpoint_point](#) (diguyCharacter)
- [get_lua_object_name](#) (diguyCharacter)
- [get_lua_object_waiting_for_input](#) (diguyApp)
- [get_lua_objects_visible](#) (diguyScenario)
- [get_lua_state](#) (diguyScenario)
- [get_mapped_appearance](#) (diguyCharacterTypeMap)
- [get_mapped_character_type](#) (diguyCharacterTypeMap)
- [get_mass](#) (diguyChainSettings)
- [get_material](#) (diguyGraphicsState)
- [get_material_parameter_4f](#) (diguyGraphicsShape)
- [get_max](#) (diguySceneObject)
- [get_max_LOS_travel_distance](#) (diguyAgentParams)
- [get_max_character_shader_quality_level](#) (diguyApp)
- [get_max_distance_planning_multiplier](#) (diguyScenario)
- [get_max_focus_distance](#) (diguyAgentParams)
- [get_max_fov](#) (diguyViewCamera)
- [get_max_fov](#) (diguyViewCameraSettings)
- [get_max_orientation_change_rate](#) (diguyAgentParams)
- [get_max_reps](#) (diguyCharacterGesture)
- [get_max_scene_object_shader_quality_level](#) (diguyApp)
- [get_max_shader_quality_level](#) (diguyCharacter)
- [get_max_ticks_behind_until_drop](#) (diguyScenario)
- [get_max_track_distance](#) (diguyViewCamera)
- [get_max_unprocessed_impacts](#) (diguyCharacter)
- [get_maximum_possible_current_cpl](#) (diguyCharacter)
- [get_member_at_index](#) (diguyCharacterGroup)
- [get_member_at_index](#) (diguyCrowd)
- [get_mesh_at_index_for_lod](#) (diguyGraphicsShape)
- [get_mesh_origin](#) (diguyAuthorVisualRegionMesh)
- [get_mesh_region_array_visible](#) (diguyScenario)
- [get_min](#) (diguySceneObject)
- [get_min_fov](#) (diguyViewCamera)
- [get_min_fov](#) (diguyViewCameraSettings)
- [get_min_track_distance](#) (diguyViewCamera)
- [get_mind_always_updates](#) (diguyCharacter)
- [get_mind_base_class](#) (diguyCrowdProfile)
- [get_mind_current_state_name](#) (diguyCharacter)
- [get_mind_field_as_float](#) (diguyCharacter)
- [get_mind_field_as_int](#) (diguyCharacter)
- [get_mind_field_as_string](#) (diguyCharacter)
- [get_mind_init_argument](#) (diguyCrowdProfile)
- [get_mind_name](#) (diguyCharacter)
- [get_mind_name_at_index](#) (diguyScenario)
- [get_minimum_cpl](#) (diguyCharacter)
- [get_minimum_full_update_period](#) (diguyCharacter)
- [get_minimum_full_update_period](#) (diguyLoadManager)
- [get_minimum_height](#) (diguyViewLabel)
- [get_minimum_width](#) (diguyViewLabel)
- [get_mipmap_data_offset](#) (diguyGraphicsTexture)

- [get_mipmap_data_size](#) (diguyGraphicsTexture)
- [get_most_recent_alternate_value_matches](#) (diguyCharacterTypeMap)
- [get_most_recent_exact_value_matches](#) (diguyCharacterTypeMap)
- [get_most_recent_forced_action](#) (diguyCharacter)
- [get_most_recent_forced_action_t](#) (diguyCharacter)
- [get_most_recent_forced_action_t_offset](#) (diguyCharacter)
- [get_most_recent_match_level](#) (diguyCharacterTypeMap)
- [get_most_recent_match_passed](#) (diguyCharacterTypeMap)
- [get_most_recent_wildcard_value_matches](#) (diguyCharacterTypeMap)
- [get_motion_interpolation_flag](#) (diguyCharacter)
- [get_motion_lod](#) (diguyCharacter)
- [get_motion_reps](#) (diguyCharacterPathActionBead)
- [get_mouse_input_enabled](#) (diguyIGuyController)
- [get_mouse_look_changes_orientation](#) (diguyIGuyController)
- [get_mouse_screen_xy](#) (diguyApp)
- [get_move_mode](#) (diguyViewCamera)
- [get_move_mode](#) (diguyViewCameraSettings)
- [get_move_on_t_max](#) (diguyAgentParams)
- [get_move_on_t_min](#) (diguyAgentParams)
- [get_multithreaded_path_planning_enabled](#) (diguyScenario)
- [get_munition_type](#) (diguyImpact)
- [get_munition_type_from_DIS_septet](#) (diguyApp)
- [get_muzzle_link](#) (diguyCharacter)
- [get_muzzle_offset](#) (diguyCharacter)
- [get_name](#) (diguyAuthorVisualMaterial)
- [get_name](#) (diguyChainSettings)
- [get_name](#) (diguyCharacter)
- [get_name](#) (diguyCharacterFaceExpression)
- [get_name](#) (diguyCharacterGesture)
- [get_name](#) (diguyCharacterGroup)
- [get_name](#) (diguyCharacterGuide)
- [get_name](#) (diguyCharacterPath)
- [get_name](#) (diguyCharacterPathActionBead)
- [get_name](#) (diguyCharacterPathAimBead)
- [get_name](#) (diguyCharacterPathDecisionBead)
- [get_name](#) (diguyCharacterPathEvent)
- [get_name](#) (diguyCharacterPathGazeBead)
- [get_name](#) (diguyCharacterPathScriptBead)
- [get_name](#) (diguyCrowd)
- [get_name](#) (diguyCrowdProfile)
- [get_name](#) (diguyFormation)
- [get_name](#) (diguyGraphicsLink)
- [get_name](#) (diguyGraphicsMaterial)
- [get_name](#) (diguyGraphicsMesh)
- [get_name](#) (diguyGraphicsShaderProgram)
- [get_name](#) (diguyGraphicsShaderTechnique)
- [get_name](#) (diguyGraphicsShape)
- [get_name](#) (diguyGraphicsTexture)
- [get_name](#) (diguyInfoPopup)
- [get_name](#) (diguyInteractionMachine)
- [get_name](#) (diguyPathShape)

- `get_name` (diguyRegion)
- `get_name` (diguyScenario)
- `get_name` (diguySceneObject)
- `get_name` (diguySensorRegion)
- `get_name` (diguySignal)
- `get_name` (diguySound)
- `get_name` (diguySoundInstance)
- `get_name` (diguyVariable)
- `get_name` (diguyView)
- `get_name` (diguyViewButtonPanel)
- `get_name` (diguyViewCamera)
- `get_name` (diguyViewCameraSettings)
- `get_name` (diguyViewFog)
- `get_name` (diguyViewFogSettings)
- `get_name` (diguyViewLabel)
- `get_name` (diguyViewLight)
- `get_name` (diguyViewLightSettings)
- `get_name` (diguyWaypoint)
- `get_name_from_entity` (diguyNetInterface)
- `get_name_label` (diguyCharacter)
- `get_name_of_plugin` (diguyApp)
- `get_near_plane` (diguyViewCamera)
- `get_near_plane` (diguyViewCameraSettings)
- `get_near_plane` (diguyViewLight)
- `get_nearest_active_character` (diguyCharacter)
- `get_nearest_active_character_in_group` (diguyCharacter)
- `get_nearest_character_to_point` (diguyCharacterGroup)
- `get_nearest_character_type_map` (diguyScenario)
- `get_nearest_character_type_map_id_match` (diguyScenario)
- `get_nearest_character_type_map_using_field_strings` (diguyScenario)
- `get_nearest_point_in_region` (diguyRegion)
- `get_nearest_reachable_point` (diguyRegion)
- `get_nearest_waypoint_index_in_path` (diguyCharacter)
- `get_negative_wraps_x` (diguyViewLabel)
- `get_negative_wraps_y` (diguyViewLabel)
- `get_neighbor2_spring_b` (diguyChainSettings)
- `get_neighbor2_spring_k` (diguyChainSettings)
- `get_neighbor2_spring_length` (diguyChainSettings)
- `get_net_id` (diguyExerciseInterface)
- `get_network_datetime` (diguyScenario)
- `get_network_datetime_broadcast_interval` (diguyNetInterface)
- `get_network_datetime_enable_diguy_custom_pdus` (diguyNetInterface)
- `get_network_datetime_enable_dis_start_resume_pdus` (diguyNetInterface)
- `get_network_datetime_mode` (diguyNetInterface)
- `get_network_entity_number` (diguyCharacter)
- `get_network_marking` (diguyCharacter)
- `get_network_translation` (diguyScenario)
- `get_next_participant_name` (diguyNetInterface)
- `get_nod_azimuth` (diguyCharacter)
- `get_nod_elevation` (diguyCharacter)
- `get_nodename` (diguySceneObject)

- [get_normal_offset \(diguyGraphicsMesh\)](#)
- [get_normal_stride \(diguyGraphicsMesh\)](#)
- [get_normal_x \(diguyImpact\)](#)
- [get_normal_y \(diguyImpact\)](#)
- [get_normal_z \(diguyImpact\)](#)
- [get_num_action_beads \(diguyCharacterPath\)](#)
- [get_num_active_fire_effects \(diguyScenario\)](#)
- [get_num_active_members \(diguyCharacterGroup\)](#)
- [get_num_active_point_lights \(diguyScenario\)](#)
- [get_num_additional_feature_names \(diguyGraphicsShaderProgram\)](#)
- [get_num_additional_shaders \(diguyGraphicsShaderTechnique\)](#)
- [get_num_aim_beads \(diguyCharacterPath\)](#)
- [get_num_appearances_of_type \(diguyCharacter\)](#)
- [get_num_blend_functions \(diguyApp\)](#)
- [get_num_camera_settings \(diguyScenario\)](#)
- [get_num_chain_settings \(diguyScenario\)](#)
- [get_num_channels \(diguyCharacterGesture\)](#)
- [get_num_character_type_map_entries \(diguyScenario\)](#)
- [get_num_character_types \(diguyScenario\)](#)
- [get_num_character_types_with_appearance \(diguyScenario\)](#)
- [get_num_characters \(diguyScenario\)](#)
- [get_num_child_links \(diguyGraphicsLink\)](#)
- [get_num_children \(diguyCharacter\)](#)
- [get_num_coils \(diguyChainSettings\)](#)
- [get_num_columns \(diguyRegion\)](#)
- [get_num_components \(diguyGraphicsTexture\)](#)
- [get_num_coordinate_systems \(diguyNetInterface\)](#)
- [get_num_crowd_profiles \(diguyScenario\)](#)
- [get_num_crowds \(diguyScenario\)](#)
- [get_num_decision_beads \(diguyCharacterPath\)](#)
- [get_num_edges \(diguyAuthorVisualRegionMesh\)](#)
- [get_num_entries \(diguyCrowdProfile\)](#)
- [get_num_face_expressions \(diguyScenario\)](#)
- [get_num_facefx_nodes \(diguyCharacter\)](#)
- [get_num_fog_settings \(diguyScenario\)](#)
- [get_num_followers \(diguyFormation\)](#)
- [get_num_formation \(diguyScenario\)](#)
- [get_num_gaze_beads \(diguyCharacterPath\)](#)
- [get_num_group_memberships \(diguyCharacter\)](#)
- [get_num_groups \(diguyScenario\)](#)
- [get_num_hits \(diguyImpact\)](#)
- [get_num_included_mipmaps \(diguyGraphicsTexture\)](#)
- [get_num_info_popups \(diguyScenario\)](#)
- [get_num_inputs \(diguyInteractionMachine\)](#)
- [get_num_interaction_machines \(diguyScenario\)](#)
- [get_num_labels \(diguyScenario\)](#)
- [get_num_layers \(diguyAuthorVisualMaterial\)](#)
- [get_num_light_settings \(diguyScenario\)](#)
- [get_num_link_matrices \(diguyGraphicsShaderInstance\)](#)
- [get_num_link_shapes \(diguyCharacter\)](#)
- [get_num_links \(diguyCharacter\)](#)

- [get_num_lods](#) (diguyGraphicsShape)
- [get_num_log_channels](#) (diguyApp)
- [get_num_masses](#) (diguyChainSettings)
- [get_num_members](#) (diguyCharacterGroup)
- [get_num_members](#) (diguyCrowd)
- [get_num_meshes](#) (diguyGraphicsShape)
- [get_num_minds](#) (diguyScenario)
- [get_num_panels](#) (diguyScenario)
- [get_num_path_shapes](#) (diguyScenario)
- [get_num_paths](#) (diguyCharacter)
- [get_num_plugins](#) (diguyApp)
- [get_num_points](#) (diguyAuthorVisualMultiline)
- [get_num_pose_overrides](#) (diguyCharacter)
- [get_num_progression_stages](#) (diguyCharacterGesture)
- [get_num_quads](#) (diguyAuthorVisualRegionMesh)
- [get_num_regions](#) (diguyScenario)
- [get_num_rendering_passes](#) (diguyApp)
- [get_num_rows](#) (diguyRegion)
- [get_num_scenarios](#) (diguyApp)
- [get_num_scene_objects](#) (diguyScenario)
- [get_num_script_beads](#) (diguyCharacterPath)
- [get_num_secondary_views](#) (diguyScenario)
- [get_num_segments](#) (diguyWaypoint)
- [get_num_sensor_regions](#) (diguyScenario)
- [get_num_shader_lod_programs](#) (diguyGraphicsShaderTechnique)
- [get_num_shader_programs](#) (diguyApp)
- [get_num_shader_techniques](#) (diguyApp)
- [get_num_shapes](#) (diguyGraphicsLink)
- [get_num_signals](#) (diguyScenario)
- [get_num_slices](#) (diguyAuthorVisualMaterial)
- [get_num_sounds](#) (diguyScenario)
- [get_num_stages](#) (diguyCharacterGesture)
- [get_num_states](#) (diguyInteractionMachine)
- [get_num_subgroups](#) (diguyFormation)
- [get_num_switched_graphics_states](#) (diguyGraphicsMesh)
- [get_num_switched_graphics_states](#) (diguyGraphicsShape)
- [get_num_table_actions](#) (diguyCharacterGesture)
- [get_num_texture_repetitions](#) (diguyChainSettings)
- [get_num_textures](#) (diguyGraphicsState)
- [get_num_threads](#) (diguyApp)
- [get_num_type_map_categories](#) (diguyScenario)
- [get_num_type_map_character_classes](#) (diguyScenario)
- [get_num_type_map_subcategories](#) (diguyScenario)
- [get_num_variables](#) (diguyCharacter)
- [get_num_variables](#) (diguyCharacterGroup)
- [get_num_variables](#) (diguyGraphicsShaderInstance)
- [get_num_variables](#) (diguyGraphicsShaderProgram)
- [get_num_variables](#) (diguyScenario)
- [get_num_vars](#) (diguyCharacterPoseOverride)
- [get_num_waypoints](#) (diguyCharacterPath)
- [get_num_waypoints](#) (diguyPathShape)

- [get_num_wildcards](#) (diguyCharacterTypeMap)
- [get_number_of_channels](#) (diguySound)
- [get_offset_rotation](#) (diguyGraphicsLink)
- [get_offset_rotation](#) (diguyGraphicsShape)
- [get_offset_scale](#) (diguyGraphicsShape)
- [get_offset_translation](#) (diguyGraphicsLink)
- [get_offset_translation](#) (diguyGraphicsShape)
- [get_one_way_portal](#) (diguyPathShape)
- [get_online](#) (diguyNetInterface)
- [get_orientation](#) (diguyAuthorVisualObjectHandle)
- [get_orientation](#) (diguyCharacter)
- [get_orientation](#) (diguySceneObject)
- [get_orientation_acquired](#) (diguyCharacterGuide)
- [get_orientation_change_enabled](#) (diguyIGuyController)
- [get_orientation_eyes](#) (diguyCharacter)
- [get_orientation_in](#) (diguyWaypoint)
- [get_orientation_offset](#) (diguyScenarioMergeSettings)
- [get_orientation_out](#) (diguyWaypoint)
- [get_orientation_pitch](#) (diguyViewCamera)
- [get_orientation_pitch](#) (diguyViewCameraSettings)
- [get_orientation_relative_to_parent](#) (diguyCharacter)
- [get_orientation_roll](#) (diguyViewCamera)
- [get_orientation_roll](#) (diguyViewCameraSettings)
- [get_orientation_yaw](#) (diguyViewCamera)
- [get_orientation_yaw](#) (diguyViewCameraSettings)
- [get_overall_duration](#) (diguyCharacterGesture)
- [get_overall_history_type](#) (diguyScenario)
- [get_painting_subregion](#) (diguyAuthorVisualRegionPaintbrush)
- [get_paintrush_mode](#) (diguyAuthorVisualRegionPaintbrush)
- [get_panel_at_index](#) (diguyScenario)
- [get_parent](#) (diguyCharacter)
- [get_parent_link](#) (diguyGraphicsLink)
- [get_parent_link_name](#) (diguyCharacter)
- [get_particle_system_merge_flag](#) (diguyScenarioMergeSettings)
- [get_path](#) (diguyCharacterPathEvent)
- [get_path_at_index](#) (diguyCharacter)
- [get_path_follow_pause_t_when_off_path](#) (diguyAgentParams)
- [get_path_follow_position_lookahead_distance](#) (diguyAgentParams)
- [get_path_follow_unpause_t_at_radius](#) (diguyAgentParams)
- [get_path_name_at_index](#) (diguyCharacter)
- [get_path_planner_cost_rule](#) (diguyAgentParams)
- [get_path_planner_fallback_region](#) (diguyAgentParams)
- [get_path_planner_neutral_cost_bias](#) (diguyAgentParams)
- [get_path_planner_preferred_cost_bias](#) (diguyAgentParams)
- [get_path_planner_preferred_region_mask](#) (diguyAgentParams)
- [get_path_planner_repulsed_cost_bias](#) (diguyAgentParams)
- [get_path_planner_repulsed_region_mask](#) (diguyAgentParams)
- [get_path_planning_result](#) (diguyCharacter)
- [get_path_shape](#) (diguyCharacterPath)
- [get_path_shape_at_index](#) (diguyScenario)
- [get_path_shape_merge_flag](#) (diguyScenarioMergeSettings)

- [get_pause_scenario_while_active](#) (diguyInteractionMachine)
- [get_pen2_color](#) (diguyAuthorVisualMaterial)
- [get_pen_color](#) (diguyAuthorVisualMaterial)
- [get_phoneme_filename](#) (diguySound)
- [get_physics_collision_group](#) (diguyCharacter)
- [get_physics_collision_group](#) (diguySceneObject)
- [get_physics_sim_ground_plane_enabled](#) (diguyApp)
- [get_physics_sim_ground_plane_height](#) (diguyApp)
- [get_pitch_coupled](#) (diguyWaypoint)
- [get_pitch_coupling_offset](#) (diguyWaypoint)
- [get_pitch_has_limits](#) (diguyViewCamera)
- [get_pitch_has_limits](#) (diguyViewCameraSettings)
- [get_pitch_in](#) (diguyWaypoint)
- [get_pitch_lower_limit](#) (diguyViewCamera)
- [get_pitch_lower_limit](#) (diguyViewCameraSettings)
- [get_pitch_offset](#) (diguyWaypoint)
- [get_pitch_out](#) (diguyWaypoint)
- [get_pitch_smooth](#) (diguyWaypoint)
- [get_pitch_upper_limit](#) (diguyViewCamera)
- [get_pitch_upper_limit](#) (diguyViewCameraSettings)
- [get_pixel_shader_filename](#) (diguyGraphicsShaderProgram)
- [get_playback_loop](#) (diguyScenario)
- [get_playback_mode](#) (diguyScenario)
- [get_point_at_distance](#) (diguyCharacterPath)
- [get_point_at_distance](#) (diguyPathShape)
- [get_point_in_region](#) (diguyRegion)
- [get_point_in_region_near_point](#) (diguyRegion)
- [get_point_los_function](#) (diguyCharacter)
- [get_point_position](#) (diguyAuthorVisualMultiline)
- [get_point_position_double](#) (diguyAuthorVisualMultiline)
- [get_port](#) (diguyNetInterface)
- [get_pose_array_size](#) (diguyCharacter)
- [get_pose_in_radians](#) (diguyCharacter)
- [get_pose_in_radians](#) (diguyCharacterPoseOverride)
- [get_pose_override_at_index](#) (diguyCharacter)
- [get_pose_update_rate](#) (diguyCharacter)
- [get_position](#) (diguyAuthorVisualObjectHandle)
- [get_position](#) (diguyAuthorVisualRegionPaintbrush)
- [get_position](#) (diguyCharacter)
- [get_position](#) (diguySceneObject)
- [get_position](#) (diguyViewButtonPanel)
- [get_position](#) (diguyViewLight)
- [get_position](#) (diguyWaypoint)
- [get_position_accumulation_disabled](#) (diguyCharacter)
- [get_position_acquired](#) (diguyCharacterGuide)
- [get_position_at_distance](#) (diguyCharacterPath)
- [get_position_at_distance_double](#) (diguyCharacterPath)
- [get_position_at_t](#) (diguyCharacterPath)
- [get_position_at_t_double](#) (diguyCharacterPath)
- [get_position_change_enabled](#) (diguyIGuyController)
- [get_position_double](#) (diguyAuthorVisualObjectHandle)

- [get_position_double \(diguyCharacter\)](#)
- [get_position_double \(diguyViewCamera\)](#)
- [get_position_double \(diguyWaypoint\)](#)
- [get_position_link \(diguyCharacter\)](#)
- [get_position_mode \(diguyCharacter\)](#)
- [get_position_offset \(diguyNetInterface\)](#)
- [get_position_offset \(diguyScenarioMergeSettings\)](#)
- [get_position_relative_to_parent \(diguyCharacter\)](#)
- [get_position_segment_offset \(diguyViewCamera\)](#)
- [get_position_update_rate \(diguyCharacter\)](#)
- [get_position_x \(diguyViewCamera\)](#)
- [get_position_x \(diguyViewCameraSettings\)](#)
- [get_position_y \(diguyViewCamera\)](#)
- [get_position_y \(diguyViewCameraSettings\)](#)
- [get_position_z \(diguyViewCamera\)](#)
- [get_position_z \(diguyViewCameraSettings\)](#)
- [get_posture \(diguyAgentParams\)](#)
- [get_posture \(diguyIGuyController\)](#)
- [get_posture_change_enabled \(diguyIGuyController\)](#)
- [get_precompute_quaternions_enabled \(diguyApp\)](#)
- [get_primary_view \(diguyScenario\)](#)
- [get_program_index_by_distance \(diguyGraphicsShaderTechnique\)](#)
- [get_progression_stage_duration \(diguyCharacterGesture\)](#)
- [get_progression_stage_optimal_duration \(diguyCharacterGesture\)](#)
- [get_project_filename \(diguyScenario\)](#)
- [get_projection_mode \(diguyViewCamera\)](#)
- [get_projection_mode \(diguyViewCameraSettings\)](#)
- [get_prop_culling_distance \(diguyScenario\)](#)
- [get_proportion \(diguyCrowdProfile\)](#)
- [get_pursue_offset \(diguyAgentParams\)](#)
- [get_pursue_speed_match_enabled \(diguyAgentParams\)](#)
- [get_pursue_speed_match_max_factor \(diguyAgentParams\)](#)
- [get_pursue_speed_match_update_dt \(diguyAgentParams\)](#)
- [get_pursue_uses_path_planner \(diguyAgentParams\)](#)
- [get_quad_point \(diguyAuthorVisualRegionMesh\)](#)
- [get_quads_data_ptr \(diguyAuthorVisualRegionMesh\)](#)
- [get_quality_level \(diguyGraphicsShaderProgram\)](#)
- [get_quat_data_index \(diguyGraphicsLink\)](#)
- [get_quat_data_ptr \(diguyGraphicsLink\)](#)
- [get_radius \(diguyAuthorVisualObjectHandle\)](#)
- [get_radius \(diguyChainSettings\)](#)
- [get_radius \(diguyFormation\)](#)
- [get_radius_variation \(diguyAgentParams\)](#)
- [get_rampdown_blend_function_name \(diguyCharacterGesture\)](#)
- [get_rampdown_duration \(diguyCharacterGesture\)](#)
- [get_rampup_blend_function_name \(diguyCharacterGesture\)](#)
- [get_rampup_duration \(diguyCharacterGesture\)](#)
- [get_random_active_character \(diguyCharacter\)](#)
- [get_random_active_character_in_group \(diguyCharacter\)](#)
- [get_random_entry_index \(diguyCrowdProfile\)](#)
- [get_random_factor \(diguyCharacter\)](#)

- [get_random_factor](#) (diguyRegion)
- [get_realtime_t](#) (diguyScenario)
- [get_reclamp_waypoints_flag](#) (diguyScenarioMergeSettings)
- [get_recommended_attachment_link_name](#) (diguyGraphicsShape)
- [get_reflected_entity_message_list](#) (diguyApp)
- [get_region](#) (diguyAuthorVisualRegionMesh)
- [get_region](#) (diguyCrowd)
- [get_region_at_index](#) (diguyScenario)
- [get_region_merge_flag](#) (diguyScenarioMergeSettings)
- [get_region_paintbrush](#) (diguyAuthorInterface)
- [get_register_count](#) (diguyCharacterFaceExpression)
- [get_register_enabled](#) (diguyCharacterFaceExpression)
- [get_register_enabled_by_index](#) (diguyCharacterFaceExpression)
- [get_register_name_by_index](#) (diguyCharacterFaceExpression)
- [get_register_value](#) (diguyCharacterFaceExpression)
- [get_register_value_by_index](#) (diguyCharacterFaceExpression)
- [get_rename_prefix](#) (diguyScenarioMergeSettings)
- [get_rename_suffix](#) (diguyScenarioMergeSettings)
- [get_render_mode_shader](#) (diguyCharacter)
- [get_render_mode_shader](#) (diguySceneObject)
- [get_render_pass_flags](#) (diguyApp)
- [get_replaying_history](#) (diguyScenario)
- [get_repulsion_factor](#) (diguyAgentParams)
- [get_repulsion_radius](#) (diguyAgentParams)
- [get_repulsion_zone_enabled](#) (diguyAgentParams)
- [get_require_smooth_servo](#) (diguyViewCamera)
- [get_require_smooth_servo](#) (diguyViewCameraSettings)
- [get_reset_flag](#) (diguyScenarioMergeSettings)
- [get_resolution](#) (diguyRegion)
- [get_rifle_state](#) (diguyCharacter)
- [get_role_name](#) (diguyFormation)
- [get_roll_coupled](#) (diguyWaypoint)
- [get_roll_coupling_offset](#) (diguyWaypoint)
- [get_roll_in](#) (diguyWaypoint)
- [get_roll_offset](#) (diguyWaypoint)
- [get_roll_out](#) (diguyWaypoint)
- [get_roll_smooth](#) (diguyWaypoint)
- [get_rotation](#) (diguyGraphicsLink)
- [get_rotation_data_index](#) (diguyGraphicsLink)
- [get_rotation_is_static](#) (diguyGraphicsLink)
- [get_samples_per_second](#) (diguySound)
- [get_scale](#) (diguyCharacter)
- [get_scale](#) (diguyGraphicsLink)
- [get_scale](#) (diguySceneObject)
- [get_scale_data_index](#) (diguyGraphicsLink)
- [get_scale_graphics_lod_ranges_from_view_settings](#) (diguyScenario)
- [get_scale_is_static](#) (diguyGraphicsLink)
- [get_scale_zone_extents_from_view_settings](#) (diguyLoadManager)
- [get_scenario](#) (diguyCharacter)
- [get_scenario](#) (diguyVariable)
- [get_scenario_at_index](#) (diguyApp)

- [get_scenario_camera](#) (diguyScenario)
- [get_scenario_camera_affected_by_script_events](#) (diguyScenario)
- [get_scenario_camera_affected_by_script_events_when_iguy_active](#) (diguyScenario)
- [get_scenario_fog](#) (diguyScenario)
- [get_scenario_light](#) (diguyScenario)
- [get_scenario_settings_merge_flag](#) (diguyScenarioMergeSettings)
- [get_scene_object_at_index](#) (diguyScenario)
- [get_scene_object_culling_enabled](#) (diguyScenario)
- [get_scene_object_file_name_from_ui](#) (diguyApp)
- [get_scene_object_grid_merge_flag](#) (diguyScenarioMergeSettings)
- [get_scene_object_merge_flag](#) (diguyScenarioMergeSettings)
- [get_scene_object_sub_culling_enabled](#) (diguyScenario)
- [get_scene_object_visualize_bounds_enabled](#) (diguyScenario)
- [get_scene_objects_cast_shadows](#) (diguyViewLight)
- [get_scene_objects_enabled](#) (diguyScenario)
- [get_script](#) (diguyCharacterPathScriptBead)
- [get_script_bead_at_index](#) (diguyCharacterPath)
- [get_script_merge_flag](#) (diguyScenarioMergeSettings)
- [get_script_type](#) (diguyCharacterPathScriptBead)
- [get_second_no_progress_trap_time_threshold](#) (diguyAgentParams)
- [get_second_no_progress_untrap_method](#) (diguyAgentParams)
- [get_secondary_view_at_index](#) (diguyScenario)
- [get_segment_origin](#) (diguyViewCamera)
- [get_selected](#) (diguyWaypoint)
- [get_selected_character](#) (diguyAuthorInterface)
- [get_selected_character_path](#) (diguyAuthorInterface)
- [get_selected_link](#) (diguyAuthorInterface)
- [get_selected_path_shape](#) (diguyAuthorInterface)
- [get_selected_shape](#) (diguyAuthorInterface)
- [get_selected_waypoint](#) (diguyAuthorInterface)
- [get_send_custom_pdus](#) (diguyNetInterface)
- [get_sensor_region_array_visible](#) (diguyScenario)
- [get_sensor_region_at_index](#) (diguyScenario)
- [get_sensor_region_merge_flag](#) (diguyScenarioMergeSettings)
- [get_septet_from_model](#) (diguyNetInterface)
- [get_servo_time_modifier](#) (diguyViewCamera)
- [get_settings](#) (diguyChainSimulation)
- [get_shader_by_distance](#) (diguyGraphicsShaderTechnique)
- [get_shader_instance](#) (diguyCharacter)
- [get_shader_instance](#) (diguyGraphicsMesh)
- [get_shader_instance](#) (diguySceneObject)
- [get_shader_lod](#) (diguyCharacter)
- [get_shader_lod_program_at_index](#) (diguyGraphicsShaderTechnique)
- [get_shader_lod_range_at_index](#) (diguyGraphicsShaderTechnique)
- [get_shader_matrix_data](#) (diguyGraphicsShape)
- [get_shader_matrix_data_ptr](#) (diguyGraphicsShaderInstance)
- [get_shader_matrix_index](#) (diguyGraphicsLink)
- [get_shader_program](#) (diguyCharacter)
- [get_shader_program](#) (diguyGraphicsMesh)
- [get_shader_program](#) (diguyGraphicsShaderInstance)
- [get_shader_program](#) (diguySceneObject)

- [get_shader_program_at_index](#) (diguyApp)
- [get_shader_technique](#) (diguyGraphicsShaderInstance)
- [get_shader_technique](#) (diguySceneObject)
- [get_shader_technique_at_index](#) (diguyApp)
- [get_shader_type](#) (diguyGraphicsShaderProgram)
- [get_shadow_map_size](#) (diguyScenario)
- [get_shadow_map_update_frequency](#) (diguyViewLight)
- [get_shadow_map_update_paused](#) (diguyViewLight)
- [get_shadow_offset_multiplier](#) (diguyViewLight)
- [get_shadow_pass_shader](#) (diguyGraphicsShaderTechnique)
- [get_shadow_volume_size](#) (diguyViewLight)
- [get_shape_at_index](#) (diguyGraphicsLink)
- [get_shape_callbacks_disabled](#) (diguyCharacter)
- [get_shape_set_filename_for_lod](#) (diguyGraphicsShape)
- [get_shape_set_name](#) (diguyGraphicsShape)
- [get_shape_set_shape_name](#) (diguyGraphicsShape)
- [get_shape_switches_disabled](#) (diguyCharacter)
- [get_shared_interaction_machine](#) (diguyScenario)
- [get_shininess](#) (diguyChainSettings)
- [get_shininess](#) (diguyGraphicsMaterial)
- [get_signal_at_index](#) (diguyScenario)
- [get_signal_merge_flag](#) (diguyScenarioMergeSettings)
- [get_simulation_dt](#) (diguyChainSettings)
- [get_site](#) (diguyNetInterface)
- [get_skip_transitions_flag](#) (diguyIGuyController)
- [get_sleep_timer](#) (diguyCharacter)
- [get_sorted_member_at_index](#) (diguyCharacterGroup)
- [get_sound_at_index](#) (diguyScenario)
- [get_sound_factory_is_initialized](#) (diguyApp)
- [get_sound_filename](#) (diguySound)
- [get_sound_merge_flag](#) (diguyScenarioMergeSettings)
- [get_spath_visible](#) (diguyScenario)
- [get_specular](#) (diguyGraphicsMaterial)
- [get_specular_color](#) (diguyChainSettings)
- [get_specular_color](#) (diguyViewLight)
- [get_specular_color](#) (diguyViewLightSettings)
- [get_speed](#) (diguyCharacter)
- [get_speed](#) (diguyVehicleController)
- [get_speed](#) (diguyViewCamera)
- [get_speed](#) (diguyViewCameraSettings)
- [get_speed_fov_degrees_per_wheel_click](#) (diguyViewCamera)
- [get_speed_fov_degrees_per_wheel_click](#) (diguyViewCameraSettings)
- [get_speed_meters_per_second](#) (diguyViewCamera)
- [get_speed_meters_per_second](#) (diguyViewCameraSettings)
- [get_speed_type_enabled](#) (diguyAgentParams)
- [get_speed_type_exact_action](#) (diguyAgentParams)
- [get_speed_type_radius](#) (diguyAgentParams)
- [get_speed_type_radius_hysteresis](#) (diguyAgentParams)
- [get_spotlight_angle](#) (diguyViewLight)
- [get_spring_b](#) (diguyChainSettings)
- [get_spring_k](#) (diguyChainSettings)

- `get_spring_length` (diguyChainSettings)
- `get_stage_duration` (diguyCharacterGesture)
- `get_stage_optimal_duration` (diguyCharacterGesture)
- `get_start` (diguyViewFog)
- `get_start` (diguyViewFogSettings)
- `get_state` (diguyGraphicsMesh)
- `get_state_id_at_index` (diguyInteractionMachine)
- `get_static_object_avoidance_method` (diguyCrowd)
- `get_stop_physics_sim_at_t` (diguyCharacter)
- `get_subgroup_number` (diguyCharacterGroup)
- `get_subgroup_size` (diguyFormation)
- `get_subregion_index` (diguyAuthorVisualRegionMesh)
- `get_svar` (diguyCharacter)
- `get_switch_state` (diguySceneObject)
- `get_switch_value` (diguyGraphicsShape)
- `get_switch_var_name` (diguyGraphicsShape)
- `get_switched_graphics_state` (diguyGraphicsShape)
- `get_symbolic_view` (diguyViewCamera)
- `get_sync_mode_camera_offset` (diguyNetInterface)
- `get_sync_mode_camera_slaved` (diguyNetInterface)
- `get_sync_mode_enabled` (diguyNetInterface)
- `get_sync_mode_master` (diguyNetInterface)
- `get_sync_mode_slave_cam` (diguyNetInterface)
- `get_t` (diguyScenario)
- `get_t_controlled_by_scenario_t` (diguyCharacter)
- `get_table_action_at_index` (diguyCharacterGesture)
- `get_table_action_begin` (diguyCharacterGesture)
- `get_table_action_desired` (diguyCharacterGesture)
- `get_table_action_end` (diguyCharacterGesture)
- `get_table_action_next` (diguyCharacterGesture)
- `get_table_action_now` (diguyCharacterGesture)
- `get_table_action_rep_duration` (diguyCharacterGesture)
- `get_table_transition_duration` (diguyCharacterGesture)
- `get_tangent_offset` (diguyGraphicsMesh)
- `get_tangent_stride` (diguyGraphicsMesh)
- `get_target` (diguyImpact)
- `get_target_name` (diguyImpact)
- `get_text` (diguyViewLabel)
- `get_text_justification` (diguyViewLabel)
- `get_text_offset_x` (diguyViewLabel)
- `get_text_offset_y` (diguyViewLabel)
- `get_texture` (diguyGraphicsState)
- `get_texture_compression_type` (diguyGraphicsTexture)
- `get_texture_filename` (diguyChainSettings)
- `get_texture_indices_offset` (diguyGraphicsMesh)
- `get_texture_indices_stride` (diguyGraphicsMesh)
- `get_texture_load_behavior` (diguyCharacter)
- `get_texture_load_behavior` (diguySceneObject)
- `get_texture_map_data_size` (diguyGraphicsTexture)
- `get_texture_type` (diguyGraphicsState)
- `get_texture_unload_frequency` (diguyApp)

- `get_textures_available` (diguyGraphicsState)
- `get_textures_used_bitmask` (diguyGraphicsShaderProgram)
- `get_threshold_to_unload_textures` (diguyApp)
- `get_tick_dt` (diguyScenario)
- `get_ticks_can_be_dropped` (diguyScenario)
- `get_time_of_day` (diguyScenario)
- `get_time_offset` (diguyScenarioMergeSettings)
- `get_time_required_to_reach_table_action_desired` (diguyCharacterGesture)
- `get_time_spent_path_planning_this_frame` (diguyScenario)
- `get_time_to_live` (diguyNetInterface)
- `get_time_to_reach_desired_action` (diguyCharacter)
- `get_time_to_transition` (diguyCharacter)
- `get_timeout_interval` (diguyNetInterface)
- `get_tin` (diguyCharacterGesture)
- `get_tin` (diguyCharacterPathActionBead)
- `get_tin` (diguyCharacterPathAimBead)
- `get_tin` (diguyCharacterPathDecisionBead)
- `get_tin` (diguyCharacterPathGazeBead)
- `get_tin` (diguyCharacterPathScriptBead)
- `get_tin_time_of_day` (diguyScenario)
- `get_tout` (diguyScenario)
- `get_tout_enabled` (diguyCharacter)
- `get_tout_enabled` (diguyScenario)
- `get_track_camera_avoids_obstacles` (diguyViewCamera)
- `get_track_camera_avoids_obstacles` (diguyViewCameraSettings)
- `get_track_group_name` (diguyViewCamera)
- `get_track_link_name` (diguyViewCamera)
- `get_track_link_name` (diguyViewCameraSettings)
- `get_track_name` (diguyViewCamera)
- `get_track_name` (diguyViewCameraSettings)
- `get_track_offset_is_in_world_coordinates` (diguyViewCamera)
- `get_track_offset_is_in_world_coordinates` (diguyViewCameraSettings)
- `get_track_offset_x` (diguyViewCamera)
- `get_track_offset_x` (diguyViewCameraSettings)
- `get_track_offset_y` (diguyViewCamera)
- `get_track_offset_y` (diguyViewCameraSettings)
- `get_track_offset_z` (diguyViewCamera)
- `get_track_offset_z` (diguyViewCameraSettings)
- `get_transformation_matrix` (diguyGraphicsLink)
- `get_transformation_matrix_4x4_ptr` (diguyGraphicsLink)
- `get_transformation_reference_point` (diguyScenarioMergeSettings)
- `get_transition_before_bead_flag` (diguyCharacterPathActionBead)
- `get_translation` (diguyGraphicsLink)
- `get_translation_data_index` (diguyGraphicsLink)
- `get_translation_double` (diguyGraphicsLink)
- `get_translation_is_static` (diguyGraphicsLink)
- `get_transposed_matrix_data` (diguyGraphicsShaderProgram)
- `get_turning_action_from_description` (diguyCharacter)
- `get_turning_motions_enabled` (diguyAgentParams)
- `get_type_map_category_at_index` (diguyScenario)
- `get_type_map_character_class_at_index` (diguyScenario)

- [get_type_map_field \(diguyCharacter\)](#)
- [get_type_map_field_using_string \(diguyCharacter\)](#)
- [get_type_map_id \(diguyCharacterTypeMap\)](#)
- [get_type_map_subcategory_at_index \(diguyScenario\)](#)
- [get_type_name \(diguyCharacter\)](#)
- [get_type_name \(diguyCrowd\)](#)
- [get_type_name \(diguyCrowdProfile\)](#)
- [get_type_name \(diguyFormation\)](#)
- [get_type_name \(diguyScenario\)](#)
- [get_type_name \(diguySensorRegion\)](#)
- [get_type_name \(diguySignal\)](#)
- [get_type_name \(diguySound\)](#)
- [get_type_name \(diguySoundInstance\)](#)
- [get_type_name \(diguyVariable\)](#)
- [get_type_name \(diguyView\)](#)
- [get_type_name \(diguyViewCamera\)](#)
- [get_type_name \(diguyViewCameraSettings\)](#)
- [get_type_name \(diguyViewFog\)](#)
- [get_type_name \(diguyViewFogSettings\)](#)
- [get_type_name \(diguyViewLabel\)](#)
- [get_type_name \(diguyViewLight\)](#)
- [get_type_name \(diguyViewLightSettings\)](#)
- [get_ui_appearance \(diguyInteractionMachine\)](#)
- [get_ui_name \(diguyCrowd\)](#)
- [get_uid \(diguyAuthorVisualMultiline\)](#)
- [get_uid \(diguyAuthorVisualObjectHandle\)](#)
- [get_uid \(diguyCharacter\)](#)
- [get_uid \(diguyCharacterPath\)](#)
- [get_uid \(diguyGraphicsLink\)](#)
- [get_uid \(diguyGraphicsMesh\)](#)
- [get_uid \(diguyGraphicsShape\)](#)
- [get_uid \(diguyPathShape\)](#)
- [get_uid \(diguySceneObject\)](#)
- [get_uid \(diguyWaypoint\)](#)
- [get_uid_base_type \(diguyAuthorInterface\)](#)
- [get_uid_base_type_from_string \(diguyAuthorInterface\)](#)
- [get_uniform_bind_location \(diguyGraphicsShaderProgram\)](#)
- [get_unique_color \(diguyGraphicsLink\)](#)
- [get_unique_color \(diguyGraphicsShape\)](#)
- [get_unique_crowd_name \(diguyScenario\)](#)
- [get_unit_leader \(diguyCrowd\)](#)
- [get_unit_type \(diguyCrowd\)](#)
- [get_unload_unused_after_scenario_load \(diguyApp\)](#)
- [get_unsaved \(diguyScenario\)](#)
- [get_untrap_method_ghost_duration \(diguyAgentParams\)](#)
- [get_untrap_method_random_turn_max_tries \(diguyAgentParams\)](#)
- [get_up_normal_in \(diguyWaypoint\)](#)
- [get_up_normal_out \(diguyWaypoint\)](#)
- [get_up_vector \(diguyCharacter\)](#)
- [get_use_4x3_shader_matrix_data \(diguyGraphicsShaderProgram\)](#)
- [get_use_UTM_standard_offsets \(diguyNetInterface\)](#)

- `get_use_adaptive_guide` (diguyNetInterface)
- `get_use_as_navmesh_portal` (diguyPathShape)
- `get_use_force_action_flag` (diguyIGuyController)
- `get_use_linear_traversal_draw` (diguyGraphicsLink)
- `get_use_local_tz_utc_offset_hours` (diguyScenario)
- `get_use_sRGB_textures` (diguyApp)
- `get_use_texture_variations` (diguyCharacter)
- `get_use_user_defined_coords` (diguyNetInterface)
- `get_user_data` (diguyCharacter)
- `get_utc_offset_hours` (diguyScenario)
- `get_utm` (diguyNetInterface)
- `get_valid` (diguyGraphicsShaderProgram)
- `get_value_as_float` (diguyVariable)
- `get_value_as_int` (diguyVariable)
- `get_value_as_string` (diguyVariable)
- `get_var_index` (diguyCharacterPoseOverride)
- `get_var_name_rx` (diguyGraphicsLink)
- `get_var_name_ry` (diguyGraphicsLink)
- `get_var_name_rz` (diguyGraphicsLink)
- `get_var_name_sx` (diguyGraphicsLink)
- `get_var_name_sy` (diguyGraphicsLink)
- `get_var_name_sz` (diguyGraphicsLink)
- `get_var_name_tx` (diguyGraphicsLink)
- `get_var_name_ty` (diguyGraphicsLink)
- `get_var_name_tz` (diguyGraphicsLink)
- `get_var_names` (diguyCharacterPoseOverride)
- `get_variable_at_index` (diguyCharacter)
- `get_variable_at_index` (diguyCharacterGroup)
- `get_variable_at_index` (diguyScenario)
- `get_variable_float` (diguyGraphicsShaderInstance)
- `get_variable_float` (diguyGraphicsShaderProgram)
- `get_variable_int` (diguyGraphicsShaderInstance)
- `get_variable_int` (diguyGraphicsShaderProgram)
- `get_variable_merge_flag` (diguyScenarioMergeSettings)
- `get_variable_name_at_index` (diguyGraphicsShaderInstance)
- `get_variable_name_at_index` (diguyGraphicsShaderProgram)
- `get_variable_type` (diguyGraphicsShaderInstance)
- `get_variable_type` (diguyGraphicsShaderProgram)
- `get_variable_vec2` (diguyGraphicsShaderInstance)
- `get_variable_vec2` (diguyGraphicsShaderProgram)
- `get_variable_vec3` (diguyGraphicsShaderInstance)
- `get_variable_vec3` (diguyGraphicsShaderProgram)
- `get_variable_vec4` (diguyGraphicsShaderInstance)
- `get_variable_vec4` (diguyGraphicsShaderProgram)
- `get_variant` (diguyAgentParams)
- `get_variant` (diguyIGuyController)
- `get_variant_change_enabled` (diguyIGuyController)
- `get_variation_texture_size` (diguyGraphicsShape)
- `get_vehicle_base_influence_box` (diguyAgentParams)
- `get_vehicle_body_orientation` (diguyCharacter)
- `get_vehicle_character` (diguyVehicleController)

- `get_vehicle_controller` (diguyCharacter)
- `get_vehicle_culling_distance` (diguyScenario)
- `get_vehicle_simulation_mode` (diguyVehicleController)
- `get_vehicle_smoothing_enabled` (diguyScenario)
- `get_velocity` (diguyCharacter)
- `get_vertex_buffer_data` (diguyGraphicsMesh)
- `get_vertex_buffer_size` (diguyGraphicsMesh)
- `get_vertex_count` (diguyGraphicsMesh)
- `get_vertex_format` (diguyGraphicsMesh)
- `get_vertex_info` (diguyGraphicsMesh)
- `get_vertex_offset` (diguyGraphicsMesh)
- `get_vertex_shader_filename` (diguyGraphicsShaderProgram)
- `get_vertex_stride` (diguyGraphicsMesh)
- `get_view` (diguyPluginKeyboardEvent)
- `get_view` (diguyPluginMouseEvent)
- `get_viewPainter` (diguyApp)
- `get_viewport_size` (diguyViewPainter)
- `get_viseme` (diguyCharacter)
- `get_visible` (diguyAuthorVisualMultiline)
- `get_visible` (diguyAuthorVisualObjectHandle)
- `get_visible` (diguyAuthorVisualRegionMesh)
- `get_visible` (diguyAuthorVisualRegionPaintbrush)
- `get_visible` (diguyAuthorVisualWaypoint)
- `get_visible` (diguySceneObject)
- `get_visible` (diguyViewLabel)
- `get_visual_material` (diguyAuthorVisualMultiline)
- `get_visual_material` (diguyAuthorVisualObjectHandle)
- `get_visual_material` (diguyAuthorVisualRegionMesh)
- `get_visual_material` (diguyAuthorVisualRegionPaintbrush)
- `get_visual_material` (diguyAuthorVisualWaypoint)
- `get_wander_uses_path_planner` (diguyAgentParams)
- `get_waypoint` (diguyAuthorVisualWaypoint)
- `get_waypoint_array_visible` (diguyScenario)
- `get_waypoint_at_index` (diguyCharacterPath)
- `get_waypoint_at_index` (diguyPathShape)
- `get_weapon_aim_vector` (diguyCharacter)
- `get_weapon_fire_count` (diguyCharacter)
- `get_weapon_fire_effect_data` (diguyScenario)
- `get_weapon_fire_effects_enabled` (diguyScenario)
- `get_weapon_fires_live_rounds` (diguyCharacter)
- `get_weapon_horizontal_error` (diguyCharacter)
- `get_weapon_max_azimuth_error` (diguyCharacter)
- `get_weapon_max_elevation_error` (diguyCharacter)
- `get_weapon_munition_type` (diguyCharacter)
- `get_weapon_munition_type_override` (diguyCharacter)
- `get_weapon_muzzle_munition_type` (diguyCharacter)
- `get_weapon_projectile_is_visible` (diguyCharacter)
- `get_weapon_rounds_until_tracer` (diguyCharacter)
- `get_weapon_shell_eject_enabled` (diguyCharacter)
- `get_weapon_smoke_enabled` (diguyCharacter)
- `get_weapon_sound_gain_override` (diguyCharacter)

- `get_weapon_sound_override` (diguyCharacter)
- `get_weapon_tracers_enabled` (diguyCharacter)
- `get_weapon_vertical_error` (diguyCharacter)
- `get_wedged_trap_time_threshold` (diguyAgentParams)
- `get_wedged_untrap_method` (diguyAgentParams)
- `get_weight_coupled` (diguyWaypoint)
- `get_weight_in` (diguyWaypoint)
- `get_weight_out` (diguyWaypoint)
- `get_width` (diguyGraphicsTexture)
- `get_width` (diguyView)
- `get_width` (diguyViewButtonPanel)
- `get_width` (diguyViewLabel)
- `get_window_justification_x` (diguyViewLabel)
- `get_window_justification_y` (diguyViewLabel)
- `get_window_title` (diguyView)
- `get_world_bounds` (diguyScenario)
- `get_x` (diguyImpact)
- `get_x` (diguyPluginKeyboardEvent)
- `get_x` (diguyPluginMouseEvent)
- `get_x` (diguyView)
- `get_x` (diguyViewLabel)
- `get_x` (diguyWaypoint)
- `get_xyz_under_mouse` (diguyApp)
- `get_y` (diguyImpact)
- `get_y` (diguyPluginKeyboardEvent)
- `get_y` (diguyPluginMouseEvent)
- `get_y` (diguyView)
- `get_y` (diguyViewLabel)
- `get_y` (diguyWaypoint)
- `get_yaw_coupled` (diguyWaypoint)
- `get_yaw_coupling_offset` (diguyWaypoint)
- `get_yaw_in` (diguyWaypoint)
- `get_yaw_offset` (diguyWaypoint)
- `get_yaw_out` (diguyWaypoint)
- `get_yaw_smooth` (diguyWaypoint)
- `get_z` (diguyImpact)
- `get_z` (diguyWaypoint)
- `get_zone_all_blends_disabled` (diguyLoadManager)
- `get_zone_altitude_update_rate` (diguyLoadManager)
- `get_zone_far_extent` (diguyLoadManager)
- `get_zone_graphics_api_shape_update_disabled` (diguyLoadManager)
- `get_zone_minimum_cpl` (diguyLoadManager)
- `get_zone_motion_lod` (diguyLoadManager)
- `get_zone_pose_update_rate` (diguyLoadManager)
- `get_zone_position_update_rate` (diguyLoadManager)
- `get_zone_shape_callbacks_disabled` (diguyLoadManager)
- `get_zone_shape_switches_disabled` (diguyLoadManager)
- `global_to_local` (diguyCharacter)
- `go_offline` (diguyNetInterface)
- `go_online` (diguyNetInterface)
- `greater_than_float` (diguyVariable)

- `greater_than_int` (diguyVariable)
- `greater_than_string` (diguyVariable)
- `ground_clamp` (diguyScenario)
- `has_additional_feature` (diguyGraphicsShaderProgram)
- `has_particle_description` (diguyScenario)
- `has_registered_character_event_handler` (diguyScenario)
- `has_registered_scenario_event_handler` (diguyApp)
- `has_registered_sensor_region_event_handler` (diguyScenario)
- `has_registered_signal_event_handler` (diguyScenario)
- `has_registered_variable_event_handler` (diguyScenario)
- `has_variable` (diguyGraphicsShaderInstance)
- `has_variable` (diguyGraphicsShaderProgram)
- `has_wakeup_callback` (diguyCharacter)
- `hide` (diguyAuthorVisualMultiline)
- `hide` (diguyAuthorVisualObjectHandle)
- `hide` (diguyAuthorVisualRegionMesh)
- `hide` (diguyAuthorVisualRegionPaintbrush)
- `hide` (diguyAuthorVisualWaypoint)
- `hide` (diguyGraphicsShape)
- `hide` (diguyInfoPopup)
- `hide` (diguySceneObject)
- `hide` (diguyView)
- `include_object_library` (diguyScenario)
- `increase_speed` (diguyViewCamera)
- `increment_value_as_int` (diguyVariable)
- `int`
- `intersect_static_geometry` (diguyScenario)
- `is_active_character_in_group_visible` (diguyCharacter)
- `is_api_mode` (diguyApp)
- `is_crowd_dead` (diguyCrowd)
- `is_current_behavior` (diguyCrowd)
- `is_current_behavior_and_focus_character` (diguyCrowd)
- `is_current_behavior_and_focus_group` (diguyCrowd)
- `is_debug` (diguyApp)
- `is_group_member` (diguyCharacter)
- `is_member` (diguyCharacterGroup)
- `is_member` (diguyCrowd)
- `is_module_enabled` (diguyApp)
- `is_playing_sound` (diguyCharacter)
- `is_point_in_region` (diguyRegion)
- `is_showing` (diguyInfoPopup)
- `is_showing` (diguyView)
- `is_speaking` (diguyCharacter)
- `is_valid_action` (diguyCharacter)
- `is_valid_character_group_pointer` (diguyScenario)
- `is_valid_character_pointer` (diguyScenario)
- `is_within_distance_n_of_character` (diguyCharacter)
- `is_within_distance_n_of_member_of_group` (diguyCharacter)
- `jump_to_action_bead` (diguyCharacter)
- `leave_path` (diguyCharacter)
- `legacy_appearance_effect_is_active` (diguyCharacter)

- `less_than_float` (diguyVariable)
- `less_than_int` (diguyVariable)
- `less_than_string` (diguyVariable)
- `link` (diguyGraphicsShaderProgram)
- `load` (diguyGraphicsFile)
- `load` (diguyGraphicsShaderProgram)
- `load` (diguyScenario)
- `load_camera_settings` (diguyScenario)
- `load_current_fog` (diguyScenario)
- `load_fog_settings` (diguyScenario)
- `load_light_settings` (diguyScenario)
- `load_module_library` (diguyApp)
- `load_package` (diguyScenario)
- `load_package_with_dependancy` (diguyScenario)
- `load_pixel_shader_source` (diguyGraphicsShaderProgram)
- `load_plugin` (diguyApp)
- `load_settings` (diguyViewCamera)
- `load_settings` (diguyViewFog)
- `load_settings` (diguyViewLight)
- `load_shader_program` (diguyApp)
- `load_transient_script_events` (diguyScenario)
- `load_vertex_shader_source` (diguyGraphicsShaderProgram)
- `local_to_global` (diguyCharacter)
- `look_at_character` (diguyScenario)
- `lpoint_at_angle` (diguyCharacter)
- `lpoint_at_angle_local` (diguyCharacter)
- `lpoint_at_character` (diguyCharacter)
- `lpoint_at_point` (diguyCharacter)
- `lpoint_at_point_local` (diguyCharacter)
- `lua_evaluate_global_function_4a_flex` (diguyScenario)
- `lua_evaluate_object_function` (diguyScenario)
- `lua_evaluate_object_function_2a` (diguyScenario)
- `lua_evaluate_object_function_3a` (diguyScenario)
- `lua_evaluate_object_function_4a` (diguyScenario)
- `lua_evaluate_object_function_4a_flex` (diguyScenario)
- `lua_get_object_field_as_string` (diguyScenario)
- `lua_log_printf` (diguyScenario)
- `lua_send_message_to_object` (diguyScenario)
- `make_instance` (diguyViewButtonPanel)
- `manually_invoke_callbacks_now` (diguyScenario)
- `manually_invoke_event_handler` (diguyCharacter)
- `manually_invoke_event_handler` (diguyScenario)
- `map_color_to_impact` (diguyScenario)
- `map_event_handler_to_callback_id` (diguyCharacter)
- `map_event_handler_to_callback_id` (diguyCrowd)
- `map_event_handler_to_callback_id` (diguyScenario)
- `map_event_handler_to_callback_id` (diguySensorRegion)
- `map_event_handler_to_callback_id` (diguySignal)
- `map_event_handler_to_callback_id` (diguyVariable)
- `maximize_current_cpl` (diguyCharacter)
- `memory_tracker_drop_marker` (diguyApp)

- `memory_tracker_print_report` (diguyApp)
- `merge` (diguyScenario)
- `merge_asset` (diguyCharacter)
- `merge_asset` (diguyScenario)
- `merge_object` (diguyCharacter)
- `merge_object` (diguyScenario)
- `nod_head` (diguyCharacter)
- `optimized_update` (diguyCharacter)
- `optimized_update` (diguyScenario)
- `paint` (diguyRegion)
- `paint_rectangle` (diguyRegion)
- `parse_dms_to_radians` (diguyNetInterface)
- `pick_shader_program` (diguyGraphicsShaderTechnique)
- `play` (diguySoundInstance)
- `play_3d_sound` (diguyCharacter)
- `play_3d_sound` (diguyScenario)
- `play_facefx_animation` (diguyCharacter)
- `play_sound` (diguyCharacter)
- `pop` (diguyGraphicsLink)
- `post_build` (diguyGraphicsLink)
- `post_draw` (diguyScenario)
- `preload_appearance` (diguyScenario)
- `preload_character_type` (diguyScenario)
- `preload_gesture` (diguyScenario)
- `preload_octree` (diguyScenario)
- `preload_sound` (diguyScenario)
- `print_shader_info_to_log` (diguyGraphicsShaderProgram)
- `print_to_log` (diguyScenario)
- `push` (diguyGraphicsLink)
- `push_path` (diguyCharacter)
- `push_qt_pointer_to_lua` (diguyScenario)
- `push_transient_script_event` (diguyScenario)
- `register_character_event_handler` (diguyScenario)
- `register_character_event_handler_from_library` (diguyScenario)
- `register_character_event_handler_script` (diguyScenario)
- `register_scenario_event_handler` (diguyApp)
- `register_scenario_event_handler_from_library` (diguyApp)
- `register_scenario_event_handler_script` (diguyApp)
- `register_sensor_region_event_handler` (diguyScenario)
- `register_sensor_region_event_handler_from_library` (diguyScenario)
- `register_sensor_region_event_handler_script` (diguyScenario)
- `register_signal_event_handler` (diguyScenario)
- `register_signal_event_handler_from_library` (diguyScenario)
- `register_signal_event_handler_script` (diguyScenario)
- `register_variable_event_handler` (diguyScenario)
- `register_variable_event_handler_from_library` (diguyScenario)
- `register_variable_event_handler_script` (diguyScenario)
- `reinitialize` (diguyInteractionMachine)
- `reload` (diguyGraphicsShaderProgram)
- `reload_shader_programs` (diguyApp)
- `reload_texture_files` (diguyApp)

- [remove_all_disabled_guides](#) (diguyCharacter)
- [remove_all_guides](#) (diguyCharacter)
- [remove_all_items](#) (diguyViewButtonPanel)
- [remove_all_members](#) (diguyCharacterGroup)
- [remove_all_members](#) (diguyCrowd)
- [remove_all_wakeup_callbacks](#) (diguyCharacter)
- [remove_appearance](#) (diguyScenario)
- [remove_callback](#) (diguyCharacter)
- [remove_callback](#) (diguyCharacterGesture)
- [remove_callback](#) (diguyCharacterPath)
- [remove_callback](#) (diguyCrowd)
- [remove_callback](#) (diguyScenario)
- [remove_callback](#) (diguySensorRegion)
- [remove_callback](#) (diguySignal)
- [remove_callback](#) (diguyVariable)
- [remove_callback](#) (diguyView)
- [remove_callback](#) (diguyViewCamera)
- [remove_callback](#) (diguyViewFog)
- [remove_callback](#) (diguyViewLabel)
- [remove_callback](#) (diguyViewLight)
- [remove_callback_script](#) (diguyCharacter)
- [remove_callback_script](#) (diguyCharacterGesture)
- [remove_callback_script](#) (diguyCrowd)
- [remove_callback_script](#) (diguyScenario)
- [remove_callback_script](#) (diguySensorRegion)
- [remove_callback_script](#) (diguySignal)
- [remove_callback_script](#) (diguyVariable)
- [remove_callback_script](#) (diguyView)
- [remove_callback_script](#) (diguyViewCamera)
- [remove_callback_script](#) (diguyViewFog)
- [remove_callback_script](#) (diguyViewLabel)
- [remove_callback_script](#) (diguyViewLight)
- [remove_callback_with_user_data](#) (diguyCharacter)
- [remove_callback_with_user_data](#) (diguyCharacterGesture)
- [remove_callback_with_user_data](#) (diguyCharacterPath)
- [remove_callback_with_user_data](#) (diguyCrowd)
- [remove_callback_with_user_data](#) (diguyScenario)
- [remove_callback_with_user_data](#) (diguySensorRegion)
- [remove_callback_with_user_data](#) (diguySignal)
- [remove_callback_with_user_data](#) (diguyVariable)
- [remove_callback_with_user_data](#) (diguyView)
- [remove_callback_with_user_data](#) (diguyViewCamera)
- [remove_callback_with_user_data](#) (diguyViewFog)
- [remove_callback_with_user_data](#) (diguyViewLabel)
- [remove_callback_with_user_data](#) (diguyViewLight)
- [remove_character_from_octtree](#) (diguyScenario)
- [remove_companion_crowd](#) (diguyCrowd)
- [remove_default_callback](#) (diguyScenario)
- [remove_default_callback_script](#) (diguyScenario)
- [remove_default_callback_with_user_data](#) (diguyScenario)
- [remove_default_camera_callback](#) (diguyScenario)

- `remove_default_camera_callback_script` (diguyScenario)
- `remove_default_camera_callback_with_user_data` (diguyScenario)
- `remove_default_character_callback` (diguyScenario)
- `remove_default_character_callback_script` (diguyScenario)
- `remove_default_character_callback_with_user_data` (diguyScenario)
- `remove_default_fog_callback` (diguyScenario)
- `remove_default_fog_callback_script` (diguyScenario)
- `remove_default_fog_callback_with_user_data` (diguyScenario)
- `remove_default_light_callback` (diguyScenario)
- `remove_default_light_callback_script` (diguyScenario)
- `remove_default_light_callback_with_user_data` (diguyScenario)
- `remove_default_path_callback` (diguyScenario)
- `remove_default_path_callback_with_user_data` (diguyScenario)
- `remove_default_scenario_callback` (diguyApp)
- `remove_default_scenario_callback_script` (diguyApp)
- `remove_default_scenario_callback_with_user_data` (diguyApp)
- `remove_default_sensor_region_callback` (diguyScenario)
- `remove_default_sensor_region_callback_script` (diguyScenario)
- `remove_default_sensor_region_callback_with_user_data` (diguyScenario)
- `remove_default_signal_callback` (diguyScenario)
- `remove_default_signal_callback_script` (diguyScenario)
- `remove_default_signal_callback_with_user_data` (diguyScenario)
- `remove_default_variable_callback` (diguyScenario)
- `remove_default_variable_callback_script` (diguyScenario)
- `remove_default_variable_callback_with_user_data` (diguyScenario)
- `remove_default_view_callback` (diguyScenario)
- `remove_default_view_callback_script` (diguyScenario)
- `remove_default_view_callback_with_user_data` (diguyScenario)
- `remove_guide` (diguyCharacter)
- `remove_guide_with_name` (diguyCharacter)
- `remove_input` (diguyInteractionMachine)
- `remove_member` (diguyCharacterGroup)
- `remove_member` (diguyCrowd)
- `remove_secondary_heartbeat` (diguyCharacter)
- `remove_state` (diguyInteractionMachine)
- `remove_wakeup_callback` (diguyCharacter)
- `rename_state` (diguyInteractionMachine)
- `resample_mesh` (diguyRegion)
- `reserve_num_threads` (diguyApp)
- `reset` (diguyImpact)
- `reset` (diguyInteractionMachine)
- `reset` (diguyScenario)
- `reset` (diguySignal)
- `reset` (diguyViewLabel)
- `reset_current_shader_to_default` (diguyCharacter)
- `reset_current_shader_to_default` (diguySceneObject)
- `reset_labels` (diguyScenario)
- `reset_render_mode_shader_to_default` (diguyCharacter)
- `reset_render_mode_shader_to_default` (diguySceneObject)
- `reset_shader_instance_variables_to_initial_values` (diguyCharacter)
- `reset_shader_instance_variables_to_initial_values` (diguySceneObject)

- [reset_shader_instance_variables_to_program_defaults \(diguyCharacter\)](#)
- [reset_shader_instance_variables_to_program_defaults \(diguySceneObject\)](#)
- [reset_signals \(diguyScenario\)](#)
- [reset_variables_to_initial_values \(diguyGraphicsShaderInstance\)](#)
- [reset_variables_to_program_defaults \(diguyGraphicsShaderInstance\)](#)
- [resolve_module_library_symbol \(diguyApp\)](#)
- [resume_interrupted_path \(diguyCharacter\)](#)
- [retrieve_character_from_recycle_bin \(diguyScenario\)](#)
- [revive_now \(diguyCharacter\)](#)
- [rotate_about_point \(diguyCharacterPath\)](#)
- [rotate_about_point \(diguyPathShape\)](#)
- [rotate_all_paths_about_current_waypoint \(diguyCharacter\)](#)
- [rotate_all_paths_about_current_waypoint \(diguyCharacterGroup\)](#)
- [rotate_all_paths_about_point \(diguyCharacter\)](#)
- [rotate_all_paths_about_point \(diguyCharacterGroup\)](#)
- [save \(diguyScenario\)](#)
- [save_as \(diguyScenario\)](#)
- [save_review_data \(diguyScenario\)](#)
- [save_transient_script_events \(diguyScenario\)](#)
- [select_input \(diguyInteractionMachine\)](#)
- [send_character_to_recycle_bin \(diguyScenario\)](#)
- [send_label_to_back \(diguyScenario\)](#)
- [send_label_to_front \(diguyScenario\)](#)
- [send_message_to_all_members \(diguyCharacterGroup\)](#)
- [send_message_to_all_members \(diguyCrowd\)](#)
- [send_message_to_character \(diguyExerciseInterface\)](#)
- [send_message_to_group \(diguyExerciseInterface\)](#)
- [send_pdu_xc_control \(diguyExerciseInterface\)](#)
- [send_pdu_xc_enter_slave_mode \(diguyExerciseInterface\)](#)
- [send_pdu_xc_eval_script \(diguyExerciseInterface\)](#)
- [send_pdu_xc_exit_slave_mode \(diguyExerciseInterface\)](#)
- [send_pdu_xc_load_scenario \(diguyExerciseInterface\)](#)
- [send_pdu_xc_merge_scenario \(diguyExerciseInterface\)](#)
- [send_pdu_xc_reset_scenario \(diguyExerciseInterface\)](#)
- [send_pdu_xc_save_scenario \(diguyExerciseInterface\)](#)
- [send_pdu_xc_set_tin_tod \(diguyExerciseInterface\)](#)
- [send_pdu_xc_time_now \(diguyExerciseInterface\)](#)
- [send_pdu_xc_trigger_script \(diguyExerciseInterface\)](#)
- [set_DIS_damage_state \(diguyCharacter\)](#)
- [set_DIS_flames_state \(diguyCharacter\)](#)
- [set_DIS_power_plant_state \(diguyCharacter\)](#)
- [set_DIS_smoke_state \(diguyCharacter\)](#)
- [set_DIS_smoke_trail_state \(diguyCharacter\)](#)
- [set_DIS_trailing_effect_state \(diguyCharacter\)](#)
- [set_action \(diguyCharacterPathActionBead\)](#)
- [set_action_bead_labels_visible \(diguyScenario\)](#)
- [set_action_channel_weight \(diguyCharacter\)](#)
- [set_active \(diguyViewLight\)](#)
- [set_aim \(diguyCharacter\)](#)
- [set_aim_algorithm \(diguyCharacter\)](#)
- [set_aim_converge_every_frame \(diguyCharacter\)](#)

- [set_aim_param_acquire_factor](#) (diguyCharacter)
- [set_aim_param_azimuth_acquired_range](#) (diguyCharacter)
- [set_aim_param_elevation_acquired_range](#) (diguyCharacter)
- [set_aim_param_max_per_frame_adj_factor](#) (diguyCharacter)
- [set_aim_param_new_aim_duration](#) (diguyCharacter)
- [set_aim_param_new_aim_factor](#) (diguyCharacter)
- [set_aim_trajectories_visible](#) (diguyScenario)
- [set_air_b](#) (diguyChainSettings)
- [set_all_blends_disabled](#) (diguyCharacter)
- [set_all_members_current_tin](#) (diguyCharacterGroup)
- [set_all_members_current_tout](#) (diguyCharacterGroup)
- [set_all_members_enabled](#) (diguyCharacterGroup)
- [set_all_members_enabled](#) (diguyCrowd)
- [set_all_merge_flags_to_dont_merge](#) (diguyScenarioMergeSettings)
- [set_all_registers_enabled](#) (diguyCharacterFaceExpression)
- [set_allow_toggle](#) (diguyViewButtonPanel)
- [set_altitude_function](#) (diguyCharacter)
- [set_altitude_function](#) (diguyScenario)
- [set_altitude_function_enabled_in_free_position_mode](#) (diguyCharacter)
- [set_altitude_function_enabled_in_path_position_mode](#) (diguyCharacter)
- [set_altitude_interpolation_rate](#) (diguyCharacter)
- [set_altitude_max_climb](#) (diguyScenario)
- [set_altitude_max_drop](#) (diguyScenario)
- [set_altitude_update_rate](#) (diguyCharacter)
- [set_always_run_flag](#) (diguyIGuyController)
- [set_ambient_color](#) (diguyChainSettings)
- [set_ambient_color](#) (diguyViewLight)
- [set_ambient_color](#) (diguyViewLightSettings)
- [set_appearance](#) (diguyCharacter)
- [set_appearance](#) (diguyCrowdProfile)
- [set_appearance_begin_cap](#) (diguyChainSettings)
- [set_appearance_end_cap](#) (diguyChainSettings)
- [set_apply_actor_scale_to_action_bead_travel](#) (diguyCharacter)
- [set_apply_actor_scale_to_action_bead_xy_travel](#) (diguyCharacter)
- [set_aspect](#) (diguyViewCamera)
- [set_aspect](#) (diguyViewCameraSettings)
- [set_attack_distance](#) (diguyAgentParams)
- [set_attack_fire_interval](#) (diguyAgentParams)
- [set_attack_stands_ground](#) (diguyAgentParams)
- [set_attack_zone](#) (diguyAgentParams)
- [set_attack_zone_enabled](#) (diguyAgentParams)
- [set_author_selection_handles_visible](#) (diguyScenario)
- [set_author_ui_panel_visible](#) (diguyAuthorInterface)
- [set_author_ui_select_code_browser_script](#) (diguyAuthorInterface)
- [set_author_ui_select_crowd_character](#) (diguyAuthorInterface)
- [set_author_ui_window_visible](#) (diguyAuthorInterface)
- [set_automatic_blink_duration_in](#) (diguyCharacter)
- [set_automatic_blink_duration_out](#) (diguyCharacter)
- [set_automatic_blink_target_in](#) (diguyCharacter)
- [set_automatic_blink_target_out](#) (diguyCharacter)
- [set_automatic_blinks_enabled](#) (diguyCharacter)

- [set_automatic_cpl_switching_enabled](#) (diguyCharacter)
- [set_automatic_destroy_flag](#) (diguyCharacterGesture)
- [set_automatic_destroy_flag](#) (diguySoundInstance)
- [set_automatic_graphics_lod_switching](#) (diguyScenario)
- [set_automatic_motion_lod_switching](#) (diguyScenario)
- [set_autosense_candidates_group](#) (diguySensorRegion)
- [set_autosense_members_group](#) (diguySensorRegion)
- [set_awareness_radius](#) (diguyCrowd)
- [set_background_color](#) (diguyViewLabel)
- [set_background_enabled](#) (diguyViewLabel)
- [set_background_image](#) (diguyViewLabel)
- [set_bar_graph_line_color](#) (diguyViewLabel)
- [set_bar_graph_line_magnitude_pixels](#) (diguyViewLabel)
- [set_bar_graph_line_magnitude_ratio](#) (diguyViewLabel)
- [set_base_datetime](#) (diguyScenario)
- [set_base_datetime_default_to_now](#) (diguyScenario)
- [set_base_datetime_default_to_today](#) (diguyScenario)
- [set_base_input_mode](#) (diguyApp)
- [set_bead_array_visible](#) (diguyScenario)
- [set_begin_at_object_name](#) (diguyCharacterPathAimBead)
- [set_begin_at_object_name](#) (diguyCharacterPathDecisionBead)
- [set_begin_at_object_name](#) (diguyCharacterPathGazeBead)
- [set_begin_at_object_name](#) (diguyCharacterPathScriptBead)
- [set_begin_at_object_offset](#) (diguyCharacterPathAimBead)
- [set_begin_at_object_offset](#) (diguyCharacterPathDecisionBead)
- [set_begin_at_object_offset](#) (diguyCharacterPathGazeBead)
- [set_begin_at_object_offset](#) (diguyCharacterPathScriptBead)
- [set_begin_state_entry_script](#) (diguyInteractionMachine)
- [set_behavior](#) (diguyAgentParams)
- [set_behavior_path_shape](#) (diguyAgentParams)
- [set_behavior_region](#) (diguyAgentParams)
- [set_behavior_region_border_is_solid](#) (diguyAgentParams)
- [set_behavior_subregion](#) (diguyAgentParams)
- [set_border_color](#) (diguyViewLabel)
- [set_border_enabled](#) (diguyViewLabel)
- [set_bounding_radius](#) (diguyCharacter)
- [set_brightness_multiplier](#) (diguyViewLight)
- [set_brush_color](#) (diguyViewPainter)
- [set_brush_size](#) (diguyAuthorVisualRegionPaintbrush)
- [set_button_name](#) (diguyViewButtonPanel)
- [set_cache_optimized_motion_files](#) (diguyApp)
- [set_camera_3rd_person_look_at_offset](#) (diguyIGuyController)
- [set_camera_3rd_person_look_from_offset](#) (diguyIGuyController)
- [set_camera_attachment_link](#) (diguyIGuyController)
- [set_camera_change_enabled](#) (diguyIGuyController)
- [set_camera_details_out_of_date](#) (diguyNetInterface)
- [set_camera_settings_merge_flag](#) (diguyScenarioMergeSettings)
- [set_camera_use_1st_person_flag](#) (diguyIGuyController)
- [set_careful_interpolation_threshold](#) (diguyCharacter)
- [set_casts_shadows](#) (diguyViewLight)
- [set_cfg_file](#) (diguyNetInterface)

- `set_chain_mass_connection_character` (diguyChainSimulation)
- `set_chain_mass_position` (diguyChainSimulation)
- `set_chain_mass_position_is_fixed` (diguyChainSimulation)
- `set_chain_mass_positions_along_path_shape` (diguyChainSimulation)
- `set_chain_settings_merge_flag` (diguyScenarioMergeSettings)
- `set_channel_weight` (diguyCharacterGesture)
- `set_channel_weight_target` (diguyCharacterGesture)
- `set_character_culling_distance` (diguyScenario)
- `set_character_culling_enabled` (diguyScenario)
- `set_character_group` (diguyScenarioMergeSettings)
- `set_character_labels_visible` (diguyScenario)
- `set_character_loss_function` (diguyCharacter)
- `set_character_merge_flag` (diguyScenarioMergeSettings)
- `set_character_type` (diguyCharacter)
- `set_character_type` (diguyCrowdProfile)
- `set_character_type_and_appearance` (diguyCrowdProfile)
- `set_character_visible` (diguyScenario)
- `set_character_visualize_bounds_enabled` (diguyScenario)
- `set_checkpoint_frequency` (diguyScenario)
- `set_checkpointing_enabled` (diguyScenario)
- `set_clamp_at_end_flag` (diguyPathShape)
- `set_coil_taper_begin_mass_count` (diguyChainSettings)
- `set_coil_taper_end_mass_count` (diguyChainSettings)
- `set_coil_thickness` (diguyChainSettings)
- `set_collision_group_detection_flag` (diguyApp)
- `set_color` (diguyViewFog)
- `set_color` (diguyViewFogSettings)
- `set_companion_waypoint_name` (diguyCharacterPathActionBead)
- `set_coordinate_system_type` (diguyNetInterface)
- `set_create_network_crowds` (diguyScenario)
- `set_cross_section` (diguyChainSettings)
- `set_crowd_behavior_visible` (diguyScenario)
- `set_crowd_feelers_visible` (diguyScenario)
- `set_crowd_influence_visible` (diguyScenario)
- `set_crowd_merge_flag` (diguyScenarioMergeSettings)
- `set_crowd_profile_merge_flag` (diguyScenarioMergeSettings)
- `set_crowd_regions_visible` (diguyScenario)
- `set_culling_enabled` (diguyNetInterface)
- `set_current_appearance` (diguyCharacter)
- `set_current_behavior` (diguyCrowd)
- `set_current_behavior_path_shape` (diguyCrowd)
- `set_current_behavior_region` (diguyCrowd)
- `set_current_behavior_region_border_is_solid` (diguyCrowd)
- `set_current_character` (diguyScenario)
- `set_current_character_simulator` (diguyCharacter)
- `set_current_cpl` (diguyCharacter)
- `set_current_crowd` (diguyScenario)
- `set_current_crowd_profile` (diguyScenario)
- `set_current_crowd_profile_by_name` (diguyScenario)
- `set_current_datetime` (diguyScenario)
- `set_current_datetime_advance_factor` (diguyScenario)

- [set_current_datetime_advance_method](#) (diguyScenario)
- [set_current_facefx_animation_group](#) (diguyCharacter)
- [set_current_focus_character](#) (diguyCrowd)
- [set_current_focus_group](#) (diguyCrowd)
- [set_current_head_appearance](#) (diguyCharacter)
- [set_current_painting_subregion](#) (diguyApp)
- [set_current_params_from_profile](#) (diguyCrowd)
- [set_current_physics_simulation_mode](#) (diguyCharacter)
- [set_current_postures](#) (diguyCrowd)
- [set_current_region](#) (diguyScenario)
- [set_current_render_mode](#) (diguyScenario)
- [set_current_settings_index](#) (diguyInteractionMachine)
- [set_current_state](#) (diguyInteractionMachine)
- [set_current_tin](#) (diguyCharacter)
- [set_current_tin_to_now](#) (diguyCharacter)
- [set_current_tout](#) (diguyCharacter)
- [set_current_tout_to_now](#) (diguyCharacter)
- [set_current_variants](#) (diguyCrowd)
- [set_current_voice_actor](#) (diguyCharacter)
- [set_custom_orientation_rx_and_ry](#) (diguyCharacter)
- [set_datetimes_are_utc](#) (diguyScenario)
- [set_decision_merge_flag](#) (diguyScenarioMergeSettings)
- [set_default_acceleration](#) (diguyVehicleController)
- [set_default_agent_feeler_function](#) (diguyScenario)
- [set_default_character_altitude_function](#) (diguyScenario)
- [set_default_character_los_function](#) (diguyScenario)
- [set_default_character_scene_graph_mask](#) (diguyScenario)
- [set_default_character_shader](#) (diguyScenario)
- [set_default_crowd_size](#) (diguyCrowdProfile)
- [set_default_deceleration](#) (diguyVehicleController)
- [set_default_fire_weapon_intersection_function](#) (diguyScenario)
- [set_default_intersection_function](#) (diguyScenario)
- [set_default_lod_ranges](#) (diguyScenario)
- [set_default_particle_system_shader](#) (diguyScenario)
- [set_default_point_los_function](#) (diguyScenario)
- [set_default_scene_object_scene_graph_mask](#) (diguyScenario)
- [set_default_scene_object_shader](#) (diguyScenario)
- [set_default_weapon_muzzle](#) (diguyCharacter)
- [set_default_weapon_sound](#) (diguyCharacter)
- [set_default_weapon_sound_gain](#) (diguyCharacter)
- [set_default_weight](#) (diguyCharacterPoseOverride)
- [set_density](#) (diguyViewFog)
- [set_density](#) (diguyViewFogSettings)
- [set_depth](#) (diguyViewButtonPanel)
- [set_derive_desired_duration_flag](#) (diguyCharacterPathActionBead)
- [set_derive_duration_flag](#) (diguyCharacterPathActionBead)
- [set_derive_length_flag](#) (diguyCharacterPathActionBead)
- [set_derive_motion_reps_flag](#) (diguyCharacterPathActionBead)
- [set_desired_action](#) (diguyCharacter)
- [set_desired_duration](#) (diguyCharacterPathActionBead)
- [set_desired_fix](#) (diguyViewCamera)

- `set_desired_front_wheels_steer_angle` (diguyVehicleController)
- `set_desired_navpath_distance_threshold` (diguyAgentParams)
- `set_desired_orientation` (diguyCharacter)
- `set_desired_orientation_to_current_orientation` (diguyCharacter)
- `set_desired_orientation_towards_position` (diguyCharacter)
- `set_desired_position` (diguyCharacter)
- `set_desired_position` (diguyViewCamera)
- `set_desired_position_to_current_position` (diguyCharacter)
- `set_desired_position_to_path_shape_waypoint` (diguyCharacter)
- `set_desired_position_to_path_waypoint` (diguyCharacter)
- `set_desired_position_to_waypoint` (diguyCharacter)
- `set_desired_posture` (diguyCharacter)
- `set_desired_speed_with_acceleration` (diguyVehicleController)
- `set_desired_speed_with_acceleration_time` (diguyVehicleController)
- `set_desired_variant` (diguyCharacter)
- `set_desired_velocity` (diguyCharacter)
- `set_diffuse_color` (diguyChainSettings)
- `set_diffuse_color` (diguyViewLight)
- `set_diffuse_color` (diguyViewLightSettings)
- `set_dimensions` (diguyViewButtonPanel)
- `set_direction` (diguyViewLight)
- `set_direction` (diguyViewLightSettings)
- `set_display_remote_chars` (diguyNetInterface)
- `set_distance` (diguyViewCamera)
- `set_distance` (diguyViewCameraSettings)
- `set_distance_along_path` (diguyCharacter)
- `set_distance_into_path` (diguyCharacterPathActionBead)
- `set_distance_into_path` (diguyCharacterPathAimBead)
- `set_distance_into_path` (diguyCharacterPathDecisionBead)
- `set_distance_into_path` (diguyCharacterPathGazeBead)
- `set_distance_into_path` (diguyCharacterPathScriptBead)
- `set_distance_smooth` (diguyWaypoint)
- `set_dof_state` (diguySceneObject)
- `set_dof_state_flt` (diguySceneObject)
- `set_dolly_link_name` (diguyViewCamera)
- `set_dolly_link_name` (diguyViewCameraSettings)
- `set_dolly_name` (diguyViewCamera)
- `set_dolly_name` (diguyViewCameraSettings)
- `set_dolly_offset` (diguyViewCamera)
- `set_dolly_offset` (diguyViewCameraSettings)
- `set_dolly_offset_is_in_world_coordinates` (diguyViewCamera)
- `set_dolly_offset_is_in_world_coordinates` (diguyViewCameraSettings)
- `set_draw_character_shadow_disks` (diguyScenario)
- `set_draw_limit` (diguyViewLabel)
- `set_drawn_by_scenario_flag` (diguyCharacter)
- `set_duration` (diguyCharacterPathActionBead)
- `set_duration` (diguyCharacterPathAimBead)
- `set_duration` (diguyCharacterPathDecisionBead)
- `set_duration` (diguyCharacterPathGazeBead)
- `set_duration` (diguyCharacterPathScriptBead)
- `set_dynamic_object_avoidance_method` (diguyCrowd)

- `set_emit_light` (diguyCharacter)
- `set_enabled` (diguyCharacter)
- `set_enabled` (diguyCharacterGuide)
- `set_enabled` (diguyCrowd)
- `set_enabled` (diguyLoadManager)
- `set_enabled` (diguySceneObject)
- `set_end` (diguyViewFog)
- `set_end` (diguyViewFogSettings)
- `set_end_at_object_name` (diguyCharacterPathAimBead)
- `set_end_at_object_name` (diguyCharacterPathDecisionBead)
- `set_end_at_object_name` (diguyCharacterPathGazeBead)
- `set_end_at_object_name` (diguyCharacterPathScriptBead)
- `set_end_at_object_offset` (diguyCharacterPathAimBead)
- `set_end_at_object_offset` (diguyCharacterPathDecisionBead)
- `set_end_at_object_offset` (diguyCharacterPathGazeBead)
- `set_end_at_object_offset` (diguyCharacterPathScriptBead)
- `set_eval_decisions_as_scripts` (diguyScenario)
- `set_executing_table_gesture_desired_action` (diguyCharacter)
- `set_exercise` (diguyNetInterface)
- `set_eye_droop` (diguyCharacter)
- `set_face_exp_merge_flag` (diguyScenarioMergeSettings)
- `set_face_expression` (diguyCharacter)
- `set_face_ry_desired` (diguyIGuyController)
- `set_face_rz_desired` (diguyIGuyController)
- `set_facefx_node_value` (diguyCharacter)
- `set_far_plane` (diguyViewCamera)
- `set_far_plane` (diguyViewCameraSettings)
- `set_far_plane` (diguyViewLight)
- `set_federation_name` (diguyNetInterface)
- `set_feeler_collision_distance` (diguyAgentParams)
- `set_feeler_enabled` (diguyAgentParams)
- `set_feeler_height` (diguyAgentParams)
- `set_feeler_rz_offset` (diguyAgentParams)
- `set_feeler_turn_direction` (diguyAgentParams)
- `set_feeler_turn_distance` (diguyAgentParams)
- `set_filename` (diguySceneObject)
- `set_fire_weapon_intersection_function` (diguyScenario)
- `set_first_no_progress_trap_time_threshold` (diguyAgentParams)
- `set_first_no_progress_untrap_method` (diguyAgentParams)
- `set_first_subgroup_index` (diguyFormation)
- `set_fix` (diguyViewCamera)
- `set_fix` (diguyViewCameraSettings)
- `set_fix_double` (diguyViewCamera)
- `set_focus_affects_gaze` (diguyAgentParams)
- `set_focus_character` (diguyAgentParams)
- `set_focus_group` (diguyAgentParams)
- `set_fog_mode` (diguyViewFog)
- `set_fog_mode` (diguyViewFogSettings)
- `set_fog_settings_merge_flag` (diguyScenarioMergeSettings)
- `set_follower_offset` (diguyFormation)
- `set_follower_rz_offset` (diguyFormation)

- `set_formation_merge_flag` (diguyScenarioMergeSettings)
- `set_fov` (diguyViewCamera)
- `set_fov` (diguyViewCameraSettings)
- `set_front_wheels_steer_angle` (diguyVehicleController)
- `set_full_window` (diguyViewLabel)
- `set_fvar` (diguyCharacter)
- `set_gaze_can_change_posture` (diguyCharacter)
- `set_gaze_can_change_variant` (diguyCharacter)
- `set_gaze_change_enabled` (diguyIGuyController)
- `set_gaze_link_azimuth_limits` (diguyCharacter)
- `set_gaze_link_azimuth_tolerance` (diguyCharacter)
- `set_gaze_link_elevation_limits` (diguyCharacter)
- `set_gaze_link_elevation_tolerance` (diguyCharacter)
- `set_gaze_link_max_angular_vel` (diguyCharacter)
- `set_gaze_link_scale_factor` (diguyCharacter)
- `set_gaze_link_uses_desired_locate_time` (diguyCharacter)
- `set_gaze_param_desired_locate_time` (diguyCharacter)
- `set_gaze_params_to_defaults` (diguyCharacter)
- `set_global_wind` (diguyScenario)
- `set_graphics_api_node_ptr` (diguyCharacter)
- `set_graphics_api_node_ptr` (diguyScenario)
- `set_graphics_api_shape_update_disabled` (diguyCharacter)
- `set_graphics_attach_ptr` (diguyScenario)
- `set_graphics_lod` (diguyCharacter)
- `set_graphics_lod` (diguyGraphicsShape)
- `set_graphics_lod_ranges` (diguyGraphicsShape)
- `set_graphics_ptr` (diguyCharacter)
- `set_gravity` (diguyChainSettings)
- `set_ground_b` (diguyChainSettings)
- `set_ground_clamp_max_climb` (diguyNetInterface)
- `set_ground_clamp_max_drop` (diguyNetInterface)
- `set_ground_clamp_reflected_land_entities_enabled` (diguyNetInterface)
- `set_ground_clamp_reflected_land_entities_from_host` (diguyNetInterface)
- `set_ground_clamp_when_on_path` (diguyCharacterPath)
- `set_ground_clamp_when_on_path` (diguyPathShape)
- `set_ground_height` (diguyChainSettings)
- `set_ground_k` (diguyChainSettings)
- `set_ground_lateral_b` (diguyChainSettings)
- `set_ground_stiction_engage_vel` (diguyChainSettings)
- `set_ground_stiction_release_force` (diguyChainSettings)
- `set_group_merge_flag` (diguyScenarioMergeSettings)
- `set_guide_algorithm_float_parameter` (diguyCharacterGuide)
- `set_guide_algorithm_string_parameter` (diguyCharacterGuide)
- `set_guide_merge_flag` (diguyScenarioMergeSettings)
- `set_heading` (diguyInteractionMachine)
- `set_heading_color_bg` (diguyInteractionMachine)
- `set_heading_color_fg` (diguyInteractionMachine)
- `set_heading_font` (diguyInteractionMachine)
- `set_heading_visible` (diguyInteractionMachine)
- `set_heartbeat_dt` (diguyCharacter)
- `set_height` (diguyView)

- set_hide_author_selection_handles_during_play (diguyScenario)
- set_history_max_sound_instances (diguyScenario)
- set_history_size_increment_t (diguyCharacter)
- set_history_size_t (diguyCharacter)
- set_history_type (diguyCharacter)
- set_history_type (diguyScenario)
- set_hit_link_index (diguyImpact)
- set_hit_scene_object (diguyImpact)
- set_hit_shape_index (diguyImpact)
- set_ignore_mouse drags (diguyViewCamera)
- set_ignore_mouse drags (diguyViewCameraSettings)
- set_iguy_character (diguyIGuyController)
- set_impact_been_processed (diguyImpact)
- set_impact_number (diguyImpact)
- set_include_in_octtree (diguySceneObject)
- set_include_in_physics_sim_scene (diguySceneObject)
- set_index_of_chain_mass_connected_to_character (diguyChainSettings)
- set_influence_box_enabled (diguyAgentParams)
- set_influence_center_shift (diguyAgentParams)
- set_info (diguyInteractionMachine)
- set_info_color_bg (diguyInteractionMachine)
- set_info_color_fg (diguyInteractionMachine)
- set_info_font (diguyInteractionMachine)
- set_info_popup_default_encoding (diguyScenario)
- set_info_popup_merge_flag (diguyScenarioMergeSettings)
- set_info_text (diguySignal)
- set_info_visible (diguyInteractionMachine)
- set_initial_character_simulator (diguyCharacter)
- set_initial_heading (diguyInteractionMachine)
- set_initial_info (diguyInteractionMachine)
- set_initial_motion_texture (diguyCharacter)
- set_initial_motion_texture_gain (diguyCharacter)
- set_initial_orientation (diguyCharacter)
- set_initial_params_from_profile (diguyCrowd)
- set_initial_path (diguyCharacter)
- set_initial_physics_simulation_mode (diguyCharacter)
- set_initial_position (diguyCharacter)
- set_initial_render_mode (diguyScenario)
- set_initial_tin (diguyCharacter)
- set_initial_tout (diguyCharacter)
- set_initial_ui_appearance (diguyInteractionMachine)
- set_initial_up_vector (diguyCharacter)
- set_initial_value_as_float (diguyVariable)
- set_initial_value_as_int (diguyVariable)
- set_initial_value_as_string (diguyVariable)
- set_input_capture_enabled (diguyIGuyController)
- set_input_response_script (diguyInteractionMachine)
- set_input_shortcuts_visible (diguyInteractionMachine)
- set_inputs_color_bg (diguyInteractionMachine)
- set_inputs_color_fg (diguyInteractionMachine)
- set_inputs_font (diguyInteractionMachine)

- `set_inputs_visible` (diguyInteractionMachine)
- `set_interaction_subject` (diguyInteractionMachine)
- `set_internal_float` (diguyScenario)
- `set_internal_int` (diguyScenario)
- `set_interpolation_flag` (diguyCharacterGesture)
- `set_intersection_function` (diguyScenario)
- `set_invisible_flag` (diguyCharacter)
- `set_ipaddr` (diguyNetInterface)
- `set_is_3d_sound_flag` (diguySoundInstance)
- `set_is_clickable` (diguyViewLabel)
- `set_is_frozen` (diguyChainSimulation)
- `set_is_immune_to_impacts` (diguyCharacter)
- `set_is_load_managed` (diguyCharacter)
- `set_is_network_published` (diguyCharacter)
- `set_is_noise` (diguyViewLabel)
- `set_is_scene_object` (diguyCharacter)
- `set_is_sky_box` (diguySceneObject)
- `set_is_temporary` (diguyViewLabel)
- `set_item_visibility` (diguyViewButtonPanel)
- `set_ivar` (diguyCharacter)
- `set_joystick_input_enabled` (diguyIGuyController)
- `set_justifications` (diguyViewButtonPanel)
- `set_key_pressed` (diguyAuthorInterface)
- `set_keyboard_alt` (diguyAuthorInterface)
- `set_keyboard_ctrl` (diguyAuthorInterface)
- `set_keyboard_input_enabled` (diguyIGuyController)
- `set_keyboard_shift` (diguyAuthorInterface)
- `set_label_background_color` (diguyCharacter)
- `set_label_background_color_vec4` (diguyCharacter)
- `set_label_is_visible` (diguyCharacter)
- `set_label_link_offset_override` (diguyCharacter)
- `set_label_offset_above_centroid` (diguyCharacterGroup)
- `set_label_shows_character_state` (diguyCharacter)
- `set_label_text` (diguyCharacter)
- `set_label_text_color` (diguyCharacter)
- `set_label_text_color_vec4` (diguyCharacter)
- `set_label_text_shadow_color` (diguyCharacter)
- `set_label_text_shadow_color_vec4` (diguyCharacter)
- `set_label_text_shadow_is_visible` (diguyCharacter)
- `set_last_clicked_xyz` (diguyApp)
- `set_last_env_impact_info` (diguyScenario)
- `set_last_impact_been_processed` (diguyCharacter)
- `set_last_impact_been_processed` (diguyScenario)
- `set_last_input_variable` (diguyInteractionMachine)
- `set_lat_lon` (diguyNetInterface)
- `set_length` (diguyCharacterPathActionBead)
- `set_length` (diguyCharacterPathAimBead)
- `set_length` (diguyCharacterPathDecisionBead)
- `set_length` (diguyCharacterPathGazeBead)
- `set_length` (diguyCharacterPathScriptBead)
- `set_library_function_merge_flag` (diguyScenarioMergeSettings)

- set_light_array_visible (diguyScenario)
- set_light_settings_merge_flag (diguyScenarioMergeSettings)
- set_line_width (diguyViewPainter)
- set_link_enabled_for_gaze (diguyCharacter)
- set_link_enabled_for_lpoint (diguyCharacter)
- set_location (diguyImpact)
- set_lod_ranges (diguyCharacter)
- set_lua_object_waiting_for_input (diguyApp)
- set_lua_objects_visible (diguyScenario)
- set_margin_size (diguyViewLabel)
- set_mass (diguyChainSettings)
- set_material_parameter_4f (diguyGraphicsShape)
- set_max_LOS_travel_distance (diguyAgentParams)
- set_max_character_shader_quality_level (diguyApp)
- set_max_distance_planning_multiplier (diguyScenario)
- set_max_focus_distance (diguyAgentParams)
- set_max_fov (diguyViewCamera)
- set_max_fov (diguyViewCameraSettings)
- set_max_orientation_change_rate (diguyAgentParams)
- set_max_reps (diguyCharacterGesture)
- set_max_scene_object_shader_quality_level (diguyApp)
- set_max_shader_quality_level (diguyCharacter)
- set_max_ticks_behind_until_drop (diguyScenario)
- set_max_track_distance (diguyViewCamera)
- set_max_unprocessed_impacts (diguyCharacter)
- set_max_unprocessed_impacts (diguyScenario)
- set_mesh_region_array_visible (diguyScenario)
- set_min_fov (diguyViewCamera)
- set_min_fov (diguyViewCameraSettings)
- set_min_track_distance (diguyViewCamera)
- set_mind_always_updates (diguyCharacter)
- set_mind_base_class (diguyCrowdProfile)
- set_mind_field_as_float (diguyCharacter)
- set_mind_field_as_int (diguyCharacter)
- set_mind_field_as_string (diguyCharacter)
- set_minimize_script (diguyViewButtonPanel)
- set_minimized_state (diguyViewButtonPanel)
- set_minimum_cpl (diguyCharacter)
- set_minimum_full_update_period (diguyCharacter)
- set_minimum_full_update_period (diguyLoadManager)
- set_minimum_height (diguyViewLabel)
- set_minimum_width (diguyViewLabel)
- set_motion_interpolation_flag (diguyCharacter)
- set_motion_lod (diguyCharacter)
- set_motion_reps (diguyCharacterPathActionBead)
- set_mouse_button_l (diguyAuthorInterface)
- set_mouse_button_m (diguyAuthorInterface)
- set_mouse_button_r (diguyAuthorInterface)
- set_mouse_input_enabled (diguyIGuyController)
- set_mouse_look_changes_orientation (diguyIGuyController)
- set_mouse_screen_coordinates (diguyAuthorInterface)

- `set_mouse_wheel_change` (diguyAuthorInterface)
- `set_mouse_world_coordinates` (diguyAuthorInterface)
- `set_move_mode` (diguyViewCamera)
- `set_move_mode` (diguyViewCameraSettings)
- `set_move_on_t_max` (diguyAgentParams)
- `set_move_on_t_min` (diguyAgentParams)
- `set_multithreaded_path_planning_enabled` (diguyScenario)
- `set_munition_type_DIS_impact_septet` (diguyApp)
- `set_munition_type_DIS_tracer_septet` (diguyApp)
- `set_name` (diguyChainSettings)
- `set_name` (diguyCharacter)
- `set_name` (diguyCharacterGroup)
- `set_name` (diguyCharacterGuide)
- `set_name` (diguyCharacterPath)
- `set_name` (diguyCharacterPathActionBead)
- `set_name` (diguyCharacterPathAimBead)
- `set_name` (diguyCharacterPathDecisionBead)
- `set_name` (diguyCharacterPathGazeBead)
- `set_name` (diguyCharacterPathScriptBead)
- `set_name` (diguyCrowd)
- `set_name` (diguyCrowdProfile)
- `set_name` (diguyFormation)
- `set_name` (diguyInfoPopup)
- `set_name` (diguyPathShape)
- `set_name` (diguySceneObject)
- `set_name` (diguySensorRegion)
- `set_name` (diguySignal)
- `set_name` (diguyVariable)
- `set_near_plane` (diguyViewCamera)
- `set_near_plane` (diguyViewCameraSettings)
- `set_near_plane` (diguyViewLight)
- `set_negative_wraps_x` (diguyViewLabel)
- `set_negative_wraps_y` (diguyViewLabel)
- `set_neighbor2_spring_b` (diguyChainSettings)
- `set_neighbor2_spring_k` (diguyChainSettings)
- `set_neighbor2_spring_length` (diguyChainSettings)
- `set_network_datetime_broadcast_interval` (diguyNetInterface)
- `set_network_datetime_enable_diguy_custom_pdus` (diguyNetInterface)
- `set_network_datetime_enable_dis_start_resume_pdus` (diguyNetInterface)
- `set_network_datetime_mode` (diguyNetInterface)
- `set_network_marking` (diguyCharacter)
- `set_network_translation` (diguyScenario)
- `set_nod` (diguyCharacter)
- `set_normal` (diguyImpact)
- `set_num_active_point_lights` (diguyScenario)
- `set_num_coils` (diguyChainSettings)
- `set_num_followers` (diguyFormation)
- `set_num_masses` (diguyChainSettings)
- `set_num_rendering_passes` (diguyApp)
- `set_num_segments` (diguyWaypoint)
- `set_num_texture_repetitions` (diguyChainSettings)

- set_num_threads (diguyApp)
- set_one_way_portal (diguyPathShape)
- set_open_script (diguyViewButtonPanel)
- set_orientation (diguyCharacter)
- set_orientation (diguySceneObject)
- set_orientation (diguyViewCamera)
- set_orientation (diguyViewCameraSettings)
- set_orientation_acquired (diguyCharacterGuide)
- set_orientation_change_enabled (diguyIGuyController)
- set_orientation_eye_left (diguyCharacter)
- set_orientation_eye_right (diguyCharacter)
- set_orientation_eyes (diguyCharacter)
- set_orientation_in (diguyWaypoint)
- set_orientation_offset (diguyScenarioMergeSettings)
- set_orientation_out (diguyWaypoint)
- set_orientation_relative_to_parent (diguyCharacter)
- set_orientation_roll (diguyViewCamera)
- set_orientation_roll (diguyViewCameraSettings)
- set_overall_duration (diguyCharacterGesture)
- set_overall_history_type (diguyScenario)
- set_painting_subregion (diguyAuthorVisualRegionPaintbrush)
- set_paintrush_mode (diguyAuthorVisualRegionPaintbrush)
- set_parent (diguyCharacter)
- set_particle_module_disabled (diguyScenario)
- set_particle_system_merge_flag (diguyScenarioMergeSettings)
- set_path_follow_pause_t_when_off_path (diguyAgentParams)
- set_path_follow_position_lookahead_distance (diguyAgentParams)
- set_path_follow_unpause_t_at_radius (diguyAgentParams)
- set_path_planner_cost_rule (diguyAgentParams)
- set_path_planner_fallback_region (diguyAgentParams)
- set_path_planner_neutral_cost_bias (diguyAgentParams)
- set_path_planner_preferred_cost_bias (diguyAgentParams)
- set_path_planner_preferred_region_mask (diguyAgentParams)
- set_path_planner_repulsed_cost_bias (diguyAgentParams)
- set_path_planner_repulsed_region_mask (diguyAgentParams)
- set_path_shape_merge_flag (diguyScenarioMergeSettings)
- set_pause_scenario_while_active (diguyInteractionMachine)
- set_pen2_color (diguyViewPainter)
- set_pen_color (diguyViewPainter)
- set_phoneme_filename (diguySound)
- set_physics_collision_group (diguyCharacter)
- set_physics_collision_group (diguySceneObject)
- set_physics_sim_ground_plane_enabled (diguyApp)
- set_physics_sim_ground_plane_height (diguyApp)
- set_pitch_coupled (diguyWaypoint)
- set_pitch_coupling_offset (diguyWaypoint)
- set_pitch_in (diguyWaypoint)
- set_pitch_lower_limit (diguyViewCamera)
- set_pitch_lower_limit (diguyViewCameraSettings)
- set_pitch_offset (diguyWaypoint)
- set_pitch_out (diguyWaypoint)

- [set_pitch_smooth \(diguyWaypoint\)](#)
- [set_pitch_upper_limit \(diguyViewCamera\)](#)
- [set_pitch_upper_limit \(diguyViewCameraSettings\)](#)
- [set_playback_loop \(diguyScenario\)](#)
- [set_playback_mode \(diguyScenario\)](#)
- [set_playback_mode_play \(diguyScenario\)](#)
- [set_playback_mode_stop \(diguyScenario\)](#)
- [set_point_los_function \(diguyCharacter\)](#)
- [set_port \(diguyNetInterface\)](#)
- [set_pose_in_radians \(diguyCharacterPoseOverride\)](#)
- [set_pose_update_rate \(diguyCharacter\)](#)
- [set_position \(diguyCharacter\)](#)
- [set_position \(diguySceneObject\)](#)
- [set_position \(diguyViewButtonPanel\)](#)
- [set_position \(diguyViewCamera\)](#)
- [set_position \(diguyViewCameraSettings\)](#)
- [set_position \(diguyViewLight\)](#)
- [set_position \(diguyWaypoint\)](#)
- [set_position_accumulation_disabled \(diguyCharacter\)](#)
- [set_position_acquired \(diguyCharacterGuide\)](#)
- [set_position_change_enabled \(diguyIGuyController\)](#)
- [set_position_double \(diguyCharacter\)](#)
- [set_position_double \(diguyViewCamera\)](#)
- [set_position_double \(diguyWaypoint\)](#)
- [set_position_offset \(diguyNetInterface\)](#)
- [set_position_offset \(diguyScenarioMergeSettings\)](#)
- [set_position_relative_to_parent \(diguyCharacter\)](#)
- [set_position_to_other_character_position \(diguyCharacter\)](#)
- [set_position_update_rate \(diguyCharacter\)](#)
- [set_posture \(diguyAgentParams\)](#)
- [set_posture \(diguyIGuyController\)](#)
- [set_posture_change_enabled \(diguyIGuyController\)](#)
- [set_precompute_quaternions_enabled \(diguyApp\)](#)
- [set_progression_stage_duration \(diguyCharacterGesture\)](#)
- [set_project_filename \(diguyScenario\)](#)
- [set_projection_mode \(diguyViewCamera\)](#)
- [set_projection_mode \(diguyViewCameraSettings\)](#)
- [set_prop_culling_distance \(diguyScenario\)](#)
- [set_proportion \(diguyCrowdProfile\)](#)
- [set_publish_local_chars \(diguyNetInterface\)](#)
- [set_pursue_offset \(diguyAgentParams\)](#)
- [set_pursue_speed_match_enabled \(diguyAgentParams\)](#)
- [set_pursue_speed_match_max_factor \(diguyAgentParams\)](#)
- [set_pursue_speed_match_update_dt \(diguyAgentParams\)](#)
- [set_pursue_uses_path_planner \(diguyAgentParams\)](#)
- [set_radius \(diguyChainSettings\)](#)
- [set_radius_variation \(diguyAgentParams\)](#)
- [set_rampdown_blend_function_name \(diguyCharacterGesture\)](#)
- [set_rampdown_duration \(diguyCharacterGesture\)](#)
- [set_rampup_blend_function_name \(diguyCharacterGesture\)](#)
- [set_rampup_duration \(diguyCharacterGesture\)](#)

- `set_reclamp_waypoints_flag` (diguyScenarioMergeSettings)
- `set_reflected_entity_message_list` (diguyApp)
- `set_region_merge_flag` (diguyScenarioMergeSettings)
- `set_register_enabled` (diguyCharacterFaceExpression)
- `set_register_enabled_by_index` (diguyCharacterFaceExpression)
- `set_register_value` (diguyCharacterFaceExpression)
- `set_register_value_by_index` (diguyCharacterFaceExpression)
- `set_rename_prefix` (diguyScenarioMergeSettings)
- `set_rename_suffix` (diguyScenarioMergeSettings)
- `set_render_camera` (diguyScenario)
- `set_render_mode_shader` (diguyCharacter)
- `set_render_mode_shader` (diguySceneObject)
- `set_render_pass_flags` (diguyApp)
- `set_repulsion_factor` (diguyAgentParams)
- `set_repulsion_radius` (diguyAgentParams)
- `set_repulsion_zone_enabled` (diguyAgentParams)
- `set_require_smooth_servo` (diguyViewCamera)
- `set_require_smooth_servo` (diguyViewCameraSettings)
- `set_reset_flag` (diguyScenarioMergeSettings)
- `set_role_name` (diguyFormation)
- `set_roll_coupled` (diguyWaypoint)
- `set_roll_coupling_offset` (diguyWaypoint)
- `set_roll_in` (diguyWaypoint)
- `set_roll_offset` (diguyWaypoint)
- `set_roll_out` (diguyWaypoint)
- `set_roll_smooth` (diguyWaypoint)
- `set_scale` (diguyCharacter)
- `set_scale` (diguySceneObject)
- `set_scale_graphics_lod_ranges_from_view_settings` (diguyScenario)
- `set_scale_zone_extents_from_view_settings` (diguyLoadManager)
- `set_scenario_camera_affected_by_script_events` (diguyScenario)
- `set_scenario_camera_affected_by_script_events_when_iguy_active` (diguyScenario)
- `set_scenario_settings_merge_flag` (diguyScenarioMergeSettings)
- `set_scene_graph_mask` (diguyCharacter)
- `set_scene_graph_mask` (diguySceneObject)
- `set_scene_object_culling_enabled` (diguyScenario)
- `set_scene_object_grid_merge_flag` (diguyScenarioMergeSettings)
- `set_scene_object_merge_flag` (diguyScenarioMergeSettings)
- `set_scene_object_sub_culling_enabled` (diguyScenario)
- `set_scene_object_visualize_bounds_enabled` (diguyScenario)
- `set_scene_objects_cast_shadows` (diguyViewLight)
- `set_scene_objects_enabled` (diguyScenario)
- `set_script` (diguyCharacterPathScriptBead)
- `set_script_merge_flag` (diguyScenarioMergeSettings)
- `set_script_type` (diguyCharacterPathScriptBead)
- `set_second_no_progress_trap_time_threshold` (diguyAgentParams)
- `set_second_no_progress_untrap_method` (diguyAgentParams)
- `set_secondary_heartbeat_dt` (diguyCharacter)
- `set_selected_button` (diguyViewButtonPanel)
- `set_selected_button_by_name` (diguyViewButtonPanel)
- `set_selected_diguy_uid` (diguyAuthorInterface)

- `set_selected_diguy_uid_from_string` (diguyAuthorInterface)
- `set_send_custom_pdus` (diguyNetInterface)
- `set_sensor_region_array_visible` (diguyScenario)
- `set_sensor_region_merge_flag` (diguyScenarioMergeSettings)
- `set_servo_time_modifier` (diguyViewCamera)
- `set_shader_instance` (diguyGraphicsLink)
- `set_shader_lod` (diguyCharacter)
- `set_shader_program` (diguySceneObject)
- `set_shadow_map_size` (diguyScenario)
- `set_shadow_map_update_frequency` (diguyViewLight)
- `set_shadow_map_update_paused` (diguyViewLight)
- `set_shadow_offset_multiplier` (diguyViewLight)
- `set_shadow_volume_size` (diguyViewLight)
- `set_shape_callbacks_disabled` (diguyCharacter)
- `set_shape_switches_disabled` (diguyCharacter)
- `set_shininess` (diguyChainSettings)
- `set_signal_merge_flag` (diguyScenarioMergeSettings)
- `set_simulation_dt` (diguyChainSettings)
- `set_site` (diguyNetInterface)
- `set_skip_transitions_flag` (diguyIGuyController)
- `set_sleep_timer` (diguyCharacter)
- `set_slider_value` (diguyViewButtonPanel)
- `set_sound_filename` (diguySound)
- `set_sound_merge_flag` (diguyScenarioMergeSettings)
- `set_spacing` (diguyViewButtonPanel)
- `set_spath_visible` (diguyScenario)
- `set_specular_color` (diguyChainSettings)
- `set_specular_color` (diguyViewLight)
- `set_specular_color` (diguyViewLightSettings)
- `set_speed` (diguyCharacter)
- `set_speed` (diguyVehicleController)
- `set_speed` (diguyViewCamera)
- `set_speed` (diguyViewCameraSettings)
- `set_speed_fov_degrees_per_wheel_click` (diguyViewCamera)
- `set_speed_fov_degrees_per_wheel_click` (diguyViewCameraSettings)
- `set_speed_meters_per_second` (diguyViewCamera)
- `set_speed_meters_per_second` (diguyViewCameraSettings)
- `set_speed_type_enabled` (diguyAgentParams)
- `set_speed_type_exact_action` (diguyAgentParams)
- `set_speed_type_radius` (diguyAgentParams)
- `set_speed_type_radius_hysteresis` (diguyAgentParams)
- `set_spotlight_angle` (diguyViewLight)
- `set_spring_b` (diguyChainSettings)
- `set_spring_k` (diguyChainSettings)
- `set_spring_length` (diguyChainSettings)
- `set_stage_duration` (diguyCharacterGesture)
- `set_start` (diguyViewFog)
- `set_start` (diguyViewFogSettings)
- `set_state_entry_script` (diguyInteractionMachine)
- `set_static_object_avoidance_method` (diguyCrowd)
- `set_stop_physics_sim_at_t` (diguyCharacter)

- `set_stop_physics_sim_in_t_seconds` (diguyCharacter)
- `set_style_sheet` (diguyViewButtonPanel)
- `set_subgroup_size` (diguyFormation)
- `set_svar` (diguyCharacter)
- `set_switch_state` (diguySceneObject)
- `set_sync_mode_camera_offset` (diguyNetInterface)
- `set_sync_mode_camera_slaved` (diguyNetInterface)
- `set_sync_mode_enabled` (diguyNetInterface)
- `set_sync_mode_master` (diguyNetInterface)
- `set_sync_mode_slave_cam` (diguyNetInterface)
- `set_t_controlled_by_scenario_t` (diguyCharacter)
- `set_t_scale_factor` (diguyCharacter)
- `set_table_action_begin` (diguyCharacterGesture)
- `set_table_action_desired` (diguyCharacterGesture)
- `set_table_action_end` (diguyCharacterGesture)
- `set_target` (diguyImpact)
- `set_temporary_current_shader` (diguyCharacter)
- `set_temporary_current_shader` (diguySceneObject)
- `set_text` (diguyViewLabel)
- `set_text_color` (diguyViewLabel)
- `set_text_justification` (diguyViewLabel)
- `set_text_offset_x` (diguyViewLabel)
- `set_text_offset_y` (diguyViewLabel)
- `set_text_shadow_color` (diguyViewLabel)
- `set_text_shadow_enabled` (diguyViewLabel)
- `set_texture_filename` (diguyChainSettings)
- `set_texture_load_behavior` (diguyCharacter)
- `set_texture_load_behavior` (diguySceneObject)
- `set_texture_name` (diguyViewLight)
- `set_texture_unload_frequency` (diguyApp)
- `set_textures_used_bitmask` (diguyGraphicsShaderProgram)
- `set_threshold_to_unload_textures` (diguyApp)
- `set_tick_dt` (diguyScenario)
- `set_ticks_can_be_dropped` (diguyScenario)
- `set_time_offset` (diguyScenarioMergeSettings)
- `set_time_to_live` (diguyNetInterface)
- `set_timeout_interval` (diguyNetInterface)
- `set_tin` (diguyCharacterPathActionBead)
- `set_tin` (diguyCharacterPathAimBead)
- `set_tin` (diguyCharacterPathDecisionBead)
- `set_tin` (diguyCharacterPathGazeBead)
- `set_tin` (diguyCharacterPathScriptBead)
- `set_tin_time_of_day` (diguyScenario)
- `set_title` (diguyViewButtonPanel)
- `set_tout` (diguyScenario)
- `set_tout_enabled` (diguyCharacter)
- `set_tout_enabled` (diguyScenario)
- `set_track_camera_avoids_obstacles` (diguyViewCamera)
- `set_track_camera_avoids_obstacles` (diguyViewCameraSettings)
- `set_track_group_name` (diguyViewCamera)
- `set_track_link_name` (diguyViewCamera)

- `set_track_link_name` (diguyViewCameraSettings)
- `set_track_name` (diguyViewCamera)
- `set_track_name` (diguyViewCameraSettings)
- `set_track_offset` (diguyViewCamera)
- `set_track_offset` (diguyViewCameraSettings)
- `set_track_offset_is_in_world_coordinates` (diguyViewCamera)
- `set_track_offset_is_in_world_coordinates` (diguyViewCameraSettings)
- `set_transformation_reference_point` (diguyScenarioMergeSettings)
- `set_transition_before_bead_flag` (diguyCharacterPathActionBead)
- `set_turning_motions_enabled` (diguyAgentParams)
- `set_ui_appearance` (diguyInteractionMachine)
- `set_ui_name` (diguyCrowd)
- `set_unit_leader` (diguyCrowd)
- `set_unit_type` (diguyCrowd)
- `set_unload_unused_after_scenario_load` (diguyApp)
- `set_unsaved` (diguyScenario)
- `set_untrap_method_ghost_duration` (diguyAgentParams)
- `set_untrap_method_random_turn_max_tries` (diguyAgentParams)
- `set_up_vector` (diguyCharacter)
- `set_use_UTM_standard_offsets` (diguyNetInterface)
- `set_use_adaptive_guide` (diguyNetInterface)
- `set_use_as_navmesh_portal` (diguyPathShape)
- `set_use_database_coords` (diguyNetInterface)
- `set_use_force_action_flag` (diguyIGuyController)
- `set_use_linear_traversal_draw` (diguyGraphicsLink)
- `set_use_local_tz_utc_offset_hours` (diguyScenario)
- `set_use_texture_variations` (diguyCharacter)
- `set_use_user_defined_coords` (diguyNetInterface)
- `set_user_data` (diguyCharacter)
- `set_utc_offset_hours` (diguyScenario)
- `set_utm` (diguyNetInterface)
- `set_utm_zone` (diguyNetInterface)
- `set_valid_impact` (diguyImpact)
- `set_value_as_float` (diguyVariable)
- `set_value_as_int` (diguyVariable)
- `set_value_as_string` (diguyVariable)
- `set_variable_float` (diguyGraphicsShaderInstance)
- `set_variable_float` (diguyGraphicsShaderProgram)
- `set_variable_float` (diguyGraphicsShaderTechnique)
- `set_variable_int` (diguyGraphicsShaderInstance)
- `set_variable_int` (diguyGraphicsShaderProgram)
- `set_variable_int` (diguyGraphicsShaderTechnique)
- `set_variable_merge_flag` (diguyScenarioMergeSettings)
- `set_variable_vec2` (diguyGraphicsShaderInstance)
- `set_variable_vec2` (diguyGraphicsShaderProgram)
- `set_variable_vec2` (diguyGraphicsShaderTechnique)
- `set_variable_vec3` (diguyGraphicsShaderInstance)
- `set_variable_vec3` (diguyGraphicsShaderProgram)
- `set_variable_vec3` (diguyGraphicsShaderTechnique)
- `set_variable_vec4` (diguyGraphicsShaderInstance)
- `set_variable_vec4` (diguyGraphicsShaderProgram)

- `set_variable_vec4` (diguyGraphicsShaderTechnique)
- `set_variant` (diguyAgentParams)
- `set_variant` (diguyIGuyController)
- `set_variant_change_enabled` (diguyIGuyController)
- `set_vehicle_base_influence_box` (diguyAgentParams)
- `set_vehicle_culling_distance` (diguyScenario)
- `set_vehicle_simulation_mode` (diguyVehicleController)
- `set_vehicle_smoothing_enabled` (diguyScenario)
- `set_viseme` (diguyCharacter)
- `set_visible` (diguySceneObject)
- `set_visible` (diguyViewButtonPanel)
- `set_visible` (diguyViewLabel)
- `set_wander_max_wander_dist` (diguyAgentParams)
- `set_wander_sets_desired_orientation` (diguyAgentParams)
- `set_wander_uses_path_planner` (diguyAgentParams)
- `set_waypoint_array_visible` (diguyScenario)
- `set_weapon_far_range` (diguyCharacter)
- `set_weapon_fire_effects_enabled` (diguyScenario)
- `set_weapon_fires_live_rounds` (diguyCharacter)
- `set_weapon_hit_ratio` (diguyCharacter)
- `set_weapon_horizontal_error` (diguyCharacter)
- `set_weapon_max_azimuth_error` (diguyCharacter)
- `set_weapon_max_elevation_error` (diguyCharacter)
- `set_weapon_max_impacts` (diguyCharacter)
- `set_weapon_munition_type` (diguyCharacter)
- `set_weapon_munition_type_override` (diguyCharacter)
- `set_weapon_near_range` (diguyCharacter)
- `set_weapon_projectile_is_visible` (diguyCharacter)
- `set_weapon_shell_eject_enabled` (diguyCharacter)
- `set_weapon_smoke_enabled` (diguyCharacter)
- `set_weapon_sound_gain_override` (diguyCharacter)
- `set_weapon_sound_override` (diguyCharacter)
- `set_weapon_spread` (diguyCharacter)
- `set_weapon_tracers_enabled` (diguyCharacter)
- `set_weapon_vertical_error` (diguyCharacter)
- `set_wedged_trap_time_threshold` (diguyAgentParams)
- `set_wedged_untrap_method` (diguyAgentParams)
- `set_weight_coupled` (diguyWaypoint)
- `set_weight_in` (diguyWaypoint)
- `set_weight_out` (diguyWaypoint)
- `set_width` (diguyView)
- `set_window_justification_x` (diguyViewLabel)
- `set_window_justification_y` (diguyViewLabel)
- `set_window_title` (diguyView)
- `set_world_bounds` (diguyScenario)
- `set_x` (diguyView)
- `set_x` (diguyViewLabel)
- `set_x` (diguyWaypoint)
- `set_y` (diguyView)
- `set_y` (diguyViewLabel)
- `set_y` (diguyWaypoint)

- [set_yaw_coupled \(diguyWaypoint\)](#)
- [set_yaw_coupling_offset \(diguyWaypoint\)](#)
- [set_yaw_in \(diguyWaypoint\)](#)
- [set_yaw_offset \(diguyWaypoint\)](#)
- [set_yaw_out \(diguyWaypoint\)](#)
- [set_yaw_smooth \(diguyWaypoint\)](#)
- [set_z \(diguyWaypoint\)](#)
- [set_zone_all_blends_disabled \(diguyLoadManager\)](#)
- [set_zone_altitude_update_rate \(diguyLoadManager\)](#)
- [set_zone_far_extent \(diguyLoadManager\)](#)
- [set_zone_graphics_api_shape_update_disabled \(diguyLoadManager\)](#)
- [set_zone_minimum_cpl \(diguyLoadManager\)](#)
- [set_zone_motion_lod \(diguyLoadManager\)](#)
- [set_zone_pose_update_rate \(diguyLoadManager\)](#)
- [set_zone_position_update_rate \(diguyLoadManager\)](#)
- [set_zone_shape_callbacks_disabled \(diguyLoadManager\)](#)
- [set_zone_shape_switches_disabled \(diguyLoadManager\)](#)
- [shake_head \(diguyCharacter\)](#)
- [show \(diguyAuthorVisualMultiline\)](#)
- [show \(diguyAuthorVisualObjectHandle\)](#)
- [show \(diguyAuthorVisualRegionMesh\)](#)
- [show \(diguyAuthorVisualRegionPaintbrush\)](#)
- [show \(diguyAuthorVisualWaypoint\)](#)
- [show \(diguyGraphicsShape\)](#)
- [show \(diguyInfoPopup\)](#)
- [show \(diguySceneObject\)](#)
- [show \(diguyView\)](#)
- [show_generate_navmesh_ui \(diguyApp\)](#)
- [show_notification_with_shared_interaction_machine \(diguyScenario\)](#)
- [shrink \(diguyRegion\)](#)
- [signal_palette_hide \(diguySignal\)](#)
- [signal_palette_hide_all \(diguyScenario\)](#)
- [signal_palette_show \(diguySignal\)](#)
- [signal_palette_show_all \(diguyScenario\)](#)
- [sort_group_based_on_pos \(diguyCharacterGroup\)](#)
- [speak \(diguyCharacter\)](#)
- [stop \(diguySoundInstance\)](#)
- [stop_all_appearance_effects \(diguyCharacter\)](#)
- [stop_all_sounds \(diguyCharacter\)](#)
- [stop_all_sounds \(diguyScenario\)](#)
- [stop_appearance_effect \(diguyCharacter\)](#)
- [stop_facefx_animations \(diguyCharacter\)](#)
- [stop_sound \(diguyCharacter\)](#)
- [this_character \(diguyScenario\)](#)
- [this_event_bead \(diguyScenario\)](#)
- [to_lua_string \(diguyFormation\)](#)
- [toggle_earth_coord_format \(diguyNetInterface\)](#)
- [translate \(diguyCharacterPath\)](#)
- [translate \(diguyPathShape\)](#)
- [translate_all_paths \(diguyCharacter\)](#)
- [translate_all_paths \(diguyCharacterGroup\)](#)

- trigger (diguySignal)
- trigger_decision (diguyScenario)
- trigger_decision_delayed (diguyScenario)
- trigger_detonation (diguyScenario)
- trigger_detonation_on_character (diguyCharacter)
- trigger_script (diguyScenario)
- trigger_script_delayed (diguyScenario)
- trigger_shader_refresh (diguyCharacter)
- triggered (diguySignal)
- triggered_at_least_n_times (diguySignal)
- unbind (diguyGraphicsShaderInstance)
- unbind (diguyGraphicsShaderProgram)
- unbuild (diguyAuthorVisualMultiline)
- unbuild (diguyAuthorVisualObjectHandle)
- unbuild (diguyAuthorVisualRegionMesh)
- unbuild (diguyAuthorVisualRegionPaintbrush)
- unbuild (diguyAuthorVisualWaypoint)
- unbuild (diguyGraphicsLink)
- unbuild (diguyGraphicsMaterial)
- unbuild (diguyGraphicsMesh)
- unbuild (diguyGraphicsShape)
- unbuild (diguyGraphicsState)
- unbuild (diguyGraphicsTexture)
- uninclude_object_library (diguyScenario)
- unload (diguyGraphicsFile)
- unload_all_plugins (diguyApp)
- unload_module_library (diguyApp)
- unload_plugin (diguyApp)
- unload_unreferenced_graphics_files (diguyApp)
- unmap_event_handler_from_callback_id (diguyCharacter)
- unmap_event_handler_from_callback_id (diguyCrowd)
- unmap_event_handler_from_callback_id (diguyScenario)
- unmap_event_handler_from_callback_id (diguySensorRegion)
- unmap_event_handler_from_callback_id (diguySignal)
- unmap_event_handler_from_callback_id (diguyVariable)
- unregister_character_event_handler (diguyScenario)
- unregister_scenario_event_handler (diguyApp)
- unregister_sensor_region_event_handler (diguyScenario)
- unregister_signal_event_handler (diguyScenario)
- unregister_variable_event_handler (diguyScenario)
- unset_parent (diguyCharacter)
- unset_t_scale_factor (diguyCharacter)
- update (diguyAuthorVisualMultiline)
- update (diguyAuthorVisualObjectHandle)
- update (diguyAuthorVisualRegionMesh)
- update (diguyAuthorVisualRegionPaintbrush)
- update (diguyAuthorVisualWaypoint)
- update (diguyCharacter)
- update (diguyCharacterPath)
- update (diguyGraphicsLink)
- update (diguyGraphicsShape)

- update (diguyNetInterface)
- update (diguyPathShape)
- update (diguyScenario)
- update2 (diguyScenario)
- update_all_members (diguyCharacterGroup)
- update_frustum (diguyViewCamera)
- update_link_matrices (diguyGraphicsShaderInstance)
- update_selected_objects (diguyAuthorInterface)
- update_settings (diguyChainSimulation)
- update_ui (diguyInteractionMachine)
- update_variation_texture (diguyGraphicsShape)
- variable_decrement_value_as_int (diguyCharacter)
- variable_equal_to_float (diguyCharacter)
- variable_equal_to_int (diguyCharacter)
- variable_equal_to_string (diguyCharacter)
- variable_greater_than_float (diguyCharacter)
- variable_greater_than_int (diguyCharacter)
- variable_greater_than_string (diguyCharacter)
- variable_increment_value_as_int (diguyCharacter)
- variable_less_than_float (diguyCharacter)
- variable_less_than_int (diguyCharacter)
- variable_less_than_string (diguyCharacter)
- variable_set_value_as_float (diguyCharacter)
- variable_set_value_as_int (diguyCharacter)
- variable_set_value_as_string (diguyCharacter)
- void
- wait_cursor_pop (diguyScenario)
- wait_cursor_push (diguyScenario)
- world_to_screen (diguyViewPainter)
- y_forward_z_up_to_diguy_coords (diguyVec3f)
- z_forward_y_up_to_diguy_coords (diguyVec3f)

4 Constants List

Listed here are all of the constants and enumerations in the DI-Guy API.

- AUTHOR_UI_AI_INSPECTOR_WINDOW_NAME
- AUTHOR_UI_CHARACTER_WINDOW_NAME
- AUTHOR_UI_CROWD_WINDOW_NAME
- AUTHOR_UI_ELEMENTS_WINDOW_NAME
- AUTHOR_UI_INPUT_MODE_WINDOW_NAME
- AUTHOR_UI_LOG_WINDOW_NAME
- AUTHOR_UI_MAIN_WINDOW_NAME
- AUTHOR_UI_MIND_EDITOR_WINDOW_NAME
- AUTHOR_UI_TIME_CONTROL_WINDOW_NAME
- AUTHOR_UI_VISIBILITY_WINDOW_NAME
- BDI_LOG_ALWAYS
- BDI_LOG_BAD_CALLBACK_HANDLE
- BDI_LOG_DEBUG
- BDI_LOG_DEBUG_LV1
- BDI_LOG_DEBUG_LV2
- BDI_LOG_DEBUG_LV3
- BDI_LOG_DEBUG_LV4
- BDI_LOG_DEBUG_LV5
- BDI_LOG_DEBUG_LV6
- BDI_LOG_DEBUG_LV7
- BDI_LOG_DEBUG_LV8
- BDI_LOG_DEBUG_LV9
- BDI_LOG_ERROR
- BDI_LOG EVERYTHING
- BDI_LOG_FATAL
- BDI_LOG_INFO
- BDI_LOG_MAX_CALLBACKS
- BDI_LOG_NOTHING
- BDI_LOG_WARN
- CHECK_PRINTF_FORMAT
- DIGS_EDIT_ACTION_COUNT
- DIGS_EDIT_ACTION_EDIT_COPY
- DIGS_EDIT_ACTION_EDIT_CUT
- DIGS_EDIT_ACTION_EDIT_DELETE
- DIGS_EDIT_ACTION_EDIT_PASTE
- DIGS_EDIT_ACTION_EDIT_PREFERENCES
- DIGS_EDIT_ACTION_EDIT_UNDO
- DIGS_EDIT_ACTION_FILE_CLOSE
- DIGS_EDIT_ACTION_FILE_EXIT
- DIGS_EDIT_ACTION_FILE_LOAD_REVIEW_DATA
- DIGS_EDIT_ACTION_FILE_MERGE
- DIGS_EDIT_ACTION_FILE_OPEN
- DIGS_EDIT_ACTION_FILE_SAVE
- DIGS_EDIT_ACTION_FILE_SAVE_AS
- DIGS_EDIT_ACTION_FILE_SAVE_REVIEW_DATA
- DIGS_EDIT_ACTION_HELP_ABOUT

- DIGS_EDIT_ACTION_HELP_API_DOCS
- DIGS_EDIT_ACTION_HELP_KEY_MAPPINGS
- DIGS_WINDOW_ACTION_SPREADSHEET
- DIGS_WINDOW_ALL_WINDOWS
- DIGS_WINDOW_CHARACTER_ELEMENTS
- DIGS_WINDOW_COUNT
- DIGS_WINDOW_CROWD_ELEMENTS
- DIGS_WINDOW_DIGUY_AI_ELEMENTS
- DIGS_WINDOW_EVENT_HANDLERS
- DIGS_WINDOW_INPUT_MODE
- DIGS_WINDOW_LOG
- DIGS_WINDOW_MOVIE_SETTINGS
- DIGS_WINDOW_NETWORK_SETTINGS
- DIGS_WINDOW_SCENARIO_ELEMENTS
- DIGS_WINDOW_SECONDARY_3D_VIEW_0
- DIGS_WINDOW_SECONDARY_3D_VIEW_1
- DIGS_WINDOW_SECONDARY_3D_VIEW_2
- DIGS_WINDOW_SECONDARY_3D_VIEW_3
- DIGS_WINDOW_SECONDARY_3D_VIEW_4
- DIGS_WINDOW_SECONDARY_3D_VIEW_5
- DIGS_WINDOW_SECONDARY_3D_VIEW_6
- DIGS_WINDOW_SECONDARY_3D_VIEW_7
- DIGS_WINDOW_SECONDARY_3D_VIEW_8
- DIGS_WINDOW_SIGNAL_PALETTE
- DIGS_WINDOW_TIME_CONTROL
- DIGS_WINDOW_VIEW_SETTINGS
- DIGUY_AGENT_BEHAVIOR_ATTACK
- DIGUY_AGENT_BEHAVIOR_FLEE
- DIGUY_AGENT_BEHAVIOR_IDLE
- DIGUY_AGENT_BEHAVIOR_MINGLE
- DIGUY_AGENT_BEHAVIOR_NONE
- DIGUY_AGENT_BEHAVIOR_NUM_BEHAVIORS
- DIGUY_AGENT_BEHAVIOR_PATH_FOLLOW
- DIGUY_AGENT_BEHAVIOR_PURSUE
- DIGUY_AGENT_BEHAVIOR_TRAVEL
- DIGUY_AGENT_BEHAVIOR_UNKNOWN
- DIGUY_AGENT_BEHAVIOR_WANDER
- DIGUY_AGENT_REPULSION_HIGH
- DIGUY_AGENT_REPULSION_LOW
- DIGUY_AGENT_REPULSION_MEDIUM
- DIGUY_AGENT_REPULSION_NONE
- DIGUY_AGENT_REPULSION_NUM_ZONES
- DIGUY_AGENT_REPULSION_SOLID
- DIGUY_AGENT_UNTRAP_METHOD_GHOST
- DIGUY_AGENT_UNTRAP_METHOD_NAV_PATH
- DIGUY_AGENT_UNTRAP_METHOD_NONE
- DIGUY_AGENT_UNTRAP_METHOD_NUM_METHODS
- DIGUY_AGENT_UNTRAP_METHOD_RANDOM_TURN
- DIGUY_AGENT_UNTRAP_METHOD_STOP_MOVING
- DIGUY_AGENT_UNTRAP_METHOD_TELEPORT
- DIGUY_AIM_ALGORITHM_7

- DIGUY_AIM_ALGORITHM_8
- DIGUY_APPEARANCE_BODY
- DIGUY_APPEARANCE_EXPRESSIVE_HEAD
- DIGUY_APPEARANCE_HAND_ITEM
- DIGUY_APPEARANCE_HEAD
- DIGUY_APPEARANCE_TYPES_COUNT
- DIGUY_CALLBACK_CONTINUE
- DIGUY_CALLBACK_STOP
- DIGUY_CAMERA_MOVE_FORWARD
- DIGUY_CAMERA_MOVE_SIDEWAYS
- DIGUY_CAMERA_MOVE_VERTICAL
- DIGUY_CAMERA_PROJECTION_PERSPECTIVE
- DIGUY_CAMERA_PROJECTION_PLANVIEW_XY
- DIGUY_CAMERA_PROJECTION_PLANVIEW_XZ
- DIGUY_CAMERA_PROJECTION_PLANVIEW_YZ
- DIGUY_CHAIN_CROSS_SECTION_COIL
- DIGUY_CHAIN_CROSS_SECTION_PLUS
- DIGUY_CHAIN_CROSS_SECTION_SQUARE
- DIGUY_CHARACTER_CLASS_ANIMAL
- DIGUY_CHARACTER_CLASS_COUNT
- DIGUY_CHARACTER_CLASS_EFFECT
- DIGUY_CHARACTER_CLASS_HUMAN
- DIGUY_CHARACTER_CLASS_HUMAN_ON_ANIMAL
- DIGUY_CHARACTER_CLASS_HUMAN_ON_OBJECT
- DIGUY_CHARACTER_CLASS_HUMAN_ON_VEHICLE
- DIGUY_CHARACTER_CLASS_OBJECT
- DIGUY_CHARACTER_CLASS_OTHER
- DIGUY_CHARACTER_CLASS_UNKNOWN
- DIGUY_CHARACTER_CLASS_VEHICLE
- DIGUY_CHARACTER_FEELER_B
- DIGUY_CHARACTER_FEELER_BL
- DIGUY_CHARACTER_FEELER_BR
- DIGUY_CHARACTER_FEELER_COUNT
- DIGUY_CHARACTER_FEELER_F
- DIGUY_CHARACTER_FEELER_FL
- DIGUY_CHARACTER_FEELER_FR
- DIGUY_CHARACTER_FEELER_L
- DIGUY_CHARACTER_FEELER_R
- DIGUY_CHARACTER_FEELER_STATUS_CONTACT_COLLIDE
- DIGUY_CHARACTER_FEELER_STATUS_CONTACT_TURN
- DIGUY_CHARACTER_FEELER_STATUS_CONTACT_TURN_AND_COLLIDE
- DIGUY_CHARACTER_FEELER_STATUS_NO_CONTACT
- DIGUY_CHARACTER_FEELER_TURN_DIRECTION_L
- DIGUY_CHARACTER_FEELER_TURN_DIRECTION_L_OR_R
- DIGUY_CHARACTER_FEELER_TURN_DIRECTION_NONE
- DIGUY_CHARACTER_FEELER_TURN_DIRECTION_R
- DIGUY_CHARACTER_GESTURE_TYPE_COUNT
- DIGUY_CHARACTER_GESTURE_TYPE_PROGRESSION
- DIGUY_CHARACTER_GESTURE_TYPE_TABLE
- DIGUY_CHARACTER_GESTURE_TYPE_UNKNOWN
- DIGUY_CHARACTER_MODE_COUNT

- DIGUY_CHARACTER_MODE_FREE
- DIGUY_CHARACTER_MODE_PATH
- DIGUY_CHARACTER_SIMULATOR_DIGUY_CHAIN_SIMULATOR
- DIGUY_CHARACTER_SIMULATOR_DIGUY_MOTION_ENGINE
- DIGUY_CHARACTER_SIMULATOR_DIGUY_VEHICLE_SIMULATOR
- DIGUY_CHARACTER_SIMULATOR_PHYSICS_SIM_MODULE
- DIGUY_CHARACTER_SIMULATOR_UNSET
- DIGUY_CHARACTER_TYPE_MAP_FIELD_AGE
- DIGUY_CHARACTER_TYPE_MAP_FIELD_CARRIED_OBJECT
- DIGUY_CHARACTER_TYPE_MAP_FIELD_CATEGORY
- DIGUY_CHARACTER_TYPE_MAP_FIELD_CHARACTER_CLASS
- DIGUY_CHARACTER_TYPE_MAP_FIELD_COUNT
- DIGUY_CHARACTER_TYPE_MAP_FIELD_COUNTRY
- DIGUY_CHARACTER_TYPE_MAP_FIELD_MILITARY_BRANCH
- DIGUY_CHARACTER_TYPE_MAP_FIELD_RACE
- DIGUY_CHARACTER_TYPE_MAP_FIELD_SEX
- DIGUY_CHARACTER_TYPE_MAP_FIELD_SUBCATEGORY
- DIGUY_CHARACTER_TYPE_MAP_FIELD_UNKNOWN
- DIGUY_CLASSID_CHARACTER
- DIGUY_CLASSID_CREATURE
- DIGUY_CLASSID_LINK
- DIGUY_CLASSID_PATH
- DIGUY_CLASSID_PATH_SHAPE
- DIGUY_CLASSID_SCENE_OBJECT
- DIGUY_CLASSID_SENSOR_REGION
- DIGUY_CLASSID_SHAPE
- DIGUY_CLASSID_START
- DIGUY_COORDINATE_SYSTEM_FLAT_EARTH
- DIGUY_COORDINATE_SYSTEM_GEOCENTRIC
- DIGUY_COORDINATE_SYSTEM_NONE
- DIGUY_COORDINATE_SYSTEM_UTM
- DIGUY_CROWD_PROFILE_SELECTION_METHOD_NUM_METHODS
- DIGUY_CROWD_PROFILE_SELECTION_METHOD_RANDOM
- DIGUY_CROWD_PROFILE_SELECTION_METHOD_SEQUENTIAL
- DIGUY_CUSTOM_PDU_ID_AI_BROADCAST_MESSAGE
- DIGUY_CUSTOM_PDU_ID_AI_BROADCAST_MESSAGE_TO_GROUP
- DIGUY_CUSTOM_PDU_ID_AI_SEND_MESSAGE_TO_CHARACTER
- DIGUY_CUSTOM_PDU_ID_AI_SEND_MESSAGE_TO_GROUP
- DIGUY_CUSTOM_PDU_ID_BEGIN_APPEARANCE_EFFECT
- DIGUY_CUSTOM_PDU_ID_BROADCAST_EXERCISE_DATETIME
- DIGUY_CUSTOM_PDU_ID_CHARACTER_SCALE
- DIGUY_CUSTOM_PDU_ID_COUNT
- DIGUY_CUSTOM_PDU_ID_END_APPEARANCE_EFFECT
- DIGUY_CUSTOM_PDU_ID_ESCAPE_RESERVED
- DIGUY_CUSTOM_PDU_ID_EXECUTE_GESTURE
- DIGUY_CUSTOM_PDU_ID_HAIL
- DIGUY_CUSTOM_PDU_ID_LOAD_SCENARIO_CAMERA
- DIGUY_CUSTOM_PDU_ID_LOAD_SCENARIO_FILE
- DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED0
- DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED1
- DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED2

- DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED3
- DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED4
- DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED5
- DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED6
- DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED7
- DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED8
- DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED9
- DIGUY_CUSTOM_PDU_ID_REQUEST_EXERCISE_DATETIME_BROADCAST
- DIGUY_CUSTOM_PDU_ID_RESET
- DIGUY_CUSTOM_PDU_ID_SAVE_SCENARIO
- DIGUY_CUSTOM_PDU_ID_SAVE_SCENARIO_COMPLETE
- DIGUY_CUSTOM_PDU_ID_SET_CAMERA_DETAILS
- DIGUY_CUSTOM_PDU_ID_SET_CHAR_ACTION
- DIGUY_CUSTOM_PDU_ID_SET_CHAR_APPEARANCE
- DIGUY_CUSTOM_PDU_ID_SET_CHAR_PAUSED
- DIGUY_CUSTOM_PDU_ID_SET_CHAR_TYPE
- DIGUY_CUSTOM_PDU_ID_SET_CHAR_VISIBLE
- DIGUY_CUSTOM_PDU_ID_SET_SCENARIO_FRAME
- DIGUY_CUSTOM_PDU_ID_SET_SCENARIO_PLAYBACK_MODE
- DIGUY_CUSTOM_PDU_ID_SET_SCENE_OBJECT_BIT
- DIGUY_CUSTOM_PDU_ID_SET_SHAPE_VARIATION
- DIGUY_CUSTOM_PDU_ID_STOP_APPEARANCE_EFFECT
- DIGUY_CUSTOM_PDU_ID_VARIABLE_STATE
- DIGUY_CUSTOM_PDU_ID_XC_CONTROL
- DIGUY_CUSTOM_PDU_ID_XC_ENTER_SLAVE_MODE
- DIGUY_CUSTOM_PDU_ID_XC_EVAL_SCRIPT
- DIGUY_CUSTOM_PDU_ID_XC_EXIT_SLAVE_MODE
- DIGUY_CUSTOM_PDU_ID_XC_LOAD_SCENARIO
- DIGUY_CUSTOM_PDU_ID_XC_LOAD_SCENARIO_COMPLETE
- DIGUY_CUSTOM_PDU_ID_XC_MERGE_SCENARIO
- DIGUY_CUSTOM_PDU_ID_XC_MERGE_SCENARIO_COMPLETE
- DIGUY_CUSTOM_PDU_ID_XC_RESET_SCENARIO
- DIGUY_CUSTOM_PDU_ID_XC_SET_FF_SPEED
- DIGUY_CUSTOM_PDU_ID_XC_SET_TIN_TOD
- DIGUY_CUSTOM_PDU_ID_XC_TIME_NOW
- DIGUY_CUSTOM_PDU_ID_XC_TRIGGER_SCRIPT
- DIGUY_CUSTOM_PDU_ID_XC_TRIGGER_SIGNAL
- DIGUY_DATETIME_ADVANCE_METHOD_MANUAL
- DIGUY_DATETIME_ADVANCE_METHOD_REAL_TIME_RATE
- DIGUY_DATETIME_ADVANCE_METHOD_SIMULATION_TIME_RATE
- DIGUY_DATETIME_NETWORK_MODE_BROADCAST
- DIGUY_DATETIME_NETWORK_MODE_LOCAL_ONLY
- DIGUY_DATETIME_NETWORK_MODE_RECEIVE
- DIGUY_DEFAULT_FLOAT
- DIGUY_DEFAULT_INT
- DIGUY_DEG2RAD
- DIGUY_DIS_DAMAGE_DESTROYED
- DIGUY_DIS_DAMAGE_MODERATE
- DIGUY_DIS_DAMAGE_NONE
- DIGUY_DIS_DAMAGE_SLIGHT
- DIGUY_DIS_FLAMES_NONE

- DIGUY_DIS_FLAMES_PRESENT
- DIGUY_DIS_FOCUS_TARGET
- DIGUY_DIS_LIFEFORM_CRAWLING
- DIGUY_DIS_LIFEFORM_CROUCHING
- DIGUY_DIS_LIFEFORM_JUMPING
- DIGUY_DIS_LIFEFORM_KNEELING
- DIGUY_DIS_LIFEFORM_PARACHUTING
- DIGUY_DIS_LIFEFORM_PRONE
- DIGUY_DIS_LIFEFORM_SITTING
- DIGUY_DIS_LIFEFORM_SQUATTING
- DIGUY_DIS_LIFEFORM_SWIMMING
- DIGUY_DIS_LIFEFORM_UNUSED
- DIGUY_DIS_LIFEFORM_UPRIGHT_RUNNING
- DIGUY_DIS_LIFEFORM_UPRIGHT_STANDING_STILL
- DIGUY_DIS_LIFEFORM_UPRIGHT_WALKING
- DIGUY_DIS_LIFEFORM_WADING
- DIGUY_DIS_MISSILE_TRAIL_OFF
- DIGUY_DIS_MISSILE_TRAIL_ON
- DIGUY_DIS_POWER_PLANT_OFF
- DIGUY_DIS_POWER_PLANT_ON
- DIGUY_DIS_PRIMARY_WEAPON_DEPLOYED
- DIGUY_DIS_PRIMARY_WEAPON_IN_FIRING_POSITION
- DIGUY_DIS_PRIMARY_WEAPON_NOT_PRESENT
- DIGUY_DIS_PRIMARY_WEAPON_STOWED
- DIGUY_DIS_SECONDARY_WEAPON_DEPLOYED
- DIGUY_DIS_SECONDARY_WEAPON_IN_FIRING_POSITION
- DIGUY_DIS_SECONDARY_WEAPON_NOT_PRESENT
- DIGUY_DIS_SECONDARY_WEAPON_STOWED
- DIGUY_DIS_SMOKE_ENGINE_SMOKE
- DIGUY_DIS_SMOKE_ENGINE_SMOKE_AND_SMOKE_PLUME
- DIGUY_DIS_SMOKE_NOT_SMOKING
- DIGUY_DIS_SMOKE_SMOKE_PLUME
- DIGUY_DIS_TRAILING_EFFECT_LARGE
- DIGUY_DIS_TRAILING_EFFECT_MEDIUM
- DIGUY_DIS_TRAILING_EFFECT_NONE
- DIGUY_DIS_TRAILING_EFFECT_SMALL
- DIGUY_DYNAMIC_AVOIDANCE_METHOD_DISABLED
- DIGUY_DYNAMIC_AVOIDANCE_METHOD_REPULSION_ZONES
- DIGUY_DYNAMIC_AVOIDANCE_METHOD_SIMPLE
- DIGUY_ERROR
- DIGUY_ERROR_DATA_INSTALLATION_MISSING
- DIGUY_ERROR_DATA_VERSION_TOO_LOW
- DIGUY_ERROR_GRAPHICS_SUBSYSTEM_INIT_FAILED
- DIGUY_ERROR_LICENSE_COUNT_TOO_LOW
- DIGUY_ERROR_LICENSE_EXPIRED
- DIGUY_ERROR_LICENSE_FAILURE
- DIGUY_ERROR_LICENSE_FILE_NOT_FOUND
- DIGUY_ERROR_LICENSE_NOT_FOUND
- DIGUY_ERROR_LICENSE_NO_DONGLE
- DIGUY_ERROR_LICENSE_NO_DONGLE_DRIVER
- DIGUY_ERROR_LICENSE_SERVER_ERROR

- DIGUY_ERROR_LICENSE_SOFTWARE_INITIALIZATION_ERROR
- DIGUY_ERROR_LICENSE_UNKNOWN_ERROR
- DIGUY_ERROR_LICENSE_VERSION_MISMATCH
- DIGUY_ERROR_RENDERER_LICENSE_CHECK_FAILED
- DIGUY_ERROR_SIMULATION_SUBSYSTEM_INIT_FAILED
- DIGUY_EULER_ANGLE_ORDER_XYZ
- DIGUY_EULER_ANGLE_ORDER_XZY
- DIGUY_EULER_ANGLE_ORDER_YXZ
- DIGUY_EULER_ANGLE_ORDER_YZX
- DIGUY_EULER_ANGLE_ORDER_ZXY
- DIGUY_EULER_ANGLE_ORDER_ZYX
- DIGUY_EULER_ANGLE_ORDER_ZYZ
- DIGUY_FOG_MODE_DISABLED
- DIGUY_FOG_MODE_EXP
- DIGUY_FOG_MODE_EXP2
- DIGUY_FOG_MODE_LINEAR
- DIGUY_FOG_MODE_PIX_EXP
- DIGUY_FOG_MODE_PIX_EXP2
- DIGUY_FOG_MODE_PIX_LINEAR
- DIGUY_GEOMETRY_TAG_DOOR
- DIGUY_GEOMETRY_TAG_INSIDE
- DIGUY_GEOMETRY_TAG_NONE
- DIGUY_GEOMETRY_TAG_OUTSIDE
- DIGUY_GEOMETRY_TAG_ROOF
- DIGUY_GRAPHICS_INDEX_TYPE_D3D_WORD
- DIGUY_GRAPHICS_INDEX_TYPE_UINT
- DIGUY_GRAPHICS_INDEX_TYPE_UNKNOWN
- DIGUY_GRAPHICS_INDEX_TYPE_USHORT
- DIGUY_GRAPHICS_JOINT_TYPE_6DOF
- DIGUY_GRAPHICS_JOINT_TYPE_BALL
- DIGUY_GRAPHICS_JOINT_TYPE_COUNT
- DIGUY_GRAPHICS_JOINT_TYPE_CYLINDER
- DIGUY_GRAPHICS_JOINT_TYPE_FROZEN
- DIGUY_GRAPHICS_JOINT_TYPE_GIMBAL
- DIGUY_GRAPHICS_JOINT_TYPE_PIN
- DIGUY_GRAPHICS_JOINT_TYPE_PLANAR
- DIGUY_GRAPHICS_JOINT_TYPE_SLIDER
- DIGUY_GRAPHICS_JOINT_TYPE_TRANSLATION_AND_GIMBAL
- DIGUY_GRAPHICS_JOINT_TYPE_TRANSLATION_AND_PIN
- DIGUY_GRAPHICS_JOINT_TYPE_UNIVERSAL
- DIGUY_GRAPHICS_JOINT_TYPE_UNKNOWN
- DIGUY_GRAPHICS_LABEL_JUSTIFIED_BOTTOM
- DIGUY_GRAPHICS_LABEL_JUSTIFIED_CENTER
- DIGUY_GRAPHICS_LABEL_JUSTIFIED_LEFT
- DIGUY_GRAPHICS_LABEL_JUSTIFIED_RIGHT
- DIGUY_GRAPHICS_LABEL_JUSTIFIED_TOP
- DIGUY_GRAPHICS_SHADER_LANGUAGE_GLSL
- DIGUY_GRAPHICS_SHADER_LANGUAGE_HLSL
- DIGUY_GRAPHICS_SHADER_PROGRAM_TYPE_GENERAL
- DIGUY_GRAPHICS_SHADER_PROGRAM_TYPE_REFLECTOR
- DIGUY_GRAPHICS_SHADER_PROGRAM_TYPE_SKIN

- DIGUY_GRAPHICS_SHADER_PROGRAM_TYPE_UNKNOWN
- DIGUY_GRAPHICS_SHADER_QUALITY_MAX
- DIGUY_GRAPHICS_SHADER_QUALITY_PER_PIXEL
- DIGUY_GRAPHICS_SHADER_QUALITY_PER_PIXEL_BUMP_MAPPED
- DIGUY_GRAPHICS_SHADER_QUALITY_PER_VERTEX
- DIGUY_GRAPHICS_SHADER_QUALITY_UNKNOWN
- DIGUY_GRAPHICS_SHADER_QUALITY_UNLIT
- DIGUY_GRAPHICS_TEXTURE_COMPRESSION_TYPE_DXT1
- DIGUY_GRAPHICS_TEXTURE_COMPRESSION_TYPE_DXT2
- DIGUY_GRAPHICS_TEXTURE_COMPRESSION_TYPE_DXT3
- DIGUY_GRAPHICS_TEXTURE_COMPRESSION_TYPE_DXT4
- DIGUY_GRAPHICS_TEXTURE_COMPRESSION_TYPE_DXT5
- DIGUY_GRAPHICS_TEXTURE_COMPRESSION_TYPE_DXT5_NM
- DIGUY_GRAPHICS_TEXTURE_COMPRESSION_TYPE_NONE
- DIGUY_GRAPHICS_VERTEX_FORMAT_UNKNOWN
- DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2
- DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_ARRAYS
- DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_I4F_W4
- DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_I4F_W4_ARRAYS
- DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_T4
- DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_T4_I4F_W4
- DIGUY_GRAPHICS_VISUAL_UPDATE_ALL
- DIGUY_GRAPHICS_VISUAL_UPDATE_GEOMETRY
- DIGUY_GRAPHICS_VISUAL_UPDATE_MATERIAL
- DIGUY_GRAPHICS_VISUAL_UPDATE_NONE
- DIGUY_GRAPHICS_VISUAL_UPDATE_POSITION
- DIGUY_HISTORY_TYPE_COMPLETE
- DIGUY_HISTORY_TYPE_FIRST
- DIGUY_HISTORY_TYPE_LAST
- DIGUY_HISTORY_TYPE_NONE
- DIGUY_HISTORY_TYPE_PARTIAL
- DIGUY_IMPACT_DIRECT_HIT
- DIGUY_IMPACT_NEARBY_DETONATION
- DIGUY_IMPACT_NO_IMPACT
- DIGUY_IMPACT_USER_GENERATED
- DIGUY_IMPACT_VEHICLE_COLLISION
- DIGUY_LOAD_ASYNC
- DIGUY_LOAD_INSTANTLY
- DIGUY_LOAD_MANAGER_NUM_NONVISIBLE_ZONE_INDEX
- DIGUY_LOAD_MANAGER_NUM_ZONES
- DIGUY_LOAD_PRIORITY_ASYNC
- DIGUY_MERGE_FLAG_DONT_MERGE
- DIGUY_MERGE_FLAG_KEEP_BOTH
- DIGUY_MERGE_FLAG_KEEP_EXISTING_ONLY
- DIGUY_MERGE_FLAG_KEEP_EXISTING_ONLY_AND_WARN
- DIGUY_MERGE_FLAG_OVERWRITE_EXISTING
- DIGUY_MERGE_FLAG_RENAME_MERGED
- DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN
- DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL
- DIGUY_MERGE_INITIAL_SETTINGS_DEFAULT
- DIGUY_MERGE_INITIAL_SETTINGS_DIGUY6_COMPATIBLE

- DIGUY_MERGE_INITIAL_SETTINGS_MERGE_EVERYTHING
- DIGUY_MERGE_INITIAL_SETTINGS_MERGE_NOTHING
- DIGUY_MERGE_INITIAL_SETTINGS_PASTE
- DIGUY_MERGE_RESET_ALL
- DIGUY_MERGE_RESET_MERGED
- DIGUY_MERGE_RESET_NONE
- DIGUY_MOTION_DIRECTION_BACKWARD
- DIGUY_MOTION_DIRECTION_COUNT
- DIGUY_MOTION_DIRECTION_DOWN
- DIGUY_MOTION_DIRECTION_FORWARD
- DIGUY_MOTION_DIRECTION_INHERIT
- DIGUY_MOTION_DIRECTION_LEFT
- DIGUY_MOTION_DIRECTION_LINEAR_OFFAXIS
- DIGUY_MOTION_DIRECTION_NONE
- DIGUY_MOTION_DIRECTION_NONLINEAR
- DIGUY_MOTION_DIRECTION_RIGHT
- DIGUY_MOTION_DIRECTION_TURN_LEFT
- DIGUY_MOTION_DIRECTION_TURN_RIGHT
- DIGUY_MOTION_DIRECTION_UNKNOWN
- DIGUY_MOTION_DIRECTION_UP
- DIGUY_MOTION_POSTURE_ANY
- DIGUY_MOTION_POSTURE_COUNT
- DIGUY_MOTION_POSTURE_CROUCHED
- DIGUY_MOTION_POSTURE_PRONE
- DIGUY_MOTION_POSTURE_SITTING
- DIGUY_MOTION_POSTURE_UNKNOWN
- DIGUY_MOTION_POSTURE_UPRIGHT
- DIGUY_MOTION_SPEED_TYPE_COUNT
- DIGUY_MOTION_SPEED_TYPE_FAST
- DIGUY_MOTION_SPEED_TYPE_FASTEST
- DIGUY_MOTION_SPEED_TYPE_MEDIUM
- DIGUY_MOTION_SPEED_TYPE_SLOW
- DIGUY_MOTION_SPEED_TYPE_STILL
- DIGUY_MOTION_SPEED_TYPE_UNKNOWN
- DIGUY_MOTION_VARIANT_AIM
- DIGUY_MOTION_VARIANT_AMBIENT
- DIGUY_MOTION_VARIANT_ANGRY
- DIGUY_MOTION_VARIANT_ANY
- DIGUY_MOTION_VARIANT_ASSAULT
- DIGUY_MOTION_VARIANT_COUNT
- DIGUY_MOTION_VARIANT_CURIOUS
- DIGUY_MOTION_VARIANT_DEAD
- DIGUY_MOTION_VARIANT_EXERCISE
- DIGUY_MOTION_VARIANT_FUNKY
- DIGUY_MOTION_VARIANT_INJURED
- DIGUY_MOTION_VARIANT_MISC0
- DIGUY_MOTION_VARIANT_MISC1
- DIGUY_MOTION_VARIANT_MISC2
- DIGUY_MOTION_VARIANT_MISC3
- DIGUY_MOTION_VARIANT_MISC4
- DIGUY_MOTION_VARIANT_MISC5

- DIGUY_MOTION_VARIANT_MISC6
- DIGUY_MOTION_VARIANT_MISC7
- DIGUY_MOTION_VARIANT_MISC8
- DIGUY_MOTION_VARIANT_MISC9
- DIGUY_MOTION_VARIANT_NORMAL
- DIGUY_MOTION_VARIANT_OTHER
- DIGUY_MOTION_VARIANT_READY
- DIGUY_MOTION_VARIANT_SICK
- DIGUY_MOTION_VARIANT_SOCIALIZE
- DIGUY_MOTION_VARIANT_UNKNOWN
- DIGUY_MUZZLE_ACCESSORY1
- DIGUY_MUZZLE_COUNT
- DIGUY_MUZZLE_DEFAULT
- DIGUY_MUZZLE_GRENADE_LAUNCHER
- DIGUY_MUZZLE_RIFLE
- DIGUY_MUZZLE_TURRET1
- DIGUY_NAVMESH_LEAST_VISIBLE
- DIGUY_NAVMESH_MESH_AVOID_EDGES
- DIGUY_NAVMESH_MESH_EDGE
- DIGUY_NAVMESH_MESH_EDGE_AND_LEAST_VIS
- DIGUY_NAVMESH_SHORTEST_PATH
- DIGUY_NAV_PATH_AT_END_COUNT
- DIGUY_NAV_PATH_AT_END_LOOP
- DIGUY_NAV_PATH_AT_END_REVERSE
- DIGUY_NAV_PATH_AT_END_STOP
- DIGUY_NAV_PATH_AT_END_SWITCH_BEHAVIOR
- DIGUY_NAV_PATH_AT_END_TELEPORT
- DIGUY_NAV_PATH_AT_END_UNKNOWN
- DIGUY_NAV_PATH_DIRECTION_BACKWARD
- DIGUY_NAV_PATH_DIRECTION_COUNT
- DIGUY_NAV_PATH_DIRECTION_FORWARD
- DIGUY_NAV_PATH_DIRECTION_UNKNOWN
- DIGUY_NAV_PATH_ERROR_FAILED_TO_FIND_PATH
- DIGUY_NAV_PATH_ERROR_INVALID_CANVAS_OR_REGION
- DIGUY_NAV_PATH_ERROR_INVALID_END_POINT
- DIGUY_NAV_PATH_ERROR_INVALID_LENGTH
- DIGUY_NAV_PATH_ERROR_INVALID_START_POINT
- DIGUY_NAV_PATH_ERROR_NONE
- DIGUY_NAV_PATH_ERROR_OUT_OF_TIME
- DIGUY_NAV_PATH_ERROR_SEARCH_QUEUED
- DIGUY_NAV_PATH_ERROR_UNKNOWN
- DIGUY_PARAMETER_COMPARISON_COUNT
- DIGUY_PARAMETER_COMPARISON_EQUAL
- DIGUY_PARAMETER_COMPARISON_GREATER
- DIGUY_PARAMETER_COMPARISON_LESS
- DIGUY_PARAMETER_COMPARISON_NEAREST
- DIGUY_PHYSICS_COLLISION_GROUP_CHARACTER0
- DIGUY_PHYSICS_COLLISION_GROUP_CHARACTER1
- DIGUY_PHYSICS_COLLISION_GROUP_CHARACTER2
- DIGUY_PHYSICS_COLLISION_GROUP_CHARACTER3
- DIGUY_PHYSICS_COLLISION_GROUP_CHARACTER_EVERYTHING

- DIGUY_PHYSICS_COLLISION_GROUP_CHARACTER_GHOST
- DIGUY_PHYSICS_COLLISION_GROUP_COUNT
- DIGUY_PHYSICS_COLLISION_GROUP_EFFECT0
- DIGUY_PHYSICS_COLLISION_GROUP_PROP0
- DIGUY_PHYSICS_COLLISION_GROUP_PROP1
- DIGUY_PHYSICS_COLLISION_GROUP_PROP2
- DIGUY_PHYSICS_COLLISION_GROUP_PROP3
- DIGUY_PHYSICS_COLLISION_GROUP_SCENE_OBJECT0
- DIGUY_PHYSICS_COLLISION_GROUP_SCENE_OBJECT1
- DIGUY_PHYSICS_COLLISION_GROUP_SCENE_OBJECT2
- DIGUY_PHYSICS_COLLISION_GROUP_SCENE_OBJECT3
- DIGUY_PHYSICS_SIMULATION_MODE_CHARACTER_DRIVES_SIM
- DIGUY_PHYSICS_SIMULATION_MODE_CHARACTER_DRIVES_SIM_GOAL_DATA
- DIGUY_PHYSICS_SIMULATION_MODE_SIM_DRIVES_CHARACTER
- DIGUY_PHYSICS_SIMULATION_MODE_UNSET
- DIGUY_PI
- DIGUY_PLUGIN_MOUSE_CURSOR_CROSSHAIR
- DIGUY_PLUGIN_MOUSE_CURSOR_FORBIDDEN
- DIGUY_PLUGIN_MOUSE_CURSOR_LEFT_ARROW
- DIGUY_PLUGIN_MOUSE_CURSOR_NONE
- DIGUY_PLUGIN_MOUSE_CURSOR_POINTING_HAND
- DIGUY_RAD2DEG
- DIGUY_REGION_PAINTBRUSH_INTERNAL_MODE_REPLACE
- DIGUY_REGION_PAINTBRUSH_INTERNAL_MODE_UPDATE_BEHAVIOR_PATH
- DIGUY_REGION_PAINTBRUSH_MODE_ADDITIVE
- DIGUY_REGION_PAINTBRUSH_MODE_ERASE
- DIGUY_REGION_PAINTBRUSH_MODE_MOVE_MESH
- DIGUY_REGION_PAINTBRUSH_MODE_REACHABLE
- DIGUY_REGION_PAINTBRUSH_MODE_UGROUND_CLAMP
- DIGUY_REGION_PAINTBRUSH_NUM_MODES
- DIGUY_RENDER_PASS_INTERSECTION
- DIGUY_RENDER_PASS_REFLECTION
- DIGUY_RENDER_PASS_SHADOW
- DIGUY_RENDER_PASS_SKIP_PARTICLES
- DIGUY_RENDER_PASS_USE_STORED_LOD_SETTINGS
- DIGUY_REQUIRED_DATA_VERSION_MAJOR
- DIGUY_REQUIRED_DATA_VERSION_MINOR
- DIGUY_REQUIRED_DATA_VERSION_POINT
- DIGUY_SCENARIO_INPUT_MODE_BEAD_CREATE
- DIGUY_SCENARIO_INPUT_MODE_BEAD_MOVE
- DIGUY_SCENARIO_INPUT_MODE_CAMERA
- DIGUY_SCENARIO_INPUT_MODE_CHARACTER_MOVE
- DIGUY_SCENARIO_INPUT_MODE_COUNT
- DIGUY_SCENARIO_INPUT_MODE_CROWD_BLITZER
- DIGUY_SCENARIO_INPUT_MODE_CROWD_EDITOR
- DIGUY_SCENARIO_INPUT_MODE_GROUP_MOVE
- DIGUY_SCENARIO_INPUT_MODE_IGUY
- DIGUY_SCENARIO_INPUT_MODE_NAVMESH_PAINTER
- DIGUY_SCENARIO_INPUT_MODE_NONE
- DIGUY_SCENARIO_INPUT_MODE_PEOPLE_BLITZER
- DIGUY_SCENARIO_INPUT_MODE_PLUGIN

- DIGUY_SCENARIO_INPUT_MODE_POINT_SELECT
- DIGUY_SCENARIO_INPUT_MODE_REGION_PAINTER
- DIGUY_SCENARIO_INPUT_MODE_SENSOR_REGION_CREATE
- DIGUY_SCENARIO_INPUT_MODE_SENSOR_REGION_MOVE
- DIGUY_SCENARIO_INPUT_MODE_WAYPOINT_CREATE
- DIGUY_SCENARIO_INPUT_MODE_WAYPOINT_MOVE
- DIGUY_SCENARIO_PLAYBACK_MODE_COUNT
- DIGUY_SCENARIO_PLAYBACK_MODE_FF
- DIGUY_SCENARIO_PLAYBACK_MODE_FF_REVERSE
- DIGUY_SCENARIO_PLAYBACK_MODE_PLAY
- DIGUY_SCENARIO_PLAYBACK_MODE_PLAY_REVERSE
- DIGUY_SCENARIO_PLAYBACK_MODE_REWIND
- DIGUY_SCENARIO_PLAYBACK_MODE_SS
- DIGUY_SCENARIO_PLAYBACK_MODE_SS_REVERSE
- DIGUY_SCENARIO_PLAYBACK_MODE_STOP
- DIGUY_SPECIAL_F1
- DIGUY_SPECIAL_F10
- DIGUY_SPECIAL_F11
- DIGUY_SPECIAL_F12
- DIGUY_SPECIAL_F2
- DIGUY_SPECIAL_F3
- DIGUY_SPECIAL_F4
- DIGUY_SPECIAL_F5
- DIGUY_SPECIAL_F6
- DIGUY_SPECIAL_F7
- DIGUY_SPECIAL_F8
- DIGUY_SPECIAL_F9
- DIGUY_SPECIAL_KEY_BACKSPACE
- DIGUY_SPECIAL_KEY_BACKTAB
- DIGUY_SPECIAL_KEY_DELETE
- DIGUY_SPECIAL_KEY_DOWN
- DIGUY_SPECIAL_KEY_END
- DIGUY_SPECIAL_KEY_ENTER
- DIGUY_SPECIAL_KEY_ESC
- DIGUY_SPECIAL_KEY_HOME
- DIGUY_SPECIAL_KEY_INSERT
- DIGUY_SPECIAL_KEY_LEFT
- DIGUY_SPECIAL_KEY_MAX
- DIGUY_SPECIAL_KEY_MIN
- DIGUY_SPECIAL_KEY_PAGE_DOWN
- DIGUY_SPECIAL_KEY_PAGE_UP
- DIGUY_SPECIAL_KEY_PAUSE
- DIGUY_SPECIAL_KEY_PRINT
- DIGUY_SPECIAL_KEY_RETURN
- DIGUY_SPECIAL_KEY_RIGHT
- DIGUY_SPECIAL_KEY_TAB
- DIGUY_SPECIAL_KEY_UP
- DIGUY_SPEED_UNITS_KILOMETERS_PER_HOUR
- DIGUY_SPEED_UNITS_METERS_PER_SECOND
- DIGUY_SPEED_UNITS_MILES_PER_HOUR
- DIGUY_STATIC_AVOIDANCE_METHOD_DISABLED

- DIGUY_STATIC_AVOIDANCE_METHOD_FEELERS
- DIGUY_SUBREGION_BASE
- DIGUY_SUBREGION_BLUE
- DIGUY_SUBREGION_CROSSWALK
- DIGUY_SUBREGION_GREEN
- DIGUY_SUBREGION_MASK_BASE
- DIGUY_SUBREGION_MASK_BLUE
- DIGUY_SUBREGION_MASK_CROSSWALK
- DIGUY_SUBREGION_MASK_GREEN
- DIGUY_SUBREGION_MASK_NONE
- DIGUY_SUBREGION_MASK_ORANGE
- DIGUY_SUBREGION_MASK_POPULATE
- DIGUY_SUBREGION_MASK_RED
- DIGUY_SUBREGION_MASK_ROAD
- DIGUY_SUBREGION_MASK_SIDEWALK
- DIGUY_SUBREGION_MASK_WHITE
- DIGUY_SUBREGION_MASK_YELLOW
- DIGUY_SUBREGION_NUM_SUBREGIONS
- DIGUY_SUBREGION_ORANGE
- DIGUY_SUBREGION_POPULATE
- DIGUY_SUBREGION_RED
- DIGUY_SUBREGION_ROAD
- DIGUY_SUBREGION_SIDEWALK
- DIGUY_SUBREGION_WHITE
- DIGUY_SUBREGION_YELLOW
- DIGUY_TACTICS_UNIT_TYPE_ARMY
- DIGUY_TACTICS_UNIT_TYPE_ARMY_BATTALION
- DIGUY_TACTICS_UNIT_TYPE_ARMY_BRIGADE
- DIGUY_TACTICS_UNIT_TYPE_ARMY_COMPANY
- DIGUY_TACTICS_UNIT_TYPE_ARMY_CORPS
- DIGUY_TACTICS_UNIT_TYPE_ARMY_DIVISION
- DIGUY_TACTICS_UNIT_TYPE_ARMY_PLATOON
- DIGUY_TACTICS_UNIT_TYPE_ARMY_SOLDIER
- DIGUY_TACTICS_UNIT_TYPE_ARMY_SQUAD
- DIGUY_TACTICS_UNIT_TYPE_MARINE
- DIGUY_TACTICS_UNIT_TYPE_MARINE_BATTALION
- DIGUY_TACTICS_UNIT_TYPE_MARINE_COMPANY
- DIGUY_TACTICS_UNIT_TYPE_MARINE_DIVISION
- DIGUY_TACTICS_UNIT_TYPE_MARINE_FIRETEAM
- DIGUY_TACTICS_UNIT_TYPE_MARINE_MARINE_CORPS
- DIGUY_TACTICS_UNIT_TYPE_MARINE_PLATOON
- DIGUY_TACTICS_UNIT_TYPE_MARINE_REGIMENT
- DIGUY_TACTICS_UNIT_TYPE_MARINE_SQUAD
- DIGUY_TACTICS_UNIT_TYPE_UNKNOWN
- DIGUY_TEXTURE_MAP_MASK_ALL
- DIGUY_TEXTURE_MAP_MASK_BUMP
- DIGUY_TEXTURE_MAP_MASK_COMPONENTS
- DIGUY_TEXTURE_MAP_MASK_DIFFUSE
- DIGUY_TEXTURE_MAP_MASK_MISC_1
- DIGUY_TEXTURE_MAP_MASK_MISC_2
- DIGUY_TEXTURE_MAP_MASK_SHADOW

- DIGUY_TEXTURE_MAP_MASK_SPECULAR
- DIGUY_TEXTURE_MAP_TYPE_BUMP
- DIGUY_TEXTURE_MAP_TYPE_COMPONENTS
- DIGUY_TEXTURE_MAP_TYPE_COMPONENT_RAMP
- DIGUY_TEXTURE_MAP_TYPE_DIFFUSE
- DIGUY_TEXTURE_MAP_TYPE_MISC_1
- DIGUY_TEXTURE_MAP_TYPE_MISC_2
- DIGUY_TEXTURE_MAP_TYPE_SHADOW
- DIGUY_TEXTURE_MAP_TYPE_SPECULAR
- DIGUY_TEXTURE_MAP_TYPE_UNKNOWN
- DIGUY_TYPE_UID_BASE_CHARACTER
- DIGUY_TYPE_UID_BASE_CHARACTER_BEAD
- DIGUY_TYPE_UID_BASE_CHARACTER_LINK
- DIGUY_TYPE_UID_BASE_CHARACTER_PATH
- DIGUY_TYPE_UID_BASE_CHARACTER_PATH_WAYPOINT
- DIGUY_TYPE_UID_BASE_CHARACTER_SHAPE
- DIGUY_TYPE_UID_BASE_COUNT
- DIGUY_TYPE_UID_BASE_CROWD
- DIGUY_TYPE_UID_BASE_LABEL
- DIGUY_TYPE_UID_BASE_NOTHING
- DIGUY_TYPE_UID_BASE_OTHER
- DIGUY_TYPE_UID_BASE_PATH_SHAPE
- DIGUY_TYPE_UID_BASE_PATH_SHAPE_WAYPOINT
- DIGUY_TYPE_UID_BASE_REGION
- DIGUY_TYPE_UID_BASE_SCENE_OBJECT
- DIGUY_TYPE_UID_BASE_SENSOR_REGION
- DIGUY_TYPE_UID_BASE_USER
- DIGUY_VEC_QW
- DIGUY_VEC_QX
- DIGUY_VEC_QY
- DIGUY_VEC_QZ
- DIGUY_VEC_RX
- DIGUY_VEC_RY
- DIGUY_VEC_RZ
- DIGUY_VEC_W
- DIGUY_VEC_X
- DIGUY_VEC_Y
- DIGUY_VEC_Z
- DIGUY_VEHICLE_SIMULATION_MODE_DEFAULT
- DIGUY_VEHICLE_SIMULATION_MODE_FULL_PHYSICS
- DIGUY_VEHICLE_SIMULATION_MODE_NONE
- DIGUY_VEHICLE_SIMULATION_MODE_REAR_AXLE_FOLLOWING
- DIGUY_VEHICLE_SIMULATION_MODE_STEERING
- DIGUY_VERSION_MAJOR
- DIGUY_VERSION_MINOR
- DIGUY_VERSION_POINT
- DIGUY_VERSION_STRING
- DIGUY_VISIBLE_ALL
- DIGUY_VISIBLE_CURRENT
- DIGUY_VISIBLE_NONE
- DIGUY_VISIBLE_PER_OBJECT

5 Types Reference

5.1 diguyAgentParams

diguyAgentParams.h

DI-Guy SDK Version 12.5.2

Includes: [diguyMotionPosture.h](#) | [diguyMotionSpeedType.h](#) | [diguyMotionVariant.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Notes](#)
- ◆ [Behavior Functions](#)
- ◆ [Attack Behavior Functions](#)
- ◆ [Wander Behavior Functions](#)
- ◆ [Pursue Speed Match Functions](#)
- ◆ [Path Follow Behavior Functions](#)
- ◆ [Trap Functions](#)
- ◆ [Speed Type Parameters](#)
- ◆ [Dynamic Avoidance Parameters](#)
- ◆ [Vehicle-Specific Avoidance Parameters](#)
- ◆ [Feeler Functions](#)
- ◆ [Path Planner Parameter Functions](#)
- ◆ [Experimental Functions](#)

5.1.1 class diguyAgentParams

```
class BDI_DECLSPEC_diguy diguyAgentParams
{
public:
    void set_posture(diguyMotionPosture posture);
    diguyMotionPosture get_posture();
    void set_variant(diguyMotionVariant variant);
    diguyMotionVariant get_variant();
    void set_behavior(diguyAgentBehavior behavior);
    diguyAgentBehavior get_behavior();
    void set_behavior_path_shape(const char* behavior_path_shape_name);
    const char* get_behavior_path_shape();
    void set_behavior_region(const char* behavior_region_name);
    const char* get_behavior_region();
    void set_behavior_subregion(diguySubregionIndex area);
    diguySubregionIndex get_behavior_subregion();
    void set_behavior_region_border_is_solid(int is_solid);
    int get_behavior_region_border_is_solid();
    void set_focus_character(const char* focus_character_name);
    const char* get_focus_character();
    void set_focus_group(const char* focus_group_name);
    const char* get_focus_group();
    void set_focus_affects_gaze(int focus_affects_gaze);
    int get_focus_affects_gaze();
    void set_max_focus_distance(float distance);
    float get_max_focus_distance();
    void set_move_on_t_min(float t_min);
```

```

float get_move_on_t_min();
void set_move_on_t_max(float t_max);
float get_move_on_t_max();
void set_attack_distance(float distance);
float get_attack_distance();
void set_attack_fire_interval(float time_interval);
float get_attack_fire_interval();
void set_attack_stands_ground(int value);
int get_attack_stands_ground();
void set_attack_zone(float x, float y, float z, float radius);
int get_attack_zone(float* x, float* y, float* z, float* radius);
void set_attack_zone_enabled(int use_attack_zone);
int get_attack_zone_enabled();
void set_wander_max_wander_dist(float distance);
void set_wander_sets_desired_orientation(int sets_desired_orientation);
void set_pursue_speed_match_enabled(int enabled);
int get_pursue_speed_match_enabled();
void set_pursue_speed_match_update_dt(float dt);
float get_pursue_speed_match_update_dt();
void set_pursue_speed_match_max_factor(float factor);
float get_pursue_speed_match_max_factor();
void set_pursue_offset(float x, float y, float z);
int get_pursue_offset(float* x, float* y, float* z);
void set_path_follow_pause_t_when_off_path(int pause_t);
int get_path_follow_pause_t_when_off_path();
void set_path_follow_unpause_t_at_radius(float radius);
float get_path_follow_unpause_t_at_radius();
void set_path_follow_position_lookahead_distance(float distance);
float get_path_follow_position_lookahead_distance();
void set_wedged_untrap_method(diguyAgentUntrapMethod untrap_method);
diguyAgentUntrapMethod get_wedged_untrap_method();
void set_wedged_trap_time_threshold(float threshold);
float get_wedged_trap_time_threshold();
void set_first_no_progress_untrap_method(diguyAgentUntrapMethod untrap_method);
diguyAgentUntrapMethod get_first_no_progress_untrap_method();
void set_first_no_progress_trap_time_threshold(float threshold);
float get_first_no_progress_trap_time_threshold();
void set_second_no_progress_untrap_method(diguyAgentUntrapMethod untrap_method);
diguyAgentUntrapMethod get_second_no_progress_untrap_method();
void set_second_no_progress_trap_time_threshold(float threshold);
float get_second_no_progress_trap_time_threshold();
void set_untrap_method_random_turn_max_tries(int max_tries);
int get_untrap_method_random_turn_max_tries();
void set_untrap_method_ghost_duration(float duration);
float get_untrap_method_ghost_duration();
int set_speed_type_enabled(diguyMotionSpeedType motion_speed_type,
    int enabled);
int get_speed_type_enabled(diguyMotionSpeedType motion_speed_type);
int set_speed_type_radius(diguyMotionSpeedType motion_speed_type,
    float radius);
float get_speed_type_radius(diguyMotionSpeedType motion_speed_type);
void set_speed_type_radius_hysteresis(float hysteresis_distance);
float get_speed_type_radius_hysteresis();
int set_speed_type_exact_action(diguyMotionSpeedType motion_speed_type,
    const char* action);
const char* get_speed_type_exact_action(diguyMotionSpeedType motion_speed_type);
void set_radius_variation(float distance);
float get_radius_variation();
int set_turning_motions_enabled(int enabled);
int get_turning_motions_enabled();
void set_repulsion_radius(diguyAgentRepulsionZone repulsion_zone,

```



```

    float radius);
float get_repulsion_radius(diguyAgentRepulsionZone repulsion_zone);
void set_repulsion_factor(diguyAgentRepulsionZone repulsion_zone,
    float factor);
float get_repulsion_factor(diguyAgentRepulsionZone repulsion_zone);
void set_repulsion_zone_enabled(diguyAgentRepulsionZone repulsion_zone,
    int enabled);
int get_repulsion_zone_enabled(diguyAgentRepulsionZone repulsion_zone);
void set_max_orientation_change_rate(float max_orientation_change_rate);
float get_max_orientation_change_rate();
void set_influence_center_shift(float x, float y, float z);
void set_vehicle_base_influence_box(float min_x, float min_y, float min_z,
    float max_x, float max_y, float max_z);
void get_vehicle_base_influence_box(float* min_x, float* min_y, float* min_z,
    float* max_x, float* max_y, float* max_z);
void set_influence_box_enabled(int enabled = 1);
int get_influence_box_enabled();
void set_feeler_enabled(int feeler_index,
    int feeler_enabled);
int get_feeler_enabled(int feeler_index);
void set_feeler_rz_offset(int feeler_index,
    float feeler_rz_offset);
float get_feeler_rz_offset(int feeler_index);
void set_feeler_height(int feeler_index,
    float feeler_height);
float get_feeler_height(int feeler_index);
void set_feeler_collision_distance(int feeler_index,
    float collision_distance);
float get_feeler_collision_distance(int feeler_index);
void set_feeler_turn_distance(int feeler_index,
    float turn_distance);
float get_feeler_turn_distance(int feeler_index);
void set_feeler_turn_direction(int feeler_index,
    diguyCharacterFeelerTurnDirection turn_direction);
diguyCharacterFeelerTurnDirection get_feeler_turn_direction(int feeler_index);
void set_path_planner_preferred_region_mask(int mask);
int get_path_planner_preferred_region_mask();
void set_path_planner_preferred_cost_bias(float value);
float get_path_planner_preferred_cost_bias();
void set_path_planner_neutral_cost_bias(float value);
float get_path_planner_neutral_cost_bias();
void set_path_planner_repulsed_region_mask(int mask);
int get_path_planner_repulsed_region_mask();
void set_path_planner_repulsed_cost_bias(float value);
float get_path_planner_repulsed_cost_bias();
void set_path_planner_fallback_region(const char* fallback_region_name);
const char* get_path_planner_fallback_region();
void set_wander_uses_path_planner(int wander_uses_path_planner);
int get_wander_uses_path_planner();
void set_pursue_uses_path_planner(int pursue_uses_path_planner);
int get_pursue_uses_path_planner();
void set_max_LOS_travel_distance(float distance);
float get_max_LOS_travel_distance();
void set_desired_navpath_distance_threshold(float threshold);
float get_desired_navpath_distance_threshold();
void set_path_planner_cost_rule(int mask);
int get_path_planner_cost_rule();

```

Notes

The following functions return a pointer to an object of this type:

- ◆ `diguyCrowdProfile::get_default_agent_params()`
- ◆ `diguyCharacter::agent_get_initial_params()`
- ◆ `diguyCharacter::agent_get_current_params()`

For agents, changes to the initial parameters will have no effect until the scenario is reset, at which point the current parameters will be set to the initial parameters.

Changes to the crowd profile default agent params will have no effect on existing agent initial or current agent params, unless the function `diguyCharacter::agent_set_current_params_from_profile()` is called.

Unless otherwise noted, all functions of this class are callable from C++ and Scripts.

5.1.2 function `diguyAgentParams::set_posture`

Prototype:

```
void set_posture(diguyMotionPosture posture);
```

Description:

Sets the desired posture of agent actions.

5.1.3 function diguyAgentParams::get_posture

Prototype:

```
diguyMotionPosture get_posture();
```

Description:

Returns the desired posture of agent actions.

5.1.4 function diguyAgentParams::set_variant

Prototype:

```
void set_variant(diguyMotionVariant variant);
```

Description:

Sets the desired variant of agent actions.

5.1.5 function diguyAgentParams::get_variant

Prototype:

```
diguyMotionVariant get_variant();
```

Description:

Returns the desired variant of agent actions.

Behavior Functions

5.1.6 function diguyAgentParams::set_behavior

Prototype:

```
void set_behavior(diguyAgentBehavior behavior);
```

Description:

Sets the desired behavior of the agent.

For information on agent behaviors see the documentation for the enumeration [diguyAgentBehavior](#), or the DI-Guy AI User Manual.

5.1.7 function diguyAgentParams::get_behavior

Prototype:

```
diguyAgentBehavior get_behavior();
```

Description:

Returns the desired behavior of the agent.

5.1.8 function diguyAgentParams::set_behavior_path_shape

Prototype:

```
void set_behavior_path_shape(const char* behavior_path_shape_name);
```

Description:

Sets the behavior path shape of the agent.

Pass the name of an existing path shape in the scenario, or pass "" to un-set the behavior path shape.

5.1.9 function diguyAgentParams::get_behavior_path_shape

Prototype:

```
const char* get_behavior_path_shape();
```

Description:

Returns the name of the behavior path shape of the agent. If there is no behavior path shape, the string "" will be returned.

5.1.10 function diguyAgentParams::set_behavior_region

Prototype:

```
void set_behavior_region(const char* behavior_region_name);
```

Description:

Sets the behavior region of the agent.

Pass the name of an existing region in the scenario, "(crowd region)" to specify the region associated with the agent's crowd if it has one, or "" to un-set the behavior region.

5.1.11 function diguyAgentParams::get_behavior_region

Prototype:

```
const char* get_behavior_region();
```

Description:

Returns the name of the behavior region of the agent. If there is no behavior region, the string "" will be returned.

5.1.12 function diguyAgentParams::set_behavior_subregion

Prototype:

```
void set_behavior_subregion(diguySubregionIndex area);
```

Description:

Sets the behavior subregion of the agent.

5.1.13 function diguyAgentParams::get_behavior_subregion

Prototype:

```
diguySubregionIndex get_behavior_subregion();
```

Description:

Returns the behavior subregion of the agent.

5.1.14 function**diguyAgentParams::set_behavior_region_border_is_solid****Prototype:**

```
void set_behavior_region_border_is_solid(int is_solid);
```

Description:

Sets whether the walls of the region the agent is in are solid. If they are, the agent will treat the walls as if they are solid walls in the environment. If not, the agent is free to walk through them.

Pass 1 to make border walls solid, 0 to make them freely passable.

5.1.15 function**diguyAgentParams::get_behavior_region_border_is_solid****Prototype:**

```
int get_behavior_region_border_is_solid();
```

Description:

Returns whether the walls of the region the agent is in are solid.

5.1.16 function diguyAgentParams::set_focus_character

Prototype:

```
void set_focus_character(const char* focus_character_name);
```

Description:

Sets the focus character of this agent. What this means depends on which behavior the agent is doing. For example, when in the wander behavior, when the agent is still it will turn to face the focus character.

Pass the name of an existing scenario character to set a new focus character. Pass "" to un-set the focus character.

5.1.17 function diguyAgentParams::get_focus_character

Prototype:

```
const char* get_focus_character();
```

Description:

Returns the name of the focus character of this agent, "" if there is none.

5.1.18 function diguyAgentParams::set_focus_group**Prototype:**

```
void set_focus_group(const char* focus_group_name);
```

Description:

Sets the focus group of this agent. The focus group is similar to the focus character, except that characters the agent will focus on are selected from the focus group instead of being a single character.

Pass the name of an existing scenario group to set a new focus group. Pass "" to un-set the focus group.

5.1.19 function diguyAgentParams::get_focus_group**Prototype:**

```
const char* get_focus_group();
```

Description:

Returns the name of the focus group of this agent, "" if there is none.

5.1.20 function diguyAgentParams::set_focus_affects_gaze**Prototype:**

```
void set_focus_affects_gaze(int focus_affects_gaze);
```

Description:

Sets the whether the agent will gaze at its focus character, or whether setting the focus leaves gaze settings alone.

5.1.21 function diguyAgentParams::get_focus_affects_gaze**Prototype:**

```
int get_focus_affects_gaze();
```

Description:

Returns whether agent changes gaze settings to gaze at focus character or group.

5.1.22 function diguyAgentParams::set_max_focus_distance**Prototype:**

```
void set_max_focus_distance(float distance);
```

Description:

Sets the how far away characters in the focus group can be and still be considered as a focus character.

5.1.23 function diguyAgentParams::get_max_focus_distance**Prototype:**

```
float get_max_focus_distance();
```

Description:

Returns the maximum focus distance for focus groups.

5.1.24 function diguyAgentParams::set_move_on_t_min

Prototype:

```
void set_move_on_t_min(float t_min);
```

Description:

Sets the minimum amount of time this agent will stay in one place before moving on to a new location. This applies to the wander and mingle behaviors.

5.1.25 function diguyAgentParams::get_move_on_t_min

Prototype:

```
float get_move_on_t_min();
```

Description:

Returns the minimum amount of time before an agent will move on to a new location in certain behaviors.

5.1.26 function diguyAgentParams::set_move_on_t_max

Prototype:

```
void set_move_on_t_max(float t_max);
```

Description:

Similar to [set_move_on_t_min\(\)](#), but sets the maximum amount of time.

5.1.27 function diguyAgentParams::get_move_on_t_max

Prototype:

```
float get_move_on_t_max();
```

Description:

Similar to [get_move_on_t_min\(\)](#), but returns the maximum amount of time.

Attack Behavior Functions

5.1.28 function diguyAgentParams::set_attack_distance

Prototype:

```
void set_attack_distance(float distance);
```

Description:

Sets the distance at which an attacking agent will stop and fire at its target.

5.1.29 function diguyAgentParams::get_attack_distance

Prototype:

```
float get_attack_distance();
```

Description:

Returns the distance at which an attacking agent will stop and fire at its target.

5.1.30 function diguyAgentParams::set_attack_fire_interval

Prototype:

```
void set_attack_fire_interval(float time_interval);
```

Description:

Sets the interval in seconds at which an attacking agent will fire its weapon when its target is in range.

5.1.31 function diguyAgentParams::get_attack_fire_interval**Prototype:**

```
float get_attack_fire_interval();
```

Description:

Returns the interval in seconds at which an attacking agent will fire its weapon when its target is in range.

5.1.32 function diguyAgentParams::set_attack_stands_ground**Prototype:**

```
void set_attack_stands_ground(int value);
```

Description:

Sets whether agents in attack behavior will stay where they are and fire at targets of opportunity, or pursue targets.

If an agent is not told to stand its ground it can pursue its current target. The functions [set_attack_zone_enabled\(\)](#) and [set_attack_zone\(\)](#) might limit the extent that the agent can pursue.

Set to 1 for agents to stand their ground, 0 to allow them to pursue targets. Defaults to 0.

5.1.33 function diguyAgentParams::get_attack_stands_ground

Prototype:

```
int get_attack_stands_ground();
```

Description:

Returns the most recent setting made by `set_attack_stands_ground()`.

5.1.34 function diguyAgentParams::set_attack_zone

Prototype:

```
void set_attack_zone(float x, float y, float z, float radius);
```

Description:

If an agent is not designated to stand its ground (see `set_attack_stands_ground()`), and an attack zone is enabled (see `set_attack_zone_enabled()`), this function sets where and how far an agent can pursue its target.

5.1.35 function diguyAgentParams::get_attack_zone

Prototype:

```
int get_attack_zone(float* x, float* y, float* z, float* radius);
```

Description:

Returns position and radius of agent's attack zone.

5.1.36 function diguyAgentParams::set_attack_zone_enabled

Prototype:

```
void set_attack_zone_enabled(int use_attack_zone);
```

Description:

If an attacking agent is not designated to stand its ground (see [set_attack_stands_ground\(\)](#)) and is therefore free to pursue its target, the attack zone can limit how far it will go in pursuit.

Pass 1 to enable the attack zone, 0 to disable.

5.1.37 function diguyAgentParams::get_attack_zone_enabled**Prototype:**

```
int get_attack_zone_enabled();
```

Description:

Returns 0 if zone is disabled and agents can pursue indefinitely.

Wander Behavior Functions

5.1.38 function**diguyAgentParams::set_wander_max_wander_dist****Prototype:**

```
void set_wander_max_wander_dist(float distance);
```

Description:

Sets the maximum distance an agent will wander from its previous waiting point.

5.1.39 function**diguyAgentParams::set_wander_sets_desired_orientation****Prototype:**

```
void set_wander_sets_desired_orientation(int sets_desired_orientation);
```

Description:

Sets whether the wander behavior can change the desired orientation.

Pursue Speed Match Functions

5.1.40 function `diguyAgentParams::set_pursue_speed_match_enabled`

Prototype:

```
void set_pursue_speed_match_enabled(int enabled);
```

Description:

Sets whether the speed match functionality is enabled.

5.1.41 function

diguyAgentParams::get_pursue_speed_match_enabled

Prototype:

```
int get_pursue_speed_match_enabled();
```

Description:

Returns whether speed match functionality is enabled. If it is, agents are able to speed up their fastest animation to match their pursuit target.

5.1.42 function

diguyAgentParams::set_pursue_speed_match_update_dt

Prototype:

```
void set_pursue_speed_match_update_dt(float dt);
```

Description:

Sets how often the speed match speed is reevaluated. The default is every two seconds.

5.1.43 function**diguyAgentParams::get_pursue_speed_match_update_dt****Prototype:**

```
float get_pursue_speed_match_update_dt();
```

Description:

Returns the frequency that the speed match speed is updated.

5.1.44 function**diguyAgentParams::set_pursue_speed_match_max_factor****Prototype:**

```
void set_pursue_speed_match_max_factor(float factor);
```

Description:

Sets the maximum multiplier that can be applied to the fastest motion when pursuing in order to match speeds with the pursued target.

Example: If an agent's fastest action moves at 3 m/s, the pursue speed match functionality will allow the character to move up to 6 m/s.

The default is 2x. This allows a character with a fast movement motion to keep up with another individual that is moving faster than them, but not a fast moving vehicle.

5.1.45 function**diguyAgentParams::get_pursue_speed_match_max_factor****Prototype:**

```
float get_pursue_speed_match_max_factor();
```

Description:

Returns the maximum speed multiplier the speed match code will use.

5.1.46 function diguyAgentParams::set_pursue_offset**Prototype:**

```
void set_pursue_offset(float x, float y, float z);
```

Description:

Building block for setting up AI formations. Sets an offset that is used by the AI to modify the pursue location. This offset is considered to be in local space of the target and then is periodically bread-crumbed as the pursue target moves. This avoids small movements wildly confusing the pursuer. Often this is used with the [diguyFormation](#) class to data drive this info.

The z argument is currently ignored.

5.1.47 function diguyAgentParams::get_pursue_offset

Prototype:

```
int get_pursue_offset(float* x, float* y, float* z);
```

Description:

Accessor for current values of set_pursue_offset.

Path Follow Behavior Functions

5.1.48 function**diguyAgentParams::set_path_follow_pause_t_when_off_path****Prototype:**

```
void set_path_follow_pause_t_when_off_path(int pause_t);
```

Description:

Sets whether time advancement on the path being followed should occur when the agent is in a different behavior.

5.1.49 function**diguyAgentParams::get_path_follow_pause_t_when_off_path****Prototype:**

```
int get_path_follow_pause_t_when_off_path();
```

Description:

Returns the path follow pause t flag.

5.1.50 function**diguyAgentParams::set_path_follow_unpause_t_at_radius****Prototype:**

```
void set_path_follow_unpause_t_at_radius(float radius);
```

Description:

Sets how close the agent must be before a paused path follow will resume.

5.1.51 function**diguyAgentParams::get_path_follow_unpause_t_at_radius****Prototype:**

```
float get_path_follow_unpause_t_at_radius();
```

Description:

Returns the resume distance of a paused path follow.

5.1.52 function**diguyAgentParams::set_path_follow_position_lookahead_distance****Prototype:**

```
void set_path_follow_position_lookahead_distance(float distance);
```

Description:

Sets how far ahead on the path being followed the agent will look ahead for its desired orientation. A value of 1.0 usually provides good results.

5.1.53 function**diguyAgentParams::get_path_follow_position_lookahead_distance****Prototype:**

```
float get_path_follow_position_lookahead_distance();
```

Description:

Returns the lookahead distance of a path follow.

Trap Functions

5.1.54 function diguyAgentParams::set_wedged_untrap_method

Prototype:

```
void set_wedged_untrap_method(diguyAgentUntrapMethod untrap_method);
```

Description:

Sets the untrap method used when an agent becomes wedged in a corner.

For information on untrap methods see the documentation for the enumeration [diguyAgentUntrapMethod](#), or the DI-Guy AI User Manual.

5.1.55 function diguyAgentParams::get_wedged_untrap_method**Prototype:**

```
diguyAgentUntrapMethod get_wedged_untrap_method();
```

Description:

Returns the wedged untrap method.

5.1.56 function diguyAgentParams::set_wedged_trap_time_threshold**Prototype:**

```
void set_wedged_trap_time_threshold(float threshold);
```

Description:

Sets the how long, in seconds, the agent will walk into a corner before it decides it is wedged and applies the wedged untrap method.

5.1.57 function `diguyAgentParams::get_wedged_trap_time_threshold`

Prototype:

```
float get_wedged_trap_time_threshold();
```

Description:

Returns the how long the agent will walk into a corner before it decides it is wedged.

5.1.58 function `diguyAgentParams::set_first_no_progress_untrap_method`

Prototype:

```
void set_first_no_progress_untrap_method(diguyAgentUntrapMethod untrap_method);
```

Description:

Sets the untrap method used when an agent has not made progress toward its desired position.

For information on untrap methods see the documentation for the enumeration [diguyAgentUntrapMethod](#), or the DI-Guy AI User Manual.

5.1.59 function**diguyAgentParams::get_first_no_progress_untrap_method****Prototype:**

```
diguyAgentUntrapMethod get_first_no_progress_untrap_method();
```

Description:

Returns the first no progress untrap method.

5.1.60 function**diguyAgentParams::set_first_no_progress_trap_time_threshold****Prototype:**

```
void set_first_no_progress_trap_time_threshold(float threshold);
```

Description:

Sets the how long, in seconds, the agent is willing to make no progress toward its desired position before it applies the first no progress untrap method.

5.1.61 function**diguyAgentParams::get_first_no_progress_trap_time_threshold****Prototype:**

```
float get_first_no_progress_trap_time_threshold();
```

Description:

Returns the how long the agent can make no progress toward its desired position before it decides to apply an untrap method.

5.1.62 function**diguyAgentParams::set_second_no_progress_untrap_method****Prototype:**

```
void set_second_no_progress_untrap_method(diguyAgentUntrapMethod untrap_method);
```

Description:

Similar to [set_first_no_progress_untrap_method\(\)](#), but the time threshold is longer before the second no progress untrap method is applied.

5.1.63 function**diguyAgentParams::get_second_no_progress_untrap_method****Prototype:**

```
diguyAgentUntrapMethod get_second_no_progress_untrap_method();
```

Description:

Similar to [get_first_no_progress_untrap_method\(\)](#), but applied to the second no progress untrap method.

5.1.64 function**diguyAgentParams::set_second_no_progress_trap_time_thresho****Prototype:**

```
void set_second_no_progress_trap_time_threshold(float threshold);
```

Description:

Sets the how long, in seconds, the agent is willing to make no progress toward its desired position before it applies the second no progress untrap method.

5.1.65 function**diguyAgentParams::get_second_no_progress_trap_time_thresho****Prototype:**

```
float get_second_no_progress_trap_time_threshold();
```

Description:

Returns the how long the agent can make no progress toward its desired position before it decides to apply the second no progress untrap method.

5.1.66 function**diguyAgentParams::set_untrap_method_random_turn_max_tries****Prototype:**

```
void set_untrap_method_random_turn_max_tries(int max_tries);
```

Description:

Sets how many times the agent is willing to try applying the random turn untrap method before it gives up and moves on to the next untrap method.

5.1.67 function**diguyAgentParams::get_untrap_method_random_turn_max_tries****Prototype:**

```
int get_untrap_method_random_turn_max_tries();
```

Description:

Returns how many times the agent is willing to try applying the random turn untrap method.

5.1.68 function**diguyAgentParams::set_untrap_method_ghost_duration****Prototype:**

```
void set_untrap_method_ghost_duration(float duration);
```

Description:

Sets how many seconds the agent will be intangible when it is applying the ghost untrap method.

5.1.69 function**diguyAgentParams::get_untrap_method_ghost_duration****Prototype:**

```
float get_untrap_method_ghost_duration();
```

Description:

Returns how many seconds the agent will be intangible when it is applying the ghost untrap method.

Speed Type Parameters

5.1.70 function diguyAgentParams::set_speed_type_enabled**Prototype:**

```
int set_speed_type_enabled(diguyMotionSpeedType motion_speed_type,
                           int enabled);
```

Description:

Sets whether the agent will make use of actions that have the specified speed type.

For information on speed types see the documentation for the enumeration [diguyMotionSpeedType](#), or the DI-Guy AI User Manual.

5.1.71 function diguyAgentParams::get_speed_type_enabled**Prototype:**

```
int get_speed_type_enabled(diguyMotionSpeedType motion_speed_type);
```

Description:

Returns whether the agent will make use of actions that have the specified speed type.

If the speed type is enabled, the character must be outside the the associated radius for the action to apply. If the character enters that radius, then the next radius down is considered. If all enabled speed types have been exhausted, then the character should stop

5.1.72 function diguyAgentParams::set_speed_type_radius**Prototype:**

```
int set_speed_type_radius(diguyMotionSpeedType motion_speed_type,  
    float radius);
```

Description:

Sets at what radius, in meters, the agent will begin to play actions with the specified speed type, if that type is enabled.

See [set_speed_type_radius_hysteresis\(\)](#) for how a hysteresis value can affect the final radius.

5.1.73 function diguyAgentParams::get_speed_type_radius**Prototype:**

```
float get_speed_type_radius(diguyMotionSpeedType motion_speed_type);
```

Description:

Returns the radius of the specified speed type.

5.1.74 function**diguyAgentParams::set_speed_type_radius_hysteresis****Prototype:**

```
void set_speed_type_radius_hysteresis(float hysteresis_distance);
```

Description:

Sets the hysteresis of when the speed type changes based on how far the agent is from its desired position. A hysteresis value can help prevent constant switching back and forth of actions if an agent is moving right along the radius separating two speed type zones.

Example: a hysteresis of 2.0 means that if an agent switches from speed type fast to speed type medium at 2 meters, it won't switch back to fast unless it goes back out to 4 meters.

5.1.75 function**diguyAgentParams::get_speed_type_radius_hysteresis****Prototype:**

```
float get_speed_type_radius_hysteresis();
```

Description:

Returns the radius hysteresis as described in [set_speed_type_radius_hysteresis\(\)](#).

5.1.76 function `diguyAgentParams::set_speed_type_exact_action`

Prototype:

```
int set_speed_type_exact_action(diguyMotionSpeedType motion_speed_type,  
    const char* action);
```

Description:

Sets the exact action the agent will use if it is to use the specified speed type. This is useful if a specific stand action is desired, for example, for the still speed type.

To use the default action, pass "".

5.1.77 function `diguyAgentParams::get_speed_type_exact_action`

Prototype:

```
const char* get_speed_type_exact_action(diguyMotionSpeedType motion_speed_type);
```

Description:

Returns the exact action the agent will use if it is to use the specified speed type.

If the default action is to be used, returns "".

5.1.78 function diguyAgentParams::set_radius_variation

Prototype:

```
void set_radius_variation(float distance);
```

Description:

Sets the radius variation that will be applied to speed type radii as set by [set_speed_type_radius\(\)](#). Each agent will have a different amount of variation added to its radii.

Value is in meters.

5.1.79 function diguyAgentParams::get_radius_variation

Prototype:

```
float get_radius_variation();
```

Description:

Returns the radius variation for this agent as set by [set_radius_variation\(\)](#).

5.1.80 function

diguyAgentParams::set_turning_motions_enabled

Prototype:

```
int set_turning_motions_enabled(int enabled);
```

Description:

Sets if the character should use turning motions when changing desired orientation. As of diguy-12 defaults to true.

5.1.81 function

diguyAgentParams::get_turning_motions_enabled

Prototype:

```
int get_turning_motions_enabled();
```

Description:

Returns if the character is using turning motions when changing desired orientation. As of diguy-12 defaults to true.

Dynamic Avoidance Parameters

5.1.82 function diguyAgentParams::set_repulsion_radius

Prototype:

```
void set_repulsion_radius(diguyAgentRepulsionZone repulsion_zone,  
                          float radius);
```

Description:

Sets the radius, in meters, of the specified repulsion zone.

5.1.83 function diguyAgentParams::get_repulsion_radius

Prototype:

```
float get_repulsion_radius(diguyAgentRepulsionZone repulsion_zone);
```

Description:

Returns the radius, in meters, of the specified repulsion zone.

5.1.84 function diguyAgentParams::set_repulsion_factor

Prototype:

```
void set_repulsion_factor(diguyAgentRepulsionZone repulsion_zone,  
float factor);
```

Description:

Sets the repulsion factor of the specified repulsion zone. Higher repulsion factors will result in agents in the specified zone being pushed away harder.

5.1.85 function diguyAgentParams::get_repulsion_factor**Prototype:**

```
float get_repulsion_factor(diguyAgentRepulsionZone repulsion_zone);
```

Description:

Returns the repulsion factor of the specified repulsion zone.

5.1.86 function diguyAgentParams::set_repulsion_zone_enabled**Prototype:**

```
void set_repulsion_zone_enabled(diguyAgentRepulsionZone repulsion_zone,  
int enabled);
```

Description:

Sets whether the specified repulsion zone is enabled. Zones that are not enabled will have no effect on agents in them.

5.1.87 function diguyAgentParams::get_repulsion_zone_enabled**Prototype:**

```
int get_repulsion_zone_enabled(diguyAgentRepulsionZone repulsion_zone);
```

Description:

Returns whether the specified repulsion zone is enabled.

5.1.88 function diguyAgentParams::set_max_orientation_change_rate**Prototype:**

```
void set_max_orientation_change_rate(float max_orientation_change_rate);
```

Description:

Sets the fastest the agent can turn, in degrees per second, in response to being repulsed by another agent.

5.1.89 function `diguyAgentParams::get_max_orientation_change_rate`

Prototype:

```
float get_max_orientation_change_rate();
```

Description:

Returns the fastest the agent can turn, in degrees per second, in response to being repulsed by another agent.

5.1.90 function `diguyAgentParams::set_influence_center_shift`

Prototype:

```
void set_influence_center_shift(float x, float y, float z);
```

Description:

Sets the offset, in meters, that the low, medium, and high repulsion zones should have from the agent's position. The solid zone is not affected. Sometimes it is desirable for the repulsion zones to be moved slightly forward, to make it so that other agents approaching this agent from the side will prefer to move behind this agent rather than in front.

Vehicle-Specific Avoidance Parameters

5.1.91 function

`diguyAgentParams::set_vehicle_base_influence_box`

Prototype:

```
void set_vehicle_base_influence_box(float min_x, float min_y, float min_z,  
    float max_x, float max_y, float max_z);
```

Description:

Sets the base influence box, in meters, of a vehicle agent. The default influence box is the vehicle's axis-aligned bounding box.

5.1.92 function**diguyAgentParams::get_vehicle_base_influence_box****Prototype:**

```
void get_vehicle_base_influence_box(float* min_x, float* min_y, float* min_z,  
float* max_x, float* max_y, float* max_z);
```

Description:

Returns the vehicle's base influence box.

5.1.93 function diguyAgentParams::set_influence_box_enabled**Prototype:**

```
void set_influence_box_enabled(int enabled = 1);
```

Description:

Sets whether or not the influence box is active (true by default).

5.1.94 function diguyAgentParams::get_influence_box_enabled

Prototype:

```
int get_influence_box_enabled();
```

Description:

Returns whether or not the influence box is active

Feeler Functions

5.1.95 function diguyAgentParams::set_feeler_enabled**Prototype:**

```
void set_feeler_enabled(int feeler_index,  
                        int feeler_enabled);
```

Description:

Sets whether the specified feeler is enabled. Feelers that are not enabled will have no effect on the agent.

5.1.96 function diguyAgentParams::get_feeler_enabled**Prototype:**

```
int get_feeler_enabled(int feeler_index);
```

Description:

Returns whether the specified feeler is enabled.

5.1.97 function diguyAgentParams::set_feeler_rz_offset**Prototype:**

```
void set_feeler_rz_offset(int feeler_index,  
    float feeler_rz_offset);
```

Description:

Sets, in degrees, how far counter-clockwise from forward the feeler is. Zero degrees is straight forward, 90 is to the left, etc. By default the eight feelers are located at 45 degree intervals around the agent.

5.1.98 function diguyAgentParams::get_feeler_rz_offset**Prototype:**

```
float get_feeler_rz_offset(int feeler_index);
```

Description:

Returns, in degrees, how far counter-clockwise from forward the feeler is.

5.1.99 function diguyAgentParams::set_feeler_height**Prototype:**

```
void set_feeler_height(int feeler_index,  
    float feeler_height);
```

Description:

Sets, in meters, how high the feeler is from the ground. In general feelers should be at about waist level for human characters.

5.1.100 function diguyAgentParams::get_feeler_height**Prototype:**

```
float get_feeler_height(int feeler_index);
```

Description:

Returns, in meters, how high the feeler is from the ground.

5.1.101 function

diguyAgentParams::set_feeler_collision_distance

Prototype:

```
void set_feeler_collision_distance(int feeler_index,  
    float collision_distance);
```

Description:

Sets, in meters, how far away from the agent the feeler will return collision results. Agents will be pushed away from the object collided with until they are not colliding anymore.

5.1.102 function

diguyAgentParams::get_feeler_collision_distance

Prototype:

```
float get_feeler_collision_distance(int feeler_index);
```

Description:

Returns, in meters, how far away from the agent the feeler will return collision results.

5.1.103 function diguyAgentParams::set_feeler_turn_distance**Prototype:**

```
void set_feeler_turn_distance(int feeler_index,  
                             float turn_distance);
```

Description:

Sets, in meters, how far away from the agent the feeler will return turn results. Agents will turn left or right to try to move along the face of a felt obstacle, instead of walking straight into it.

5.1.104 function diguyAgentParams::get_feeler_turn_distance**Prototype:**

```
float get_feeler_turn_distance(int feeler_index);
```

Description:

Returns, in meters, how far away from the agent the feeler will return turn results.

5.1.105 function diguyAgentParams::set_feeler_turn_direction**Prototype:**

```
void set_feeler_turn_direction(int feeler_index,  
    diguyCharacterFeelerTurnDirection turn_direction);
```

Description:

Sets which direction the feeler will turn the agent when it touches something within the turn distance.

5.1.106 function diguyAgentParams::get_feeler_turn_direction**Prototype:**

```
diguyCharacterFeelerTurnDirection get_feeler_turn_direction(int feeler_index);
```

Description:

Returns which direction the feeler will turn the agent when it touches something within the turn distance.

Path Planner Parameter Functions

5.1.107 function

`diguyAgentParams::set_path_planner_preferred_region_mask`

Prototype:

```
void set_path_planner_preferred_region_mask(int mask);
```

Description:

Sets the preferred traveling region mask.

When an agent plans a path, the preferred areas will be considered the easiest to cross and the path should gravitate to them.

The mask argument should be an or'ed together list of `diguySubregionMasks`. In C++ or Perl masks can be combined with the '|' operator, in Lua the bit library can be used: `bit.bor(MASK_A, MASK_B, ...)`

5.1.108 function**diguyAgentParams::get_path_planner_preferred_region_mask****Prototype:**

```
int get_path_planner_preferred_region_mask();
```

Description:**Returns:**

the preferred region mask; see
[set_path_planner_preferred_region_mask\(\)](#) and
[set_path_planner_neutral_cost_bias\(\)](#)

5.1.109 function**diguyAgentParams::set_path_planner_preferred_cost_bias****Prototype:**

```
void set_path_planner_preferred_cost_bias(float value);
```

Description:

Sets how much less expensive it is to cross spaces that are in the desired subregion. (Repulsive and neutral regions have their own cost bias; see [set_path_planner_repulsed_cost_bias\(\)](#).)

This value should be < 1 . The default value is 0.25.

5.1.110 function**diguyAgentParams::get_path_planner_preferred_cost_bias****Prototype:**

```
float get_path_planner_preferred_cost_bias();
```

Returns:

the cost bias of preferred regions during A* searches; see [set_path_planner_preferred_cost_bias\(\)](#)

5.1.111 function**diguyAgentParams::set_path_planner_neutral_cost_bias****Prototype:**

```
void set_path_planner_neutral_cost_bias(float value);
```

Description:

Sets how much more expensive it is to cross spaces that are not in a desired subregion. (Repulsive regions have their own cost bias; see [set_path_planner_repulsed_cost_bias\(\)](#).)

Defaults to 1.1, A value < 1 will cause the planner to run faster but explore fewer points, possibly missing preferred regions. Values greater than 1 will explore more points but will take more time to run.

Pass [DIGUY_DEFAULT_FLOAT](#) for the value to specify that neutral regions should be avoided completely. This can, however, cause path planning to fail more often.

5.1.112 function

diguyAgentParams::get_path_planner_neutral_cost_bias

Prototype:

```
float get_path_planner_neutral_cost_bias();
```

Returns:

the cost bias of neutral regions during A* searches; see
[set_path_planner_neutral_cost_bias\(\)](#)

5.1.113 function

diguyAgentParams::set_path_planner_repulsed_region_mask

Prototype:

```
void set_path_planner_repulsed_region_mask(int mask);
```

Description:

Sets the repulsive traveling region mask.

When an agent plans a path the repulsive areas will be considered the hardest to cross and the planner will try to avoid them.

The mask argument should be an or'ed together list of diguySubregionMasks. In C++ or Perl masks can be combined with the '|' operator, in Lua the bit library can be used: `bit.bor(MASK_A, MASK_B, ...)`

5.1.114 function

diguyAgentParams::get_path_planner_repulsed_region_mask

Prototype:

```
int get_path_planner_repulsed_region_mask();
```

Returns:

the repulsed region mask; see
[set_path_planner_repulsed_region_mask\(\)](#)

5.1.115 function

diguyAgentParams::set_path_planner_repulsed_cost_bias

Prototype:

```
void set_path_planner_repulsed_cost_bias(float value);
```

Description:

Sets how much more expensive it will be to cross spaces that are marked as repulsive.

This value should generally be much larger than 1. The default value is 100.

Pass [DIGUY_DEFAULT_FLOAT](#) for the value to specify that repulsive regions should be avoided completely. This can, however, cause path planning to fail more often.

5.1.116 function**diguyAgentParams::get_path_planner_repulsed_cost_bias****Prototype:**

```
float get_path_planner_repulsed_cost_bias();
```

Returns:

the cost bias of repulsive regions during A* searches; see [set_path_planner_repulsed_cost_bias\(\)](#)

5.1.117 function**diguyAgentParams::set_path_planner_fallback_region****Prototype:**

```
void set_path_planner_fallback_region(const char* fallback_region_name);
```

Description:

Sets the name of the region to try to use if the path planner fails to work on the current behavior region. This is often the name of a global navmesh.

5.1.118 function `diguyAgentParams::get_path_planner_fallback_region`

Prototype:

```
const char* get_path_planner_fallback_region();
```

Returns:

The name of the fallback region that the path planner will fall back to if a search on the behavior region fails.

5.1.119 function `diguyAgentParams::set_wander_uses_path_planner`

Prototype:

```
void set_wander_uses_path_planner(int wander_uses_path_planner);
```

Description:

Sets whether the wander behavior will try to use the path planner when the agent can't see it's target location with a viability check.

5.1.120 function diguyAgentParams::get_wander_uses_path_planner

Prototype:

```
int get_wander_uses_path_planner();
```

Returns:

whether the wander behavior uses the path planner; see
[set_wander_uses_path_planner\(\)](#)

5.1.121 function diguyAgentParams::set_pursue_uses_path_planner

Prototype:

```
void set_pursue_uses_path_planner(int pursue_uses_path_planner);
```

Description:

Sets whether the pursue behavior will try to use the path planner when the agent can't see it's target with a viability check.

The agent will do a new visibility check at each waypoint in the navigation path.

5.1.122 function diguyAgentParams::get_pursue_uses_path_planner

Prototype:

```
int get_pursue_uses_path_planner();
```

Returns:

whether the pursue behavior uses the path planner; see
[set_wander_uses_path_planner\(\)](#)

5.1.123 function diguyAgentParams::set_max_LOS_travel_distance

Prototype:

```
void set_max_LOS_travel_distance(float distance);
```

Description:

Sets the maximum distance the wander behavior will travel without using the path planner, when the agent can see it's target location with a Line-Of-Sight viability check.

This value defaults to 1000 meters.

Adjusting it lower is meant for users that have an environment that is well tagged with preferred and repulsed regions.

For example, just because an agent can see the store across an eight lane highway doesn't mean it should run straight there.

5.1.124 function**diguyAgentParams::get_max_LOS_travel_distance****Prototype:**

```
float get_max_LOS_travel_distance();
```

Returns:

the Line-Of-Sight travel distance threshold; see [set_max_LOS_travel_distance\(\)](#).

5.1.125 function**diguyAgentParams::set_desired_navpath_distance_threshold****Prototype:**

```
void set_desired_navpath_distance_threshold(float threshold);
```

Description:

Sets the distance from the final point on a navpath that the guide algorithm will attempt to achieve before declaring that its destination has been reached.

The default distance is 0.31 meters.

In general this value should only be changed if agent ends up "orbiting" the final desired position.

5.1.126 function**diguyAgentParams::get_desired_navpath_distance_threshold****Prototype:**

```
float get_desired_navpath_distance_threshold();
```

Returns:

distance from final point in navpath at which the desired position is considered acquired; see

[set_desired_navpath_distance_threshold\(\)](#)

Experimental Functions

5.1.127 function diguyAgentParams::set_path_planner_cost_rule**Prototype:**

```
void set_path_planner_cost_rule(int mask);
```

see [diguyNavMeshPathPlannerCostMask](#) for acceptable values.

5.1.128 function diguyAgentParams::get_path_planner_cost_rule**Prototype:**

```
int get_path_planner_cost_rule();
```

5.2 diguyApp

diguyApp.h

DI-Guy SDK Version 12.5.2

Includes: [declspec_diguy.h](#) | [diguy_typedefs.h](#) | [diguy_constants.h](#) | [stdlib.h](#)

Contents:

- ◆ [General Functions](#)
- ◆ [Directory Functions](#)
- ◆ [Networking Functions](#)
- ◆ [C++ Only Callback Functions](#)
- ◆ [C++ and Script Callback Functions](#)
- ◆ [Event Handler Functions](#)
- ◆ [Plugin Functions](#)
- ◆ [Shader Functions](#)
- ◆ [Performance Functions](#)
- ◆ [Threading Functions](#)
- ◆ [User Input Functions](#)
- ◆ [DI-Guy Author Functions](#)
- ◆ [Draw Functions](#)
- ◆ [Physics Module Functions](#)
- ◆ [Library Functions](#)
- ◆ [Experimental Munition Type Functions](#)
- ◆ [Experimental Functions](#)
- ◆ [Experimental Log Channel Functions](#)
- ◆ [Deprecated Functions](#)

Link against: `libdiguy`

5.2.1 function int

Prototype:

```
typedef int  (*diguyModuleFunctionVoid)();
```

5.2.2 class diguyApp

```
class BDI_DECLSPEC diguy diguyApp
{
public:
    static diguyApp* get_app();
    int get_num_scenarios();
    diguyScenario* get_scenario_at_index(int index);
    int get_num_blend_functions();
    const char* get_blend_function_name_at_index(int index);
    int is_debug();
    int is_api_mode();
    int is_module_enabled(const char* module_name);
    diguyScenario* get_current_scenario();
    static const char* get_diguy_base_dir_path();
    static const char* get_diguy_subdir_path(const char* subdir);
    diguyExerciseInterface* get_exercise_interface();
#ifdef CPLUSPLUS_ONLY
    int add_default_scenario_callback(int callback_id,
        diguyScenarioCallback* callback,
        void* callback_params = 0,
        void* callback_user_data = 0,
        int remove_on_scenario_load = 1);
    int remove_default_scenario_callback(int callback_id,
        diguyScenarioCallback* callback);
    int remove_default_scenario_callback_with_user_data(int callback_id,
        void* callback_user_data);
#endif
    int add_default_scenario_callback_script(int callback_id,
        const char* callback_script,
        const char* callback_script_type,
        int remove_on_scenario_load = 1);
    int remove_default_scenario_callback_script(int callback_id,
        const char* callback_script,
        const char* callback_script_type);
#ifdef CPLUSPLUS_ONLY
    int register_scenario_event_handler(const char* handler_name,
        diguyScenarioCallback* callback,
        void* callback_user_data = 0);
#endif
    int register_scenario_event_handler_from_library(const char* handler_name,
        const char* library_name,
        const char* function_name,
        void* callback_user_data = 0);
    int register_scenario_event_handler_script(const char* handler_name,
        const char* handler_script,
        const char* handler_script_type = NULL);
    int has_registered_scenario_event_handler(const char* handler_name);
    int unregister_scenario_event_handler(const char* handler_name);
    int load_plugin(const char* plugin_name,
        int required_by_scenario = 0,
```

```

        int unload_on_scenario_close = 0,
        int accepts_mouse_input = 1);
int unload_plugin(const char* plugin_name);
void unload_all_plugins();
int get_num_plugins();
const char* get_name_of_plugin(int index);
diguyGraphicsShaderProgram* load_shader_program(const char* shader_name,
        const char* vertex_shader_filename,
        const char* pixel_shader_filename);
void reload_shader_programs();
diguyGraphicsShaderProgram* find_shader_program(const char* shader_name);
int get_num_shader_programs();
diguyGraphicsShaderProgram* get_shader_program_at_index(int index);
diguyGraphicsShaderProgram* get_current_bound_shader_program();
diguyGraphicsShaderTechnique* find_shader_technique(const char* shader_technique_name);
int get_num_shader_techniques();
diguyGraphicsShaderTechnique* get_shader_technique_at_index(int index);
int set_max_character_shader_quality_level(int quality_level);
int get_max_character_shader_quality_level();
int set_max_scene_object_shader_quality_level(int quality_level);
int get_max_scene_object_shader_quality_level();
int get_use_sRGB_textures();
int set_precompute_quaternions_enabled(int enabled);
int get_precompute_quaternions_enabled();
int set_num_rendering_passes(int num_rendering_passes);
int get_num_rendering_passes();
int get_render_pass_flags();
int set_render_pass_flags(int flags);
int get_ideal_num_threads();
int set_num_threads(int num_threads);
int get_num_threads();
int reserve_num_threads(int num_reserve_threads);
int set_base_input_mode(diguyScenarioInputMode mode);
diguyScenarioInputMode get_base_input_mode();
int get_last_clicked_xyz(float* x, float* y, float* z);
void set_last_clicked_xyz(float x, float y, float z);
diguyCharacter* get_last_clicked_character();
int get_mouse_screen_xy(float* sx, float* sy);
int get_xyz_under_mouse(float* x, float* y, float* z);
void set_lua_object_waiting_for_input(const char* object_name);
const char* get_lua_object_waiting_for_input();
diguyAuthorInterface* get_author_interface();
diguyViewPainter* get_view_painter();
void set_collision_group_detection_flag(diguyPhysicsCollisionGroup collision_group1,
        diguyPhysicsCollisionGroup collision_group2,
        int enable_collisions);
int get_collision_group_detection_flag(diguyPhysicsCollisionGroup collision_group1,
        diguyPhysicsCollisionGroup collision_group2);
void set_physics_sim_ground_plane_enabled(int enabled);
int get_physics_sim_ground_plane_enabled();
void set_physics_sim_ground_plane_height(float height);
float get_physics_sim_ground_plane_height();
static const char* get_standard_bdi_library_suffix();
static const char* get_diguy_version_library_suffix();
static const char* get_compiler_library_suffix();
static void* load_module_library(const char* module_lib_name,
        int include_diguy_standard_library_suffixes = 1,
        int* load_status = NULL);
static void* resolve_module_library_symbol(void* module_handle, const char* symbol_name);
static void unload_module_library(void* module_handle);
int get_sound_factory_is_initialized();

```

```

int set_munition_type_DIS_impact_septet(const char* munition_type,
    int sep1, int sep2, int sep3, int sep4, int sep5, int sep6, int sep7);
int set_munition_type_DIS_tracer_septet(const char* munition_type,
    int sep1, int sep2, int sep3, int sep4, int sep5, int sep6, int sep7);
const char* get_munition_type_from_DIS_septet(
    int sep1, int sep2, int sep3, int sep4, int sep5, int sep6, int sep7,
    int include_last_septet_value = 1);
int unload_unreferenced_graphics_files(int garbage_collect_appearances = 1);
int reload_texture_files();
int set_threshold_to_unload_textures(int number);
int get_threshold_to_unload_textures();
int set_texture_unload_frequency(float seconds);
float get_texture_unload_frequency();
int get_unload_unused_after_scenario_load();
int set_unload_unused_after_scenario_load(int val);
int set_cache_optimized_motion_files(int enabled = 0);
int get_cache_optimized_motion_files();
int memory_tracker_print_report(int include_permanent_objects = 0, int just_markers = 0);
int memory_tracker_drop_marker();
void set_reflected_entity_message_list(const char* message);
const char* get_reflected_entity_message_list();
const char* get_scene_object_file_name_from_ui();
int show_generate_navmesh_ui();
diguyLogChannel* create_log_channel();
void destroy_log_channel(diguyLogChannel*);
int get_num_log_channels();
diguyLogChannel* get_log_channel_at_index(int index);
int set_current_painting_subregion(diguySubregionIndex index);

```

General Functions

Unless otherwise noted all functions in this class section are callable from:

- ◆ C++
- ◆ Script

In lua static function are callable as `diguyApp_function_name` or via the `this_app` global.

5.2.3 function diguyApp::get_app

Prototype:

```
static diguyApp* get_app();
```

Description:

This static function returns a pointer to the DI-Guy application. All functions below can then be called through this pointer.

Returns:

a pointer of type `diguyApp`

C++ Example:

```
diguyApp* app = diguyApp::get_app();
```

5.2.4 function diguyApp::get_num_scenarios

Prototype:

```
int get_num_scenarios();
```

Returns:

number of scenarios in the app

5.2.5 function diguyApp::get_scenario_at_index

Prototype:

```
diguyScenario* get_scenario_at_index(int index);
```

Returns:

pointer of type [diguyScenario](#); NULL if no scenario at the specified index

Arguments:

index	index of the scenario; indices start at 0

5.2.6 function diguyApp::get_num_blend_functions

Prototype:

```
int get_num_blend_functions();
```

Returns:

number of blend functions available

5.2.7 function diguyApp::get_blend_function_name_at_index

Prototype:

```
const char* get_blend_function_name_at_index(int index);
```

Returns:

name of blend function at specified index; NULL if no blend function at specified index

Arguments:

index	index of the blend function; indices start at 0

5.2.8 function diguyApp::is_debug

Prototype:

```
int is_debug();
```

Description:

This function returns whether or not the application is in debug mode. Useful for choosing which versions of shared libraries to load.

Returns:

1 if app is in debug mode; 0 if not

5.2.9 function diguyApp::is_api_mode

Prototype:

```
int is_api_mode();
```

Description:

This function returns whether or not the DI-Guy SDK is being used inside of DI-Guy Scenario or as an embedded C++ library.

Returns:

1 if being used as a library; 0 if not

5.2.10 function diguyApp::is_module_enabled

Prototype:

```
int is_module_enabled(const char* module_name);
```

Description:

This function returns whether or not the specified module is currently enabled.

Returns:

1 if module is enabled; 0 if not

5.2.11 function diguyApp::get_current_scenario

Prototype:

```
diguyScenario* get_current_scenario();
```

Returns:

pointer of type `diguyScenario`; NULL if no current scenario

Directory Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

Networking Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.2.12 function `diguyApp::get_exercise_interface`

Prototype:

```
diguyExerciseInterface* get_exercise_interface();
```

Description:

Gets a pointer to the exercise interface. If DI-Guy is not on the network, the returned pointer will be NULL.

Returns:

pointer of type `diguyExerciseInterface` on success, NULL on failure or if not on the network

C++ Only Callback Functions

5.2.13 function `diguyApp::add_default_scenario_callback`

Prototype:

```
int add_default_scenario_callback(int callback_id,  
    diguyScenarioCallback* callback,  
    void* callback_params = 0,  
    void* callback_user_data = 0,  
    int remove_on_scenario_load = 1);
```

Description:

This function sets a default user callback that will be added to all new scenarios. See [diguyScenario::add_callback\(\)](#) for details.

5.2.14 function `diguyApp::remove_default_scenario_callback`

Prototype:

```
int remove_default_scenario_callback(int callback_id,  
    diguyScenarioCallback* callback);
```

Description:

This function removes a default user callback previously added by `add_default_scenario_callback()`. See `diguyScenario::remove_callback()` for details.

Returns:

0 on success, -1 on failure

5.2.15 function `diguyApp::remove_default_scenario_callback_with_user_data`

Prototype:

```
int remove_default_scenario_callback_with_user_data(int callback_id,  
    void* callback_user_data);
```

Description:

This function removes a default user callback previously added by `add_default_scenario_callback()`. See `diguyScenario::remove_callback_with_user_data()` for details.

C++ and Script Callback Functions

5.2.16 function `diguyApp::add_default_scenario_callback_script`

Prototype:

```
int add_default_scenario_callback_script(int callback_id,  
    const char* callback_script,  
    const char* callback_script_type,  
    int remove_on_scenario_load = 1);
```

Description:

This function adds a default scenario callback script. The callback script will be added to all new scenarios. See [diguyScenario::add_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

5.2.17 function**diguyApp::remove_default_scenario_callback_script****Prototype:**

```
int remove_default_scenario_callback_script(int callback_id,  
const char* callback_script,  
const char* callback_script_type);
```

Description:

This function removes a default scenario callback script. See [diguyScenario::remove_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

Event Handler Functions

DI-Guy event handlers are similar to callbacks, but have a name with which they can be referenced. This name allows DI-Guy Scenario to map event handlers to callback ids using a UI, and allows for these mappings to be saved and restored as part of the scenario.

5.2.18 function `diguyApp::register_scenario_event_handler`

Prototype:

```
int register_scenario_event_handler(const char* handler_name,
    diguyScenarioCallback* callback,
    void* callback_user_data = 0);
```

Description:

This function is analogous to the `diguyScenario::register_character_event_handler()` function, but is for scenarios rather than characters.

The registered event handler can be mapped to an event via the DI-Guy Scenario UI, or by the function `diguyScenario::map_event_handler_to_callback_id()`.

It can also be manually invoked by the function call `diguyScenario::manually_invoke_event_handler()`.

Callable From:

◇ C++

5.2.19 function `diguyApp::register_scenario_event_handler_from_library`

Prototype:

```
int register_scenario_event_handler_from_library(const char* handler_name,
    const char* library_name,
    const char* function_name,
    void* callback_user_data = 0);
```

Description:

This function is analogous to the `diguyScenario::register_character_event_handler_from_library()` function, but is for scenarios rather than characters.

The registered event handler can be mapped to an event via the DI-Guy Scenario UI, or by the function `diguyScenario::map_event_handler_to_callback_id()`.

It can also be manually invoked by the function call `diguyScenario::manually_invoke_event_handler()`.

5.2.20 function**diguyApp::register_scenario_event_handler_script****Prototype:**

```
int register_scenario_event_handler_script(const char* handler_name,
    const char* handler_script,
    const char* handler_script_type = NULL);
```

Description:

This function is analogous to the `diguyScenario::register_character_event_handler_script()` function, but is for scenarios rather than characters.

The registered event handler can be mapped to an event via the DI-Guy Scenario UI, or by the function `diguyScenario::map_event_handler_to_callback_id()`.

It can also be manually invoked by the function call `diguyScenario::manually_invoke_event_handler()`.

5.2.21 function**diguyApp::has_registered_scenario_event_handler****Prototype:**

```
int has_registered_scenario_event_handler(const char* handler_name);
```

Description:

This function is analogous to the `diguyScenario::has_registered_character_event_handler()` function, but is for scenarios rather than characters.

5.2.22 function diguyApp::unregister_scenario_event_handler

Prototype:

```
int unregister_scenario_event_handler(const char* handler_name);
```

Description:

This function is analogous to the [diguyScenario::unregister_character_event_handler\(\)](#) function, but is for scenarios rather than characters.

Plugin Functions

See complete overview at [\ref diguyScenarioPluginOverview](#)

[\page diguyScenarioPluginOverview](#) DI-Guy Scenario Plug-in Overview

[\tableofcontents](#)

A DI-Guy plugin is a dynamic library that adds additional functionality to DI-Guy Scenario or to an application using the standard DI-Guy SDK libraries.

Plugins can be loaded in three ways:

- ◆ specified on DI-Guy Scenario command line using "+plugin xyz" argument
- ◆ added to a scenario in DI-Guy Scenario via the Scenario Objects*Specialized Objects*Plugins page
- ◆ by calling [diguyApp::load_plugin\(\)](#)

Once a plugin has successfully been loaded, functions from the plugin will be called at specific times during scenario execution.

Plugins can be unloaded in four ways:

- ◆ DI-Guy Scenario is shut down
- ◆ if part of a scenario, and the plugin is removed from the scenario by pressing Unload on the Plugins page
- ◆ if part of a scenario, "Unload on Scenario Close" on the Plugins page is checked, and a new scenario is loaded
- ◆ by calling `diguyApp::unload_plugin()`

Once a plugin has successfully been loaded, functions from the plugin will be called at specific times during scenario execution.

Following is a list of the functions that DI-Guy will look for in the library. Note that a plugin does not need to implement all of these functions; if a function is missing the plugin will quietly skip it. Only the `diguy_version()` function must be implemented.

Note that new functions may be added to plugins in the future.

void diguy_version(int version_major, int* version_minor, int* version_point)*

This function will be called once, when the plugin is being loaded. The function should return, via the passed int pointers, the DI-Guy version number as defined in `diguy_constants.h`. DI-Guy will compare these numbers against the current version number to make sure the plugin is compatible with the current version.

C++ Example:

```
__declspec(dllexport) void
diguy_version(int* version_major, int* version_minor, int* version_point)
{
    *version_major = DIGUY_VERSION_MAJOR;
    *version_minor = DIGUY_VERSION_MINOR;
    *version_point = DIGUY_VERSION_POINT;
}
```

void initialize_plugin(diguyApp app)*

This function will be called once, when the plugin is being loaded.

If the plugin is being loaded because it was specified on the DI-Guy Scenario command line, this function will be called as DI-Guy Scenario is initializing, before the first scenario is created.

If the plugin is part of a scenario (is shown in the DI-Guy Scenario Scenario ObjectsSpecialized ObjectsPlugins page), the call will happen when the scenario is loaded.

If the plugin is being loaded because of a `load_plugin()` call, this function will be called during the `load_plugin()` call.

void deinitialize_plugin(diguyApp* app)

This function will be called once, when the plugin is being unloaded. The call will happen while DI-Guy Scenario is shutting down, or during the `unload_plugin()` call if the

plugin is being explicitly unloaded by function call.

void initialize(diguyScenario* s)

This function will be called when the passed scenario first becomes aware of the plugin. This will be when the scenario is created if the plugin has already been loaded, or during the `load_plugin()` call if the plugin is loaded after the scenario was created.

void deinitialize(diguyScenario* s)

This function will be called when the scenario is deleted, or when `unload_plugin()` is called.

void pre_update(diguyScenario* s)

This function will be called during the `diguyScenario::update()` call, before anything else is changed due to the update. (i.e., characters will not have moved yet, etc.)

void post_update(diguyScenario* s)

This function will be called during the `diguyScenario::update()` call, after everything else is changed due to the update. (i.e., characters will have moved, etc.)

void pre_draw(diguyScenario* s)

This function will be called during the `diguyScenario::draw()` call, before anything else is drawn.

void post_draw(diguyScenario* s)

This function will be called during the `diguyScenario::draw()` call, after everything else is drawn.

void reset(diguyScenario* s)

This function will be called when the scenario is reset, either due to pressing the Reset button in DI-Guy Scenario, or calling the `diguyScenario::reset()` function. The scenario is also implicitly reset when a .dss file is loaded.

diguyPluginMouseCursor get_mouse_cursor(diguyScenario* s)

This function should return which type of cursor should be used when the mouse is in a 3D view. See the documentation for `diguyPluginMouseCursor` for which values are available. If this function is not exported from the plugin, the default cursor `DIGUY_PLUGIN_MOUSE_CURSOR_LEFT_ARROW` will be used.

void mouse_press(diguyScenario* s, diguyPluginMouseEvent* mouse_event)

This function is called on an initial press of any mouse button in a 3D view. Use the query functions of the `diguyPluginMouseEvent` object to retrieve which view the press occurred in and the x and y screen coordinates of the press.

void mouse_drag(diguyScenario* s, diguyPluginMouseEvent* mouse_event)

This function is called when the mouse is moved in a 3D while one or more buttons are pressed.

void mouse_release(diguyScenario* s, diguyPluginMouseEvent* mouse_event)

This function is called when any mouse button is released in a 3D view.

void mouse_move(diguyScenario s, diguyPluginMouseEvent* mouse_event)*

This function is called when the mouse is moved in a 3D while no buttons are pressed.

int key_press(diguyScenario* s, diguyPluginKeyboardEvent* keyboard_event)

This function is called on an initial press of any key in a 3D view. Use the query functions of the [diguyPluginKeyboardEvent](#) object to retrieve which view the press occurred in and the code of the key. The code can be a standard ASCII value, or one of the DIGUY_SPECIAL_KEY_* values defined in diguy_constants.h. Return 1 to indicate that the key was handled and that the default key handler should not be called. Return 0 for the default key handler to be called.

int key_release(diguyScenario* s, diguyPluginKeyboardEvent* keyboard_event)

The same as key_press(), but called when the key is released.

void save_review_data(diguyScenario* s, const char* filename)

This function will be called when after-action review data for the scenario is saved by a call to [diguyScenario::save_review_data\(\)](#).

void load_review_data(diguyScenario* s, const char* filename)

This function will be called when after-action review data for the scenario is loaded by a call to [diguyScenario::load\(\)](#).

Windows Specific:

Plugin functions should be declared with `__declspec(dllexport)` to make sure that the symbols are properly exported from the DLL.

The LIBC version that the plugin uses (MD, MDd, MT, or MTd) must be the same as that of DI-Guy Scenario or the DI-Guy libraries used. If this does not happen conflicting versions of the DI-Guy DLLs will be loaded and the program will be unstable.

5.2.23 function diguyApp::load_plugin

Prototype:

```
int load_plugin(const char* plugin_name,
               int required_by_scenario = 0,
               int unload_on_scenario_close = 0,
               int accepts_mouse_input = 1);
```

Description:

This function adds a new dynamic library plugin to the app. Plugins work within DI-Guy Scenario as well as the DI-Guy SDK.

Arguments:

<code>plugin_name</code>	name of the plugin; should be dynamic library name without the extension
<code>required_by_scenario</code>	if set to 1, this plugin is required for proper execution of this scenario; when the scenario is loaded this plugin will be loaded as well if it is not already active; default value is 0
<code>unload_on_scenario_close</code>	if set to 1, this plugin will be unloaded if this scenario is closed or a new scenario is opened; default value is 0
<code>accepts_mouse_input</code>	if set to 1, this plugin will receive mouse events via the mouse plugin functions when the input mode is "Plugin"; default value is 1

Returns:

0 on success, -1 on failure

Windows Specific:

The DLL should be placed in a directory that is in the path, or in the same directory as the DI-Guy Scenario executable.

The DLL version of DI-Guy Scenario or the SDK must be used for plugins to work.

See complete overview at [\ref diguyScenarioPluginOverview](#)

5.2.24 function diguyApp::unload_plugin

Prototype:

```
int unload_plugin(const char* plugin_name);
```

Description:

This function unloads a dynamic library plugin from the app that was added using the [load_plugin\(\)](#) function.

Arguments:

name	name of the plugin

Returns:

0 on success, -1 on failure

See complete overview at [\ref diguyScenarioPluginOverview](#)

5.2.25 function diguyApp::unload_all_plugins

Prototype:

```
void unload_all_plugins();
```

Description:

Unloads all plug-ins.

5.2.26 function diguyApp::get_num_plugins

Prototype:

```
int get_num_plugins();
```

Returns:

This function returns the number of dynamic library plugins currently loaded.

See complete overview at [\ref diguyScenarioPluginOverview](#)

5.2.27 function diguyApp::get_name_of_plugin

Prototype:

```
const char* get_name_of_plugin(int index);
```

Returns:

the name of a specific plug-in at a specified index

Arguments:

See complete overview at [\ref diguyScenarioPluginOverview](#)

Shader Functions

5.2.28 function diguyApp::load_shader_program

Prototype:

```
diguyGraphicsShaderProgram* load_shader_program(const char* shader_name,  
const char* vertex_shader_filename,  
const char* pixel_shader_filename);
```

Description:

This function loads (or reloads) shader source files into a shader program. The shader program can be attached to a character in two ways:

First, a shader may be explicitly added to a character using the `diguyCharacter::set_shader_program( )` function.

Second, a shader may be implicitly added to a character if its appearance specifies an associated shader required to make the appearance look right. Most often this is for appearances that are skinned, requiring a shader to transform the vertices correctly. DI-Guy most often uses the 'diguy_object' shader for this.

If a shader program with the same name has already been loaded, this function will replace it.

DI-Guy Scenario uses GLSL shaders.

Arguments: Returns:

pointer to object type [diguyGraphicsShaderProgram](#) on success,
NULL on failure

5.2.29 function diguyApp::reload_shader_programs

Prototype:

```
void reload_shader_programs();
```

Description:

This function reloads all previously loaded shader programs, using the same source filenames as were used when the shader was initially loaded.

This function can be used to reload/replace an existing shader with the same name, allowing for quick edit and test cycles when working on new shaders.

5.2.30 function diguyApp::find_shader_program

Prototype:

```
diguyGraphicsShaderProgram* find_shader_program(const char* shader_name);
```

Description:

This function looks for the previously loaded shader program with the given name.

Arguments: Returns:

pointer to object type [diguyGraphicsShaderProgram](#) on success,
NULL on failure

5.2.31 function diguyApp::get_num_shader_programs

Prototype:

```
int get_num_shader_programs();
```

Returns:

number of loaded shader programs

5.2.32 function diguyApp::get_shader_program_at_index

Prototype:

```
diguyGraphicsShaderProgram* get_shader_program_at_index(int index);
```

Description:

This function returns the shader program at the specified index.

Arguments: Returns:

pointer to object type [diguyGraphicsShaderProgram](#) on success,
NULL on failure

5.2.33 function diguyApp::get_current_bound_shader_program

Prototype:

```
diguyGraphicsShaderProgram* get_current_bound_shader_program();
```

Description:

This function returns the shader program that's currently active

Returns:

pointer to object type [diguyGraphicsShaderProgram](#) on success,
NULL on failure

5.2.34 function diguyApp::find_shader_technique

Prototype:

```
diguyGraphicsShaderTechnique* find_shader_technique(const char* shader_technique_name)
```

Description:

This function looks for the previously loaded shader technique with the given name.

Arguments: Returns:

pointer to object type `diguyGraphicsShaderTechnique` on success,
NULL on failure

5.2.35 function diguyApp::get_num_shader_techniques

Prototype:

```
int get_num_shader_techniques();
```

Returns:

number of loaded shader techniques

5.2.36 function diguyApp::get_shader_technique_at_index**Prototype:**

```
diguyGraphicsShaderTechnique* get_shader_technique_at_index(int index);
```

Returns:

a pointer to a [diguyGraphicsShaderTechnique](#) on success at the specified index, NULL on failure.

Arguments:**5.2.37 function diguyApp::set_max_character_shader_quality_level****Prototype:**

```
int set_max_character_shader_quality_level(int quality_level);
```

Description:

This function sets the max shader quality level for characters in the application. See [diguyGraphicsShaderProgram.h](#) for the [diguyGraphicsShaderQualityLevel](#) enum. This quality level, which can range from no lighting to normal mapped characters, will be used to pick an appropriate shader from a shader technique.

5.2.38 function

diguyApp::get_max_character_shader_quality_level

Prototype:

```
int get_max_character_shader_quality_level();
```

5.2.39 function

diguyApp::set_max_scene_object_shader_quality_level

Prototype:

```
int set_max_scene_object_shader_quality_level(int quality_level);
```

Description:

This function sets the maximum shader quality level for scene objects in the application. See

[diguyApp::set_max_character_shader_quality_level\(\)](#) for more information.

5.2.40 function**diguyApp::get_max_scene_object_shader_quality_level****Prototype:**

```
int get_max_scene_object_shader_quality_level();
```

5.2.41 function diguyApp::get_use_sRGB_textures**Prototype:**

```
int get_use_sRGB_textures();
```

Returns:

the value that was passed in for `init.use_sRGB_corrected_textures` allows end users to procedurally choose gamma correction behavior.

Performance Functions

5.2.42 function diguyApp::set_precompute_quaternions_enabled

Prototype:

```
int set_precompute_quaternions_enabled(int enabled);
```

Description:

This function tells DI-Guy to use quaternions for all joint angles in motions. It should be called after DI-Guy is initialized and before characters are created. By default DI-Guy data uses Euler angles.

Some motion operations are faster using quaternions, while some are not. In general anything that requires blending of motions will be faster when using quaternions.

This function should not have any negative effect on character motion quality.

Arguments: Returns:

0 on success, -1 on failure

C++ Example:

```
diguy_graphics_initialize();  
diguyApp* the_app = diguy_get_app();  
the_app->set_precompute_quaternions_enabled(1);
```

5.2.43 function diguyApp::get_precompute_quaternions_enabled**Prototype:**

```
int get_precompute_quaternions_enabled();
```

Returns:

current setting of whether quaternions are enabled; see
[set_precompute_quaternions_enabled\(\)](#)

5.2.44 function diguyApp::set_num_rendering_passes

Prototype:

```
int set_num_rendering_passes(int num_rendering_passes);
```

Description:

This function sets how many rendering passes DI-Guy will use to draw objects such as characters and scene objects.

By default DI-Guy uses two rendering passes. Fully opaque objects are drawn in the first pass; semi-transparent objects are drawn in the second.

If few or no objects have transparency, the number of rendering passes can be set to 1, which will increase rendering performance.

If the number of rendering passes is set to 1, semi-transparent objects may incorrectly occlude objects behind them, depending on object rendering order.

This function can be called at any time.

Arguments: Returns:

0 on success, -1 on failure

5.2.45 function diguyApp::get_num_rendering_passes

Prototype:

```
int get_num_rendering_passes();
```

Returns:

current setting of number of rendering passes; see
[set_num_rendering_passes\(\)](#)

5.2.46 function diguyApp::get_render_pass_flags

Prototype:

```
int get_render_pass_flags();
```

Description:

Gets current render pass flags. See diguyRenderPassFlags. The values are or'd together.

5.2.47 function diguyApp::set_render_pass_flags

Prototype:

```
int set_render_pass_flags(int flags);
```

Description:

Sets current render pass flags. See diguyRenderPassFlags.

Threading Functions

5.2.48 function diguyApp::get_ideal_num_threads

Prototype:

```
int get_ideal_num_threads();
```

Description:

This function returns the ideal number of threads available on the system. This will correspond to the number of available processors.

Returns:

ideal number of threads on the system; -1 on failure to detect

5.2.49 function diguyApp::set_num_threads

Prototype:

```
int set_num_threads(int num_threads);
```

Description:

This function sets how many threads are available to DI-Guy.

Arguments: Returns:

0 on success, -1 on failure

5.2.50 function diguyApp::get_num_threads

Prototype:

```
int get_num_threads();
```

Returns:

number of threads available to DI-Guy

5.2.51 function diguyApp::reserve_num_threads

Prototype:

```
int reserve_num_threads(int num_reserve_threads);
```

Description:

This function reserves threads for the client application by setting the number of threads available to DI-Guy. DI-Guy will use up to (max_num_threads - reserve_num_threads).

Arguments: Returns:

0 on success, -1 on failure

User Input Functions

5.2.52 function diguyApp::set_base_input_mode

Prototype:

```
int set_base_input_mode(diguyScenarioInputMode mode);
```

Description:

This function changes the current Input Mode of the DI-Guy Scenario application or a DI-Guy Author Host IG.

Arguments:

mode	the new base Input Mode of DI-Guy Scenario or DI-Guy Author

Returns:

-1 if mode change failed (e.g., tried for AI-related mode but AI not available, 0 on no error

5.2.53 function diguyApp::get_base_input_mode

Prototype:

```
diguyScenarioInputMode get_base_input_mode();
```

Returns:

the current Input Mode of the DI-Guy Scenario application or a DI-Guy Author Host IG.

5.2.54 function diguyApp::get_last_clicked_xyz

Prototype:

```
int get_last_clicked_xyz(float* x, float* y, float* z);
```

Description:

This function returns the 3D world position of the last clicked point in a DI-Guy Scenario 3D view.

This function is meant to complement the use of callbacks with callback id [DIGUY_SCENARIO_INPUT_MODE_POINT_SELECT](#).

Arguments:

`x, y, z` position in meters from the origin

Pass NULL for any values that are not needed.

5.2.55 function diguyApp::set_last_clicked_xyz**Prototype:**

```
void set_last_clicked_xyz(float x, float y, float z);
```

Description:

This function sets the 3D world position of the last clicked point in a DI-Guy Scenario 3D view.

This function is meant to complement the use of callbacks with callback id [DIGUY_SCENARIO_INPUT_MODE_POINT_SELECT](#).

Arguments:

`x, y, z` position in meters from the origin

5.2.56 function diguyApp::get_last_clicked_character**Prototype:**

```
diguyCharacter* get_last_clicked_character();
```

Description:

This function returns the last character clicked by the mouse in DI-Guy Scenario.

This function is meant to complement the use of callbacks with callback id [DIGUY_SCENARIO_INPUT_MODE_POINT_SELECT](#).

Returns:

pointer of type [diguyCharacter](#); may be NULL if no character has been clicked

5.2.57 function diguyApp::get_mouse_screen_xy**Prototype:**

```
int get_mouse_screen_xy(float* sx, float* sy);
```

Description:

This function returns the current screen coordinates x and y position of the mouse.

5.2.58 function diguyApp::get_xyz_under_mouse**Prototype:**

```
int get_xyz_under_mouse(float* x, float* y, float* z);
```

Description:

This function returns the current xyz coordinates under the mouse location, this is a fairly speedy test since the octree is used. Typically this function is used to help build interactive UI tools.

Arguments:

x, y, z resulting position in meters from the origin

5.2.59 function diguyApp::set_lua_object_waiting_for_input

Prototype:

```
void set_lua_object_waiting_for_input(const char* object_name);
```

Description:

This function sets up the UI to notify a lua object of mouse events via messages while in input mode

[DIGUY_SCENARIO_INPUT_MODE_POINT_SELECT](#).

The following callbacks are generated:

- CALLBACK_ID_UI_MOUSE_DOWN
- CALLBACK_ID_UI_MOUSE_UP
- CALLBACK_ID_UI_RIGHT_MOUSE_DOWN
- CALLBACK_ID_UI_RIGHT_MOUSE_UP

See `luaUIMachine:get_user_selected_point( )` to see an example of usage. Typically this function is used to help build interactive UI tools.

Arguments:

`object_name` resulting position in meters from the origin

5.2.60 function diguyApp::get_lua_object_waiting_for_input**Prototype:**

```
const char* get_lua_object_waiting_for_input();
```

Returns:

The name of the lua object that's currently waiting for input events. Can return NULL if there's no active object.

DI-Guy Author Functions

5.2.61 function diguyApp::get_author_interface

Prototype:

```
diguyAuthorInterface* get_author_interface();
```

Description:

This function is equivalent to the function `diguy_author_get_author_interface( )` in `diguy_author.h`. It is made available here so it is available to scripts.

Draw Functions

5.2.62 function diguyApp::get_viewPainter

Prototype:

```
diguyViewPainter* get_viewPainter();
```

Returns:

This function returns a pointer to a [diguyViewPainter](#) object. This object can be used to draw lines, text, and other primitives in the 3D views of DI-Guy Scenario.

Physics Module Functions

5.2.63 function diguyApp::set_collision_group_detection_flag**Prototype:**

```
void set_collision_group_detection_flag(diguyPhysicsCollisionGroup collision_group1,
    diguyPhysicsCollisionGroup collision_group2,
    int enable_collisions);
```

Description:

This function sets whether the two specified collision groups should collide against each other.

Arguments:

collision_group1 first collision group
 collision_group2 second collision group
 enable_collisions pass 1 to have groups collide, 0 to not

5.2.64 function diguyApp::get_collision_group_detection_flag**Prototype:**

```
int get_collision_group_detection_flag(diguyPhysicsCollisionGroup collision_group1,
    diguyPhysicsCollisionGroup collision_group2);
```

Description:

This function returns whether the two specified collision groups should collide against each other.

Arguments:

collision_group1 first collision group
 collision_group2 second collision group

Returns:

1 if collision groups collide, 0 if not

5.2.65 function

diguyApp::set_physics_sim_ground_plane_enabled

Prototype:

```
void set_physics_sim_ground_plane_enabled(int enabled);
```

Description:

The physics simulation module can have an optional ground plane enabled through which no objects will pass. This function sets whether the ground plane is enabled.

Arguments:

enabled pass 1 to enable ground plane, 0 to disable

5.2.66 function

diguyApp::get_physics_sim_ground_plane_enabled

Prototype:

```
int get_physics_sim_ground_plane_enabled();
```

Returns:

1 if physics sim ground plane enabled, 0 if not

5.2.67 function

diguyApp::set_physics_sim_ground_plane_height

Prototype:

```
void set_physics_sim_ground_plane_height(float height);
```

Description:

The physics simulation module can have an optional ground plane enabled through which no objects will pass. This function sets the height of the ground plane.

Arguments:

height	height in meters

5.2.68 function

diguyApp::get_physics_sim_ground_plane_height

Prototype:

```
float get_physics_sim_ground_plane_height();
```

Returns:

height of physics sim ground plane

Library Functions

These functions provide support for loading DI-Guy modules at runtime, without need for linking against module import libraries.

5.2.69 function diguyApp::load_module_library

Prototype:

```
static void* load_module_library(const char* module_lib_name,
                                int include_diguy_standard_library_suffixes = 1,
                                int* load_status = NULL);
```

Description:

Loads dynamic library module. The handle returned can be passed into `resolve_symbol()` or `unload_module_library()`.

Returns:

handle as (void *) pointer, NULL if library is not found or failed to load.

Load status is set to:

0 : load successful -1 : library not found -2 : ?

5.2.70 function diguyApp::resolve_module_library_symbol**Prototype:**

```
static void* resolve_module_library_symbol(void* module_handle, const char* symbol_name)
```

Description:

Returns pointer to function provided by already-loaded dynamic library module. Pointer should be cast to one of:

– diguyModuleFunctionInt – diguyModuleFunctionVoid –
diguyModuleFunctionVoidReturnsVoid

Returns:

pointer to function, or NULL if function not found.

5.2.71 function diguyApp::unload_module_library**Prototype:**

```
static void unload_module_library(void* module_handle);
```

Description:

Unloads dynamic library module, given handle to it.

5.2.72 function diguyApp::get_sound_factory_is_initialized

Prototype:

```
int get_sound_factory_is_initialized();
```

Description:

Returns 1 if a sound factory is initialized.

Experimental Munition Type Functions

These functions provide access to munition types defined for DI-Guy.

5.2.73 function diguyApp::set_munition_type_DIS_impact_septet**Prototype:**

```
int set_munition_type_DIS_impact_septet(const char* munition_type,  
    int sep1, int sep2, int sep3, int sep4, int sep5, int sep6, int sep7);
```

Description:

Sets override values for the specified munition type's DIS impact septet.

Munition types are defined in the config file `munition_types.cfg`, located under the DI-Guy installation config directory.

Arguments:**5.2.74 function diguyApp::set_munition_type_DIS_tracer_septet****Prototype:**

```
int set_munition_type_DIS_tracer_septet(const char* munition_type,  
    int sep1, int sep2, int sep3, int sep4, int sep5, int sep6, int sep7);
```

Description:

Similar to `set_munition_type_DIS_impact_septet()`, but overrides the septet for tracer type.

5.2.75 function diguyApp::get_munition_type_from_DIS_septet**Prototype:**

```
const char* get_munition_type_from_DIS_septet(  
    int sep1, int sep2, int sep3, int sep4, int sep5, int sep6, int sep7,  
    int include_last_septet_value = 1);
```

Description:

Returns the munition type name based on the passed septet values.

Arguments:

Experimental Functions

5.2.76 function diguyApp::unload_unreferenced_graphics_files**Prototype:**

```
int unload_unreferenced_graphics_files(int garbage_collect_appearances = 1);
```

Description:

Causes garbage-collection to unload graphics files that have been previously loaded into memory, but are unused, freeing up memory. There may be a frame hitch if graphics files are to be used again and need to be reloaded.

Arguments:**5.2.77 function diguyApp::reload_texture_files****Prototype:**

```
int reload_texture_files();
```

Description:

Causes textures that have changes to be reloaded

5.2.78 function diguyApp::set_threshold_to_unload_textures**Prototype:**

```
int set_threshold_to_unload_textures(int number);
```

Description:

How many textures in memory before texture unloading starts (defaults to 5000)

5.2.79 function diguyApp::get_threshold_to_unload_textures**Prototype:**

```
int get_threshold_to_unload_textures();
```

5.2.80 function diguyApp::set_texture_unload_frequency**Prototype:**

```
int set_texture_unload_frequency(float seconds);
```

Description:

How often to check if inactive textures should be unloaded.

5.2.81 function diguyApp::get_texture_unload_frequency**Prototype:**

```
float get_texture_unload_frequency();
```

5.2.82 function**diguyApp::get_unload_unused_after_scenario_load****Prototype:**

```
int get_unload_unused_after_scenario_load();
```

Description:

Should unused assets be garbage collected after loading a new scenario.

5.2.83 function**diguyApp::set_unload_unused_after_scenario_load****Prototype:**

```
int set_unload_unused_after_scenario_load(int val);
```

5.2.84 function diguyApp::set_cache_optimized_motion_files**Prototype:**

```
int set_cache_optimized_motion_files(int enabled = 0);
```

Description:

Sets whether optimized versions of motion files should be cached to disk when they are loaded.

This is still experimental. The default is 0, don't enable caching.

Arguments:**5.2.85 function diguyApp::get_cache_optimized_motion_files****Prototype:**

```
int get_cache_optimized_motion_files();
```

Returns:

1 if caching of optimized motion files is enabled, 0 if disabled

5.2.86 function diguyApp::memory_tracker_print_report

Prototype:

```
int memory_tracker_print_report(int include_permanent_objects = 0, int just_markers
```

Description:

Prints to the log a report about memory usage and leaks.

Arguments:

5.2.87 function diguyApp::memory_tracker_drop_marker

Prototype:

```
int memory_tracker_drop_marker();
```

Description:

Add a marker for use by [memory_tracker_print_report\(\)](#)

5.2.88 function diguyApp::set_reflected_entity_message_list**Prototype:**

```
void set_reflected_entity_message_list(const char* message);
```

Description:

Provide DI-Guy Scenario with a list of messages that the user can access from the right-click menu. When a message is selected, it is sent over the network to the AIs of remote entities. If the AIs can process the message, they will. Typically used to kill remote entities from a master server.

Arguments:**5.2.89 function diguyApp::get_reflected_entity_message_list****Prototype:**

```
const char* get_reflected_entity_message_list();
```

Description:

Returns a string containing a comma-separated list of supported reflected-entity messages, as set by `set_reflected_entity_message_list()`

5.2.90 function diguyApp::get_scene_object_file_name_from_ui**Prototype:**

```
const char* get_scene_object_file_name_from_ui();
```

Description:

In DI-Guy Scenario pops up a user interface asking the end user to pick a scene object file name.

5.2.91 function diguyApp::show_generate_navmesh_ui**Prototype:**

```
int show_generate_navmesh_ui();
```

Description:

Causes DI-Guy to display UI relevant to navmesh generation

Experimental Log Channel Functions

5.2.92 function diguyApp::create_log_channel

Prototype:

```
diguyLogChannel* create_log_channel();
```

Returns:

Newly-created log channel

5.2.93 function diguyApp::destroy_log_channel**Prototype:**

```
void destroy_log_channel(diguyLogChannel*);
```

Description:

Destroys specified log channel

5.2.94 function diguyApp::get_num_log_channels**Prototype:**

```
int get_num_log_channels();
```

Returns:

Number of log channels

5.2.95 function diguyApp::get_log_channel_at_index

Prototype:

```
diguyLogChannel* get_log_channel_at_index(int index);
```

Returns:

Log channel at index

Deprecated Functions

5.2.96 function diguyApp::set_current_painting_subregion**Prototype:**

```
int set_current_painting_subregion(diguySubregionIndex index);
```

Deprecated as of DI-Guy 12.0.0. Use

[diguyAuthorVisualRegionPaintbrush::set_painting_subregion\(\)](#)
instead.

5.3 diguyAuthorInterface

diguyAuthorInterface.h

DI-Guy SDK Version 12.5.2

Includes: [stdio.h](#) | [declspec_diguy.h](#) | [diguy_constants.h](#) | [diguy_vector_classes.h](#)

Contents:

- ◆ [Mouse Functions](#)
- ◆ [Keyboard Functions](#)
- ◆ [Selected Object Functions](#)
- ◆ [Terrain and Intersection Virtual Functions](#)
- ◆ [Author UI Functions](#)
- ◆ [Author UI Status Update Functions](#)
- ◆ [Utility Vertex Generation Functions](#)

5.3.1 constants

```
#define AUTHOR_UI_MAIN_WINDOW_NAME      "main_window"
#define AUTHOR_UI_INPUT_MODE_WINDOW_NAME "input_mode"
#define AUTHOR_UI_TIME_CONTROL_WINDOW_NAME "time_control"
#define AUTHOR_UI_LOG_WINDOW_NAME       "log"
#define AUTHOR_UI_VISIBILITY_WINDOW_NAME "visibility"
#define AUTHOR_UI_ELEMENTS_WINDOW_NAME   "elements"
#define AUTHOR_UI_CHARACTER_WINDOW_NAME  "character"
#define AUTHOR_UI_CROWD_WINDOW_NAME      "crowd"
#define AUTHOR_UI_MIND_EDITOR_WINDOW_NAME "mind_editor"
#define AUTHOR_UI_AI_INSPECTOR_WINDOW_NAME "ai_inspector"
```

Link against: `libdiguy`

Summary:

The [diguyAuthorInterface](#) class provides an interface for sending input to DI-Guy Author from a DI-Guy Author enabled IG, receiving commands from DI-Guy Author, and querying for other DI-Guy data.

5.3.2 class diguyAuthorInterface

```
class BDI_DECLSPEC_diguy diguyAuthorInterface
{
public:
    diguyAuthorVisualRegionPaintbrush* get_region_paintbrush();
    void set_mouse_screen_coordinates(int x, int y);
    void set_mouse_world_coordinates(diguyVec3f world_coords);
    void set_mouse_button_l(int button_state);
    void set_mouse_button_r(int button_state);
```

```

void set_mouse_button_m(int button_state);
void set_mouse_wheel_change(float mouse_wheel_change);
void apply_mouse_inputs();
void set_keyboard_shift(int key_state);
void set_keyboard_ctrl(int key_state);
void set_keyboard_alt(int key_state);
void set_key_pressed(int key, int key_state);
int set_selected_diguy_uid(long uid);
int set_selected_diguy_uid_from_string(const char* uid_string);
int get_uid_base_type(long uid);
int get_uid_base_type_from_string(const char* uid_string);
diguyCharacter* get_selected_character();
diguyCharacterPath* get_selected_character_path();
diguyWaypoint* get_selected_waypoint();
diguyPathShape* get_selected_path_shape();
diguyGraphicsLink* get_selected_link();
diguyGraphicsShape* get_selected_shape();
virtual int convert_screen_to_world(const diguyVec3f&screen_xyz,
    diguyVec3f* world_xyz) = 0;
virtual int convert_world_to_screen(const diguyVec3f&world_xyz,
    diguyVec3f* screen_xyz) = 0;
virtual float get_altitude(float x, float y, float old_z,
    float from_height = 10.0f,
    int* valid_result = NULL) = 0;
virtual int update_selected_objects(const diguyVec3f&screen_xyz) = 0;
virtual int check_z_clearance(const diguyVec3f&pos,
    const float radius,
    const float ground_angle,
    float z_clear_height,
    int check_below = 1) const = 0;
int set_author_ui_window_visible(const char* window_name, int visible);
int get_author_ui_window_visible(const char* window_name);
int set_author_ui_panel_visible(const char* panel_name);
int set_author_ui_select_crowd_character(diguyCharacter* character);
int set_author_ui_select_code_browser_script(diguyCharacter* character);
#ifdef CPLUSPLUS_ONLY
    virtual void author_ui_window_visibility_changed(const char* window_name,
        int is_visible);
#endif
#ifdef CPLUSPLUS_ONLY
    static int generate_sphere_vertices(float radius,
        int num_slices,
        int num_layers,
        const diguyVec3f&position_offset,
        const diguyVec3f&orientation,        // degrees
        int* num_vertices,
        diguyVec3f** vertices,
        diguyVec3f** normals,
        diguyVec2f** texture_indices,
        int indices_for_wireframe,
        int* num_indices,
        unsigned int** indices);
    static int generate_cone_vertices(float radius,
        float height,
        int num_slices,
        int num_layers,
        const diguyVec3f&position_offset,
        const diguyVec3f&orientation,        // degrees
        int* num_vertices,
        diguyVec3f** vertices,
        diguyVec3f** normals,

```

```

        diguyVec2f** texture_indices,
        int indices_for_wireframe,
        int* num_indices,
        unsigned int** indices);
static int generate_cylinder_vertices(float radius,
        float height,
        int num_slices,
        int num_layers,
        const diguyVec3f&position_offset,
        const diguyVec3f&orientation,        // degrees
        int* num_vertices,
        diguyVec3f** vertices,
        diguyVec3f** normals,
        diguyVec2f** texture_indices,
        int indices_for_wireframe,
        int* num_indices,
        unsigned int** indices);
static int generate_box_vertices(diguyVec3f min,
        diguyVec3f max,
        const diguyVec3f&position_offset,
        const diguyVec3f&orientation,        // degrees
        int* num_vertices,
        diguyVec3f** vertices,
        diguyVec3f** normals,
        diguyVec2f** texture_indices,
        int indices_for_wireframe,
        int* num_indices,
        unsigned int** indices);
static void delete_generated_vertices(diguyVec3f* vertices,
        diguyVec3f* normals,
        diguyVec2f* texture_indices,
        unsigned int* indices);
#endif

```

5.3.3 function diguyAuthorInterface::get_region_paintbrush

Prototype:

```
diguyAuthorVisualRegionPaintbrush* get_region_paintbrush();
```

Returns a pointer to the region paintbrush. Returned pointer may be NULL.

Mouse Functions

In most DI-Guy Author input modes the position of the mouse and state of the mouse buttons are very important. These functions allow the Host IG to tell DI-Guy Author the state of the mouse. They should be called from the Host IG when mouse events occur.

These functions should not be called when DI-Guy Author is inactive. DI-Guy Author is inactive when `diguyApp::get_base_input_mode( )` returns `DIGUY_SCENARIO_INPUT_MODE_NONE`.

5.3.4 function `diguyAuthorInterface::set_mouse_screen_coordinates`

Prototype:

```
void set_mouse_screen_coordinates(int x, int y);
```

Description:

Sets the current position of the mouse in screen coordinates.

This should be done whenever a DI-Guy input mode is active and the mouse moves.

Note that nothing is done with this information until `apply_mouse_inputs()` is called.

Arguments:

x	horizontal pixel from left edge of window
y	vertical pixel from bottom edge of window

5.3.5 function `diguyAuthorInterface::set_mouse_world_coordinates`

Prototype:

```
void set_mouse_world_coordinates(diguyVec3f world_coords);
```

Description:

Sets the current position of the mouse in world coordinates. This should be the intersection point of the ray that would begin at the mouse's screen position, and extend into the far distance (at least as far as the graphics far plane).

This should be done whenever a DI-Guy input mode is active, and a point in the 3D world is requested, such as during a call to `convert_screen_to_world()`.

Note that nothing is done with this information until `apply_mouse_inputs()` is called.

Arguments:

`world_coords` mouse position in world coordinates, in meters from the origin

5.3.6 function `diguyAuthorInterface::set_mouse_button_l`

Prototype:

```
void set_mouse_button_l(int button_state);
```

Description:

Sets the current state of the left mouse button.

This should be done whenever a DI-Guy input mode is active and the a mouse button is pressed.

Note that nothing is done with this information until `apply_mouse_inputs()` is called.

Arguments:

`button_state` 1 for down, 0 for up

5.3.7 function diguyAuthorInterface::set_mouse_button_r

Prototype:

```
void set_mouse_button_r(int button_state);
```

Description:

Same as `set_mouse_button_l( )`, but for the right mouse button.

5.3.8 function diguyAuthorInterface::set_mouse_button_m

Prototype:

```
void set_mouse_button_m(int button_state);
```

Description:

Same as `set_mouse_button_m( )`, but for the middle mouse button.

5.3.9 function diguyAuthorInterface::set_mouse_wheel_change

Prototype:

```
void set_mouse_wheel_change(float mouse_wheel_change);
```

Description:

Sets the change in the mouse wheel position since the last call to this function.

This should be done whenever a DI-Guy input mode is active and the mouse wheel is moved.

Note that nothing is done with this information until [apply_mouse_inputs\(\)](#) is called.

Arguments:

mouse_wheel_change (units documentation pending)

5.3.10 function diguyAuthorInterface::apply_mouse_inputs

Prototype:

```
void apply_mouse_inputs();
```

Description:

Applies effects of inputs set by [set_mouse_screen_coordinates\(\)](#), [set_mouse_button_l\(\)](#), etc. DI-Guy Author will make modifications on edited objects based on these inputs.

Keyboard Functions

In most DI-Guy Author input modes the state of various keys can change the behavior of the input mode. These functions allow the Host IG to tell DI-Guy Author the state of the keyboard.

Keyboard inputs aren't nearly important to DI-Guy Author as mouse events are, and it's easily possible for the Host IG to not want to pass key events to DI-Guy Author as the Host IG wants to handle them itself.

These functions should not be called when DI-Guy Author is inactive. DI-Guy Author is inactive when `diguyApp::get_base_input_mode()` returns `DIGUY_SCENARIO_INPUT_MODE_NONE`.

5.3.11 function diguyAuthorInterface::set_keyboard_shift

Prototype:

```
void set_keyboard_shift(int key_state);
```

Description:

Sets the current state of the shift key. The shift key can change the results of mouse inputs in the 3D window.

This should be done whenever a DI-Guy input mode is active and the state of the shift key changes.

Arguments:

`key_state` 1 for down, 0 for up

5.3.12 function diguyAuthorInterface::set_keyboard_ctrl

Prototype:

```
void set_keyboard_ctrl(int key_state);
```

Description:

Same as `set_keyboard_shift()`, but for ctrl key.

5.3.13 function diguyAuthorInterface::set_keyboard_alt

Prototype:

```
void set_keyboard_alt(int key_state);
```

Description:

Same as `set_keyboard_alt()`, but for alt key.

5.3.14 function diguyAuthorInterface::set_key_pressed

Prototype:

```
void set_key_pressed(int key, int key_state);
```

Description:

Sets the current state of the specified key. Key presses can change the input mode, cause a reset, load camera settings, etc.

Note: Often it will be better for the Host IG to handle key presses and make the appropriate DI-Guy calls manually.

Arguments:

key	ASCII code of pressed key
key_state	1 for down, 0 for up

Selected Object Functions

Most DI-Guy objects have a unique identifier (uid): a numerical value that can be used to identify the object. In DI-Guy Author they are useful for setting the names or other identifiers of objects in the 3D scene, so that if those objects are clicked on it can be determined whether the clicked point is on a DI-Guy object, and if so which one.

Different DI-Guy objects have uids in different ranges. The object type can be derived from the uid by calling `get_uid_base_type()`.

5.3.15 function diguyAuthorInterface::set_selected_diguy_uid

Prototype:

```
int set_selected_diguy_uid(long uid);
```

Description:

Sets the current selected DI-Guy object based on the passed uid. If the uid matches an object in the scenario, that object will become the selected object.

This function is typically called during a `update_selected_objects( )` call.

The type of object the uid is associated with can be determined by calling `get_uid_base_type( )`.

DI-Guy Author will internally decide what the uid code means, and set its selected object pointers appropriately. The `get_selected_*` functions below (e.g., `get_selected_character( )`) will return objects that are selected as a result of this call. Calling this function will un-select all other objects, even if they are of different types.

Note, however, that in some cases multiple objects may become selected, if the uid of the object is for an object owned by another. For example, selecting a waypoint will also select the waypoint's path, and the path's character.

Arguments:

<code>uid</code>	unique identifier of selected object

Returns:

1 if uid is valid, 0 if not (e.g., no such object)

5.3.16 function**diguyAuthorInterface::set_selected_diguy_uid_from_string****Prototype:**

```
int set_selected_diguy_uid_from_string(const char* uid_string);
```

Description:

Same as `set_selected_diguy_uid()`, but with uid in string form.

5.3.17 function diguyAuthorInterface::get_uid_base_type**Prototype:**

```
int get_uid_base_type(long uid);
```

Description:

Returns a value that describes the object type associated with the passed uid. The returned value will be one of the `DIGUY_TYPE_UID_BASE_*` values listed in `diguy_constants.h`; e.g., a return value of `DIGUY_TYPE_UID_BASE_CHARACTER` means that the uid is associated with a `diguyCharacter` object.

`DIGUY_TYPE_UID_BASE_NOthing` is returned if the passed value is not a valid DI-Guy uid.

Arguments:

<code>uid</code>	unique identifier to be checked

Returns:

uid base type enumeration

5.3.18 function diguyAuthorInterface::get_uid_base_type_from_string

Prototype:

```
int get_uid_base_type_from_string(const char* uid_string);
```

Description:

Same as [get_uid_base_type\(\)](#), but with uid in string form.

5.3.19 function diguyAuthorInterface::get_selected_character

Prototype:

```
diguyCharacter* get_selected_character();
```

Description:

Returns the character that was selected as a result of calling `set_selected_diguy_uid()`.

Returns:

pointer to type `diguyCharacter`

5.3.20 function diguyAuthorInterface::get_selected_character_path

Prototype:

```
diguyCharacterPath* get_selected_character_path();
```

Description:

`diguyCharacterPath` version of `get_selected_character()`.

5.3.21 function diguyAuthorInterface::get_selected_waypoint

Prototype:

```
diguyWaypoint* get_selected_waypoint();
```

Description:

`diguyWaypoint` version of `get_selected_character()`.

5.3.22 function diguyAuthorInterface::get_selected_path_shape

Prototype:

```
diguyPathShape* get_selected_path_shape();
```

Description:

`diguyPathShape` version of `get_selected_character()`.

5.3.23 function diguyAuthorInterface::get_selected_link

Prototype:

```
diguyGraphicsLink* get_selected_link();
```

Description:

`diguyGraphicsLink` version of `get_selected_character()`.

5.3.24 function diguyAuthorInterface::get_selected_shape

Prototype:

```
diguyGraphicsShape* get_selected_shape();
```

Description:

`diguyGraphicsShape` version of `get_selected_character()`.

Terrain and Intersection Virtual Functions

The functions in this section are used by DI-Guy Author to query for information from the world. They are virtual functions that should be overridden in the Host IG to provide the requested information.

World coordinates are in DI-Guy's standard coordinate system that has Z up.

Screen coordinates (also sometimes referred to as window coordinates) are in pixels, where:

- ◆ x is the horizontal pixel from left side of window
- ◆ y is the vertical pixel from bottom of window
- ◆ z is the depth of pixel in the Z buffer: 0 at near plane, 1 at far plane

5.3.25 function diguyAuthorInterface::convert_screen_to_world

Prototype:

```
virtual int convert_screen_to_world(const diguyVec3f&screen_xyz,
    diguyVec3f* world_xyz) = 0;
```

Description:

This function will be called when DI-Guy Author needs to know the world position behind the given screen coordinate.

See above for information on the world and screen coordinate systems.

A typical implementation will construct a ray with its beginning at the world coordinate corresponding to the screen coordinate, projected some distance into the world using a camera projection matrix.

Arguments:

screen_xyz screen coordinates as described above
world_xyz return value world intersection point

Returns:

1 if an intersection occurred, 0 if not

5.3.26 function diguyAuthorInterface::convert_world_to_screen

Prototype:

```
virtual int convert_world_to_screen(const diguyVec3f&world_xyz,
    diguyVec3f* screen_xyz) = 0;
```

Description:

This function is the inverse of [convert_screen_to_world\(\)](#). It takes a world position and returns the corresponding screen coordinate.

Arguments:

`world_xyz` point in world coordinates

`screen_xyz` return value screen coordinate that covers that point

Returns:

1 if an intersection occurred, 0 if not

5.3.27 function diguyAuthorInterface::get_altitude**Prototype:**

```
virtual float get_altitude(float x, float y, float old_z,  
    float from_height = 10.0f,  
    int* valid_result = NULL) = 0;
```

5.3.28 function diguyAuthorInterface::update_selected_objects**Prototype:**

```
virtual int update_selected_objects(const diguyVec3f&screen_xyz) = 0;
```

5.3.29 function diguyAuthorInterface::check_z_clearance

Prototype:

```
virtual int check_z_clearance(const diguyVec3f&pos,  
    const float radius,  
    const float ground_angle,  
    float z_clear_height,  
    int check_below = 1) const = 0;
```

Author UI Functions

The functions in this section are for interactions with the 2D user interface side of DI-Guy Author, as implemented in the application DIGuyAuthorUI. These functions can be called from the Host IG.

Because the Author UI is in a separate process there are a number of communication details that these functions will help with.

5.3.30 function

diguyAuthorInterface::set_author_ui_window_visible

Prototype:

```
int set_author_ui_window_visible(const char* window_name, int visible);
```

Description:

Calling this function will tell the DI-Guy Author UI to show or hide the specified window.

Arguments:

`window_name` name of the window to show or hide
`visible` pass 1 to show the window, 0 to hide it

Currently valid window names are:

- ◇ "input_mode"
- ◇ "time_control"
- ◇ "log"
- ◇ "elements"

5.3.31 function

diguyAuthorInterface::get_author_ui_window_visible

Prototype:

```
int get_author_ui_window_visible(const char* window_name);
```

Returns:

1 if the specified DI-Guy Author UI window is shown, 0 if not

Arguments:

`window_name` name of the window to show or hide

5.3.32 function**diguyAuthorInterface::set_author_ui_panel_visible****Prototype:**

```
int set_author_ui_panel_visible(const char* panel_name);
```

Description:

Calling this function will tell the DI-Guy Author UI to show the specified panel in the elements window.

Arguments:

`panel_name` name of the window to show or hide

Currently valid window names are:

◇ "Character"

◇ "Crowd"

5.3.33 function**diguyAuthorInterface::set_author_ui_select_crowd_character****Prototype:**

```
int set_author_ui_select_crowd_character(diguyCharacter* character);
```

Description:

Calling this function will tell the DI-Guy Author UI to select the specified character in the crowd page.

Arguments:

`character` name of the character to select in the crowd window

5.3.34 function**diguyAuthorInterface::set_author_ui_select_code_browser_script****Prototype:**

```
int set_author_ui_select_code_browser_script(diguyCharacter* character);
```

Description:

Calling this function will tell the DI-Guy Author UI to select the specified script for the character.

Arguments:

`character` name of the character to select in the crowd window

Author UI Status Update Functions

The functions in this section will be called by DI-Guy Author when the status of various parts of the DI-Guy Author UI application change. They are virtual functions that the Host IG should override. This gives the Host IG a chance to update its state on what is happening with the Author UI, such as enabling or disabling various native user interface controls.

5.3.35 function**diguyAuthorInterface::author_ui_window_visibility_changed****Prototype:**

```
virtual void author_ui_window_visibility_changed(const char* window_name,
        int is_visible);
```

Description:

Called if the named window in the Author UI application changes visibility state.

Utility Vertex Generation Functions

The functions in this section are utility functions for generating vertices that can be used for rendering 3D DI-Guy Author visuals in the Host IG. These functions will typically be called from diguyAuthorVisual subclasses.

For example, in DI-Guy Scenario and the DI-Guy Author Host IG reference implementation, path waypoints are shown as line segments with side wings and a vertical post, and a cone at the end of the forward facing segment representing an arrowhead. The subclass that creates the Host IG native 3D visuals can call these functions to get the recommended vertex information with which native 3D objects can be constructed. In the case of the [diguyAuthorVisualWaypoint](#) subclass, multiple calls to [generate_box_vertices\(\)](#) can be made to generate vertices for the various posts, and one call to [generate_cone_vertices\(\)](#) can be made to generate the vertices for the arrowhead.

The DI-Guy Author Host IG reference implementation that is based on Open Scene Graph provides very clear examples of how these functions can be used.

Returned pointers must be deleted by calling [delete_generated_vertices\(\)](#).

Vertices are returned in [diguyGraphicsVertexFormat](#)
[DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2](#).

5.3.36 function diguyAuthorInterface::generate_sphere_vertices

Prototype:

```
static int generate_sphere_vertices(float radius,
    int num_slices,
    int num_layers,
    const diguyVec3f&position_offset,
    const diguyVec3f&orientation,          // degrees
    int* num_vertices,
    diguyVec3f** vertices,
    diguyVec3f** normals,
    diguyVec2f** texture_indices,
    int indices_for_wireframe,
    int* num_indices,
    unsigned int** indices);
```

Description:

Generates vertices for a sphere with the passed radius. The data returned in the vertices, normals, texture_indices, and indices pointers must be deleted using [delete_generated_vertices\(\)](#).

Untransformed, the sphere's center will be at the local origin.

DI-Guy SDK Version 12.5.2 Index

The slices will radiate around the z axis. The stacks will be along the z axis.

Arguments:

<code>radius</code>	(input) radius of sphere, in meters
<code>num_slices</code>	(input) number of radial slices; think slices in a cake
<code>num_layers</code>	(input) number of vertical layers; think layers in a cake
<code>position_offset</code>	(input) position offset of sphere from local origin, in meters
<code>orientation</code>	(input) orientation offset of sphere about local origin, in degrees
<code>num_vertices</code>	(output) number of generated vertices
<code>vertices</code>	(output) vertex data
<code>normals</code>	(output) normal data
<code>texture_indices</code>	(output) texture index data (not currently used)
<code>indices_for_wireframe</code>	(input) pass 1 if vertices are for a wireframe object that will be represented by line segments, 0 if the vertices are for a solid object represented by polygons
<code>num_indices</code>	(output) number of indices telling how vertices connect
<code>indices</code>	(output) data on how vertices connect

Returns:

0 on success, -1 on error

5.3.37 function diguyAuthorInterface::generate_cone_vertices**Prototype:**

```
static int generate_cone_vertices(float radius,
    float height,
    int num_slices,
    int num_layers,
    const diguyVec3f&position_offset,
    const diguyVec3f&orientation,      // degrees
    int* num_vertices,
    diguyVec3f** vertices,
    diguyVec3f** normals,
    diguyVec2f** texture_indices,
    int indices_for_wireframe,
    int* num_indices,
    unsigned int** indices);
```

Description:

Similar to [generate_sphere_vertices\(\)](#), but for cones.

Untransformed, the cone's base will be at the local origin, with the point tapering in positive z.

The slices will radiate around the z axis. The stacks will be along the z axis.

5.3.38 function diguyAuthorInterface::generate_cylinder_vertices**Prototype:**

```
static int generate_cylinder_vertices(float radius,
    float height,
    int num_slices,
    int num_layers,
    const diguyVec3f&position_offset,
    const diguyVec3f&orientation,      // degrees
    int* num_vertices,
    diguyVec3f** vertices,
    diguyVec3f** normals,
    diguyVec2f** texture_indices,
    int indices_for_wireframe,
    int* num_indices,
    unsigned int** indices);
```

Description:

Similar to [generate_sphere_vertices\(\)](#), but for cylinders.

Untransformed, the cylinder's base will be at the local origin, the top above the base in positive z.

The slices will radiate around the z axis. The stacks will be along the z axis.

5.3.39 function diguyAuthorInterface::generate_box_vertices

Prototype:

```
static int generate_box_vertices(diguyVec3f min,
                                diguyVec3f max,
                                const diguyVec3f&position_offset,
                                const diguyVec3f&orientation,      // degrees
                                int* num_vertices,
                                diguyVec3f** vertices,
                                diguyVec3f** normals,
                                diguyVec2f** texture_indices,
                                int indices_for_wireframe,
                                int* num_indices,
                                unsigned int** indices);
```

Description:

Similar to [generate_sphere_vertices\(\)](#), but for boxes, AKA rectangular solids.

Untransformed, the box's corners are represented by the passed min and max vectors, relative to the local origin.

5.3.40 function diguyAuthorInterface::delete_generated_vertices**Prototype:**

```
static void delete_generated_vertices(diguyVec3f* vertices,  
    diguyVec3f* normals,  
    diguyVec2f* texture_indices,  
    unsigned int* indices);
```

Description:

For deleting data generated by the various generate_*_vertices() functions.

5.4 diguyAuthorVisualMaterial

diguyAuthorVisualMaterial.h

DI-Guy SDK Version 12.5.2

Includes: [declspec_diguy.h](#) | [diguy_vector_classes.h](#)

Contents:

- ◆ [Polygon Material Functions](#)
- ◆ [Edge / Line Segment Material Functions](#)

Summary:

The [diguyAuthorVisualMaterial](#) class provides data needed to render DI-Guy Author visuals in a Host IG.

5.4.1 class diguyAuthorVisualMaterial

```
class BDI_DECLSPEC_diguy diguyAuthorVisualMaterial
{
public:
    const char* get_name();
    diguyVec4f get_brush_color();
    int get_num_layers(); // number "vertical" divisions
    int get_num_slices();
    int get_fill_faces();
    diguyVec4f get_pen_color();
    diguyVec4f get_pen2_color();
    float get_line_width();
    int get_edge_faces();
```

5.4.2 function diguyAuthorVisualMaterial::get_name

Prototype:

```
const char* get_name();
```

Returns the name of this material.

Polygon Material Functions

5.4.3 function diguyAuthorVisualMaterial::get_brush_color

Prototype:

```
diguyVec4f get_brush_color();
```

Returns the material's primary brush color. The brush color is used to fill the polygons of visuals.

The returned vector is an RGBA color vector, with color channel values between 0 and 1.

5.4.4 function diguyAuthorVisualMaterial::get_num_layers

Prototype:

```
int get_num_layers(); // number "vertical" divisions
```

Returns the suggested number of "layers", or vertical divisions, that should be used to construct polygonal solids such as spheres, cones, cylinders, and other shapes that need linear approximations of circular data.

Analogous to layers in a round cake.

5.4.5 function diguyAuthorVisualMaterial::get_num_slices

Prototype:

```
int get_num_slices();
```

Returns the suggested number of "slices", or radial divisions, that should be used to construct polygonal solids such as spheres, cones, cylinders, and other shapes that need linear approximations of circular data.

Analogous to slices in a round cake.

5.4.6 function diguyAuthorVisualMaterial::get_fill_faces

Prototype:

```
int get_fill_faces();
```

Returns 1 if faces in should be visible, 0 if not. This is the same as checking whether the brush color alpha component equals 0 (complete transparency).

Shortcut for checking brush alpha not equal to 0.

Edge / Line Segment Material Functions

5.4.7 function diguyAuthorVisualMaterial::get_pen_color

Prototype:

```
diguyVec4f get_pen_color();
```

Returns the material's primary pen color. The pen color is used for line segments and edges of visuals.

The returned vector is an RGBA color vector, with color channel values between 0 and 1.

5.4.8 function diguyAuthorVisualMaterial::get_pen2_color

Prototype:

```
diguyVec4f get_pen2_color();
```

Returns the material's secondary pen color. The secondary pen isn't used by a visual unless the visual's documentation explicitly says it is.

5.4.9 function diguyAuthorVisualMaterial::get_line_width

Prototype:

```
float get_line_width();
```

Returns the recommended width of line segments.

5.4.10 function diguyAuthorVisualMaterial::get_edge_faces**Prototype:**

```
int get_edge_faces();
```

Returns 1 if edge line segments should be visible, 0 if not. This is the same as checking whether the pen color alpha component equals 0 (complete transparency).

5.5 diguyAuthorVisualMultiline

diguyAuthorVisualMultiline.h

DI-Guy SDK Version 12.5.2

Includes: `declspec_diguy.h` | [diguy_vector_classes.h](#)

Contents:

◆ [Virtual Functions](#)

Link against: `libdiguy`

Summary:

The [diguyAuthorVisualMultiline](#) class provides an interface for creating supplemental visuals needed for creating DI-Guy scenarios in a Host App.

Multiline visuals are used for representing the spline curve of character paths.

5.5.1 `class diguyAuthorVisualMultiline`

Function for setting the create function for multiline visual subclass objects. This should be called after [diguy_author_initialize\(\)](#), but before any scenario is created or loaded.

```
class BDI_DECLSPEC_diguy diguyAuthorVisualMultiline
{
public:
    diguyAuthorVisualMaterial* get_visual_material();
    int get_num_points();
    diguyVec3f get_point_position(int point_number);
    diguyVec3d get_point_position_double(int point_number);
    long get_uid();
    int get_visible();
    virtual void build();
    virtual void unbuild();
    virtual void update(int update_flags);
    virtual void draw();
    virtual void show();
    virtual void hide();
};
```

5.5.2 function diguyAuthorVisualMultiline::get_visual_material

Prototype:

```
diguyAuthorVisualMaterial* get_visual_material();
```

Returns the visual material to be used for rendering this object.

Because multilines have no polygons the brush color of the material is not used.

5.5.3 function diguyAuthorVisualMultiline::get_num_points

Prototype:

```
int get_num_points();
```

5.5.4 function diguyAuthorVisualMultiline::get_point_position

Prototype:

```
diguyVec3f get_point_position(int point_number);
```

5.5.5 function diguyAuthorVisualMultiline::get_point_position_double

Prototype:

```
diguyVec3d get_point_position_double(int point_number);
```

5.5.6 function diguyAuthorVisualMultiline::get_uid

Prototype:

```
long get_uid();
```

Returns the uid (unique identifier) of this multiline.

5.5.7 function diguyAuthorVisualMultiline::get_visible

Prototype:

```
int get_visible();
```

Returns 1 if this visual should be visible, else 0.

Virtual Functions

The functions in this section are all virtual functions that will be called by DI-Guy Author at various times during DI-Guy execution.

Because the functions are all virtual, DI-Guy programmers can create a subclass of this class and override them. Unless otherwise stated, the base class functions do not need to be called.

5.5.8 function `diguyAuthorVisualMultiline::build`

Prototype:

```
virtual void build();
```

Description:

See `diguyAuthorVisualWaypoint::build()` for general information about this function.

Multilines are mostly represented by renderer line primitives.

5.5.9 function diguyAuthorVisualMultiline::unbuild

Prototype:

```
virtual void unbuild();
```

Description:

See [diguyAuthorVisualWaypoint::unbuild\(\)](#) for general information about this function.

5.5.10 function diguyAuthorVisualMultiline::update

Prototype:

```
virtual void update(int update_flags);
```

Description:

See [diguyAuthorVisualWaypoint::update\(\)](#) for general information about this function.

DIGUY_GRAPHICS_VISUAL_UPDATE_GEOMETRY

The functions [get_num_points\(\)](#), [get_point_position\(\)](#), etc., should be used to create line primitives.

DIGUY_GRAPHICS_VISUAL_UPDATE_POSITION

Multiline visuals do not use this value. The overall positions of line segment endpoints that make up the multiline are recomputed and applied during an overall geometry update operation.

5.5.11 function diguyAuthorVisualMultiline::draw

Prototype:

```
virtual void draw();
```

Description:

See [diguyAuthorVisualWaypoint::draw\(\)](#) for general information about this function.

5.5.12 function diguyAuthorVisualMultiline::show

Prototype:

```
virtual void show();
```

Description:

See [diguyAuthorVisualWaypoint::show\(\)](#) for general information about this function.

5.5.13 function diguyAuthorVisualMultiline::hide

Prototype:

```
virtual void hide();
```

Description:

See [diguyAuthorVisualWaypoint::hide\(\)](#) for general information about this function.

5.6 diguyAuthorVisualObjectHandle

diguyAuthorVisualObjectHandle.h

DI-Guy SDK Version 12.5.2

Includes: [declspec_diguy.h](#) | [diguy_typedefs.h](#) | [diguy_vector_classes.h](#)

Contents:

◆ [Virtual Functions](#)

Link against: `libdiguy`

Summary:

The [diguyAuthorVisualObjectHandle](#) class provides an interface for creating supplemental visuals needed for creating DI-Guy scenarios in a Host IG.

Object handle visuals are used for simple visuals such as beads on character paths, selection handles, etc.

5.6.1 `class diguyAuthorVisualObjectHandle`

Function for setting the create function for object handle visual subclass objects. This should be called after [diguy_author_initialize\(\)](#), but before any scenario is created or loaded.

```
class BDI_DECLSPEC_diguy diguyAuthorVisualObjectHandle
{
public:
    diguyAuthorVisualMaterial* get_visual_material();
    enum {
        HANDLE_SHAPE_CYLINDER = 0,
        HANDLE_SHAPE_CONE,
        HANDLE_SHAPE_SPHERE,
        HANDLE_SHAPE_PATH_BEAD,
        HANDLE_SHAPE_PATH_STOP_SIGN
    };
    int get_handle_shape() {return m_handle_shape;}
    float get_radius();
    float get_height();
    float get_bead_depth();
    float get_bead_arm_thickness();
    int get_inverted();
    diguyVec3f get_position();
    diguyVec3d get_position_double();
    diguyVec3f get_orientation();
    long get_uid();
    int get_visible();
    virtual void build();
    virtual void unbuild();
};
```

```
virtual void update(int update_flags);
virtual void draw();
virtual void show();
virtual void hide();
```

5.6.2 function diguyAuthorVisualObjectHandle::get_visual_material

Prototype:

```
diguyAuthorVisualMaterial* get_visual_material();
```

Returns the visual material to be used for rendering this object.

5.6.3 enumeration

```
enum {
    HANDLE_SHAPE_CYLINDER = 0,
    HANDLE_SHAPE_CONE,
    HANDLE_SHAPE_SPHERE,
    HANDLE_SHAPE_PATH_BEAD,
    HANDLE_SHAPE_PATH_STOP_SIGN
};
```

The shapes the handle can have.

HANDLE_SHAPE_CYLINDER

A cylinder oriented along the Z axis, with its base at the origin.

HANDLE_SHAPE_CONE

A cone oriented along the Z axis, with its base at the origin. If `get_inverted()` returns 1, the point will be at the origin.

HANDLE_SHAPE_SPHERE

A sphere, centered at the origin.

HANDLE_SHAPE_PATH_BEAD

Usually a plus sign with depth, oriented so it can rotate around the X axis.

HANDLE_SHAPE_PATH_STOP_SIGN

Usually an octagon with depth, oriented so it can rotate around the X axis.

5.6.4 function `diguyAuthorVisualObjectHandle::get_handle_shape`

Prototype:

```
int get_handle_shape();
```

Returns the handle shape. This will be one of the enum values above.

5.6.5 function diguyAuthorVisualObjectHandle::get_radius

Prototype:

```
float get_radius();
```

Returns the handle radius, for those shapes that have one, including:

- ◇ cylinder
- ◇ cone
- ◇ sphere
- ◇ path_bead
- ◇ path_stop_sign

5.6.6 function diguyAuthorVisualObjectHandle::get_height

Prototype:

```
float get_height();
```

Returns the handle height, for those shapes that have one, including:

- ◇ cylinder
- ◇ cone

5.6.7 function diguyAuthorVisualObjectHandle::get_bead_depth**Prototype:**

```
float get_bead_depth();
```

Returns the bead depth along the X axis, for those shapes that have one, including:

- ◇ path_bead
- ◇ path_stop_sign

Returns the bead thickness, for those shapes that have one, including:

- ◆ path_bead

5.6.8 function diguyAuthorVisualObjectHandle::get_bead_arm_thickness**Prototype:**

```
float get_bead_arm_thickness();
```

5.6.9 function diguyAuthorVisualObjectHandle::get_inverted

Prototype:

```
int get_inverted();
```

Returns whether the handle shape should be inverted. Valid for:

◇ cone

5.6.10 function diguyAuthorVisualObjectHandle::get_position

Prototype:

```
diguyVec3f get_position();
```

Returns the position, in meters from the origin, of the object handle in the *Boston Dynamics* global coordinate system. See

[diguyCharacter::set_position\(\)](#) for a description of the coordinate system.

5.6.11 function diguyAuthorVisualObjectHandle::get_position_double

Prototype:

```
diguyVec3d get_position_double();
```

Returns the high precision position of the object handle.

5.6.12 function diguyAuthorVisualObjectHandle::get_orientation

Prototype:

```
diguyVec3f get_orientation();
```

Returns the orientation, in degrees, of the object handle in the *Boston Dynamics* global coordinate system. See

[diguyCharacter::set_position\(\)](#) for a description of the coordinate system.

5.6.13 function diguyAuthorVisualObjectHandle::get_uid**Prototype:**

```
long get_uid();
```

Returns the uid (unique identifier) of this object handle.

5.6.14 function diguyAuthorVisualObjectHandle::get_visible**Prototype:**

```
int get_visible();
```

Returns 1 if this visual should be visible, else 0.

Virtual Functions

The functions in this section are all virtual functions that will be called by DI-Guy Author at various times during DI-Guy execution.

Because the functions are all virtual, DI-Guy programmers can create a subclass of this class and override them. Unless otherwise stated, the base class functions do not need to be called.

5.6.15 function `diguyAuthorVisualObjectHandle::build`

Prototype:

```
virtual void build();
```

Description:

See `diguyAuthorVisualWaypoint::build()` for general information about this function.

5.6.16 function diguyAuthorVisualObjectHandle::unbuild

Prototype:

```
virtual void unbuild();
```

Description:

See [diguyAuthorVisualWaypoint::unbuild\(\)](#) for general information about this function.

5.6.17 function diguyAuthorVisualObjectHandle::update

Prototype:

```
virtual void update(int update_flags);
```

Description:

See [diguyAuthorVisualWaypoint::update\(\)](#) for general information about this function.

DIGUY_GRAPHICS_VISUAL_UPDATE_GEOMETRY

Object handles have different shapes based on the handle shape returned by [get_handle_shape\(\)](#).

5.6.18 function diguyAuthorVisualObjectHandle::draw

Prototype:

```
virtual void draw();
```

Description:

See [diguyAuthorVisualWaypoint::draw\(\)](#) for general information about this function.

5.6.19 function diguyAuthorVisualObjectHandle::show

Prototype:

```
virtual void show();
```

Description:

See [diguyAuthorVisualWaypoint::show\(\)](#) for general information about this function.

5.6.20 function `diguyAuthorVisualObjectHandle::hide`

Prototype:

```
virtual void hide();
```

Description:

See [diguyAuthorVisualWaypoint::hide\(\)](#) for general information about this function.

5.7 diguyAuthorVisualRegionMesh

diguyAuthorVisualRegionMesh.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

◆ Virtual Functions

Link against: libdiguy

Summary:

The [diguyAuthorVisualRegionMesh](#) class provides an interface for creating supplemental visuals needed for creating DI-Guy scenarios in a Host IG.

Region mesh visuals are used to visualize regions and navmeshes.

5.7.1 class diguyAuthorVisualRegionMesh

Function for setting the create function for region mesh visual subclass objects. This should be called after [diguy_author_initialize\(\)](#), but before any scenario is created or loaded.

```
class BDI_DECLSPEC_diguy diguyAuthorVisualRegionMesh
{
public:
    diguyAuthorVisualMaterial* get_visual_material();
    diguyRegion* get_region();
    int get_subregion_index();
    int get_num_quads();
    diguyVec3f get_quad_point(int quad, int corner);
    diguyVec3f* get_quads_data_ptr();
    int get_num_edges();
    diguyVec3f get_edge_point(int edge, int end);
    diguyVec3f* get_edges_data_ptr();
    diguyVec3f get_mesh_origin();
    int get_highlight_level();
    int get_visible();
    virtual void build();
    virtual void unbuild();
    virtual void update(int update_flags);
    virtual void draw();
    virtual void show();
    virtual void hide();
};
```

5.7.2 function diguyAuthorVisualRegionMesh::get_visual_material

Prototype:

```
diguyAuthorVisualMaterial* get_visual_material();
```

Returns the visual material to be used for rendering this object.

5.7.3 function diguyAuthorVisualRegionMesh::get_region

Prototype:

```
diguyRegion* get_region();
```

Returns the [diguyRegion](#) object that this visual is associated with. Note that one [diguyRegion](#) will most likely have multiple region mesh visuals.

5.7.4 function `diguyAuthorVisualRegionMesh::get_subregion_index`

Prototype:

```
int get_subregion_index();
```

Returns the index of the subregion of the associated `diguyRegion` that this visual is part of. This value will be one of the `diguySubregionIndex` enumerated values.

5.7.5 function `diguyAuthorVisualRegionMesh::get_num_quads`

Prototype:

```
int get_num_quads();
```

Returns the number of quad polygons to be rendered by this visual. They should be rendered using the material's brush color.

5.7.6 function diguyAuthorVisualRegionMesh::get_quad_point

Prototype:

```
diguyVec3f get_quad_point(int quad, int corner);
```

Returns one corner of one of the visual's quads. Corner argument should be between 0 and 3, quad argument between 0 and the result from `get_num_quads()` - 1.

The returned vertex will be relative to the region's origin as returned by `get_mesh_origin()`.

Note: this will be slow. Use `get_quads_data_ptr()` if possible.

5.7.7 function diguyAuthorVisualRegionMesh::get_quads_data_ptr

Prototype:

```
diguyVec3f* get_quads_data_ptr();
```

Returns a pointer to a contiguous data buffer that contains all quad polygon data for this visual. There will be four times the value returned by `get_num_quads()` vectors in the data buffer.

The returned vertices will be relative to the region's origin as returned by `get_mesh_origin()`.

5.7.8 function diguyAuthorVisualRegionMesh::get_num_edges

Prototype:

```
int get_num_edges();
```

Returns the number of line segment edges to be rendered by this visual. They should be rendered using the material's pen color.

5.7.9 function diguyAuthorVisualRegionMesh::get_edge_point

Prototype:

```
diguyVec3f get_edge_point(int edge, int end);
```

Returns one endpoint of one of the visual's edges. End argument should be 0 or 1, edge argument between 0 and the result from `get_num_edges()` - 1.

The returned vertex will be relative to the region's origin as returned by `get_mesh_origin()`.

Note: this will be slow. Use `get_edges_data_ptr()` if possible.

5.7.10 function**diguyAuthorVisualRegionMesh::get_edges_data_ptr****Prototype:**

```
diguyVec3f* get_edges_data_ptr();
```

Returns a pointer to a contiguous data buffer that contains all line segment edge data for this visual. There will be two times the value returned by `get_num_edges()` vectors in the data buffer.

The returned vertices will be relative to the region's origin as returned by `get_mesh_origin()`.

5.7.11 function diguyAuthorVisualRegionMesh::get_mesh_origin**Prototype:**

```
diguyVec3f get_mesh_origin();
```

Returns the origin of the geometry in the visual. All quads and edges will be relative to this origin.

5.7.12 function `diguyAuthorVisualRegionMesh::get_highlight_level`

Prototype:

```
int get_highlight_level();
```

Returns the level at which this mesh should be highlighted. This will be 0, 1, or 2. The higher the highlight level, the brighter the mesh should appear.

It's recommended that for highlight level 0, the colors of the visual material be scaled by a factor of 0.6; for highlight level 1, by a factor of 0.8; and for highlight level 2 don't scale.

Highlight level 2 should have a wider line width than levels 0 or 1.

5.7.13 function `diguyAuthorVisualRegionMesh::get_visible`

Prototype:

```
int get_visible();
```

Returns 1 if this visual should be visible, else 0.

Virtual Functions

The functions in this section are all virtual functions that will be called by DI-Guy Author at various times during DI-Guy execution.

Because the functions are all virtual, DI-Guy programmers can create a subclass of this class and override them. Unless otherwise stated, the base class functions do not need to be called.

5.7.14 function diguyAuthorVisualRegionMesh::build

Prototype:

```
virtual void build();
```

Description:

`build( )` will be called by DI-Guy when it is time for a renderer-specific visual object to be created. All information necessary to build the visual should be available via the Accessor Functions above when this function is called.

It is common for `build( )` and `update( )` to use the same internal, user-written function for creating geometry.

Immediate Mode:

Immediate mode renderers may override this function to create an object, for example a vertex array, that may be used when `draw( )` is called.

Scene Graph:

Scene graph renderers almost always override this function to create a scene graph node object that is attached to the scene graph and updated when `update()` is called.

5.7.15 function `diguyAuthorVisualRegionMesh::unbuild`

Prototype:

```
virtual void unbuild();
```

Description:

See `diguyAuthorVisualWaypoint::unbuild()` for general information about this function.

5.7.16 function diguyAuthorVisualRegionMesh::update

Prototype:

```
virtual void update(int update_flags);
```

Description:

See [diguyAuthorVisualWaypoint::update\(\)](#) for general information about this function. Specifics for this visual type are below.

DIGUY_GRAPHICS_VISUAL_UPDATE_GEOMETRY

Region mesh visuals need to use calls to [get_num_quads\(\)](#), [get_quads_data_ptr\(\)](#), etc. to determine geometry.

DIGUY_GRAPHICS_VISUAL_UPDATE_POSITION

Region mesh visuals do not use this value. The overall positions of polygons that make up the mesh are recomputed and applied during an overall geometry update operation.

5.7.17 function diguyAuthorVisualRegionMesh::draw

Prototype:

```
virtual void draw();
```

Description:

See [diguyAuthorVisualWaypoint::draw\(\)](#) for general information about this function.

5.7.18 function diguyAuthorVisualRegionMesh::show

Prototype:

```
virtual void show();
```

Description:

See [diguyAuthorVisualWaypoint::show\(\)](#) for general information about this function.

5.7.19 function diguyAuthorVisualRegionMesh::hide

Prototype:

```
virtual void hide();
```

Description:

See [diguyAuthorVisualWaypoint::hide\(\)](#) for general information about this function.

5.8 diguyAuthorVisualRegionPaintbrush

diguyAuthorVisualRegionPaintbrush.h

DI-Guy SDK Version 12.5.2

Includes: `declspec_diguy.h` | [diguy_constants.h](#)

Contents:

◆ Virtual Functions

Link against: `libdiguy`

Summary:

The `diguyAuthorVisualRegionPaintbrush` class provides an interface for creating supplemental visuals needed for creating DI-Guy scenarios in a Host IG.

Region paintbrush visuals are used to visualize the 3D paintbrush used to show where painted regions and meshes will appear.

Paintbrushes are commonly shown as translucent spheres when they will add to the region, or a yellow pencil-like column when they will erase from the region.

5.8.1 class diguyAuthorVisualRegionPaintbrush

Function for setting the create function for region paintbrush subclass objects. This should be called after `diguy_author_initialize()`, but before any scenario is created or loaded.

```
class BDI_DECLSPEC_diguy diguyAuthorVisualRegionPaintbrush
{
public:
    diguyAuthorVisualMaterial* get_visual_material();
    diguyVec3f get_position();
    diguySubregionIndex get_painting_subregion();
    void set_painting_subregion(diguySubregionIndex subregion_index);
    diguyRegionPaintbrushMode get_paintrush_mode();
    void set_paintrush_mode(diguyRegionPaintbrushMode mode);
    float get_brush_size();
    void set_brush_size(float radius);
    int get_brush_is_eraser();
    int get_visible();
    virtual void build();
    virtual void unbuild();
    virtual void update(int update_flags);
    virtual void draw();
    virtual void show();
    virtual void hide();
};
```


5.8.2 function diguyAuthorVisualRegionPaintbrush::get_visual_material

Prototype:

```
diguyAuthorVisualMaterial* get_visual_material();
```

Returns the visual material to be used for rendering this object.

5.8.3 function diguyAuthorVisualRegionPaintbrush::get_position

Prototype:

```
diguyVec3f get_position();
```

Returns the current position of the paintbrush in meters from the DI-Guy origin.

5.8.4 function

diguyAuthorVisualRegionPaintbrush::get_painting_subregion

Prototype:

```
diguySubregionIndex get_painting_subregion();
```

Returns the index of the subregion the paintbrush will paint into. This will be a [diguySubregionIndex](#) enumeration value.

5.8.5 function

diguyAuthorVisualRegionPaintbrush::set_painting_subregion

Prototype:

```
void set_painting_subregion(diguySubregionIndex subregion_index);
```

Sets the index of the subregion the paintbrush will paint into.

5.8.6 function `diguyAuthorVisualRegionPaintbrush::get_paintrush_mode`

Prototype:

```
diguyRegionPaintbrushMode get_paintrush_mode();
```

Returns the current paintbrush mode, a `diguyRegionPaintbrushMode` enumeration value.

5.8.7 function `diguyAuthorVisualRegionPaintbrush::set_paintrush_mode`

Prototype:

```
void set_paintrush_mode(diguyRegionPaintbrushMode mode);
```

Sets what happens when a paintbrush stroke is applied in input modes `DIGUY_SCENARIO_INPUT_MODE_REGION_PAINTER` and `DIGUY_SCENARIO_INPUT_MODE_CROWD_EDITOR`.

5.8.8 function diguyAuthorVisualRegionPaintbrush::get_brush_size

Prototype:

```
float get_brush_size();
```

Returns the radius of the paintbrush.

5.8.9 function diguyAuthorVisualRegionPaintbrush::set_brush_size

Prototype:

```
void set_brush_size(float radius);
```

Sets the radius of the paintbrush.

5.8.10 function diguyAuthorVisualRegionPaintbrush::get_brush_is_eraser

Prototype:

```
int get_brush_is_eraser();
```

Returns 1 if the paintbrush should be shown as an eraser, else 0.

5.8.11 function diguyAuthorVisualRegionPaintbrush::get_visible

Prototype:

```
int get_visible();
```

Returns 1 if this visual should be visible, else 0.

Virtual Functions

The functions in this section are all virtual functions that will be called by DI-Guy Author at various times during DI-Guy execution. Most visuals have a similar set of virtual functions.

Because the functions are all virtual, DI-Guy programmers can create a subclass of this class and override them. Unless otherwise stated, the base class functions do not need to be called.

5.8.12 function `diguyAuthorVisualRegionPaintbrush::build`

Prototype:

```
virtual void build();
```

Description:

See `diguyAuthorVisualWaypoint::build()` for general information about this function.

5.8.13 function diguyAuthorVisualRegionPaintbrush::unbuild

Prototype:

```
virtual void unbuild();
```

Description:

See [diguyAuthorVisualWaypoint::unbuild\(\)](#) for general information about this function.

5.8.14 function diguyAuthorVisualRegionPaintbrush::update

Prototype:

```
virtual void update(int update_flags);
```

Description:

See [diguyAuthorVisualWaypoint::update\(\)](#) for general information about this function.

DIGUY_GRAPHICS_VISUAL_UPDATE_GEOMETRY

Paintbrushes are commonly shown as translucent spheres when they will add to the region, or a yellow pencil-like column when they will erase from the region.

5.8.15 function diguyAuthorVisualRegionPaintbrush::draw

Prototype:

```
virtual void draw();
```

Description:

See [diguyAuthorVisualWaypoint::draw\(\)](#) for general information about this function.

5.8.16 function diguyAuthorVisualRegionPaintbrush::show

Prototype:

```
virtual void show();
```

Description:

See [diguyAuthorVisualWaypoint::show\(\)](#) for general information about this function.

5.8.17 function `diguyAuthorVisualRegionPaintbrush::hide`

Prototype:

```
virtual void hide();
```

Description:

See [diguyAuthorVisualWaypoint::hide\(\)](#) for general information about this function.

5.9 diguyAuthorVisualWaypoint

diguyAuthorVisualWaypoint.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_typedefs.h](#) | [declspec_diguy.h](#)

Contents:

◆ [Virtual Functions](#)

Link against: **libdiguy**

Summary:

The [diguyAuthorVisualWaypoint](#) class provides an interface for creating supplemental visuals needed for creating DI-Guy scenarios in a Host IG.

Waypoint visuals are used for visualizing waypoints on a character path.

5.9.1 **class** diguyAuthorVisualWaypoint

Function for setting the create function for waypoint visual subclass objects. This should be called after [diguy_author_initialize\(\)](#), but before any scenario is created or loaded.

```
class BDI_DECLSPEC_diguy diguyAuthorVisualWaypoint
{
public:
    diguyAuthorVisualMaterial* get_visual_material();
    int get_visible();
    diguyWaypoint* get_waypoint();
    virtual void build();
    virtual void unbuild();
    virtual void update(int update_flags);
    virtual void draw();
    virtual void show();
    virtual void hide();
};
```

5.9.2 function diguyAuthorVisualWaypoint::get_visual_material

Prototype:

```
diguyAuthorVisualMaterial* get_visual_material();
```

Returns the visual material to be used for rendering this object.

5.9.3 function diguyAuthorVisualWaypoint::get_visible

Prototype:

```
int get_visible();
```

Returns 1 if this visual should be visible, else 0.

5.9.4 function diguyAuthorVisualWaypoint::get_waypoint

Prototype:

```
diguyWaypoint* get_waypoint();
```

Returns a pointer to the waypoint this visual is associated with. Most of the information needed to create the visual can be retrieved using [diguyWaypoint](#) function calls.

Virtual Functions

The functions in this section are all virtual functions that will be called by DI-Guy Author at various times during DI-Guy execution.

Because the functions are all virtual, DI-Guy programmers can create a subclass of this class and override them. Unless otherwise stated, the base class functions do not need to be called.

5.9.5 function diguyAuthorVisualWaypoint::build

Prototype:

```
virtual void build();
```

Description:

`build()` will be called by DI-Guy when it is time for a renderer-specific visual object to be created. All information necessary to build the visual should be available via the Accessor Functions above when this function is called.

It is common for `build()` and `update()` to use the same internal, user-written function for creating geometry.

Waypoints are typically represented by relatively complex set of cylinders and cones. The provided reference implementation `diguyAuthorVisualWaypoint` subclass should be consulted for how to create waypoint visual components.

Immediate Mode:

Immediate mode renderers may override this function to create an object, for example a vertex array, that may be used when `draw()` is called.

Scene Graph:

Scene graph renderers almost always override this function to create a scene graph node object that is attached to the scene graph and updated when `update()` is called.

5.9.6 function diguyAuthorVisualWaypoint::unbuild

Prototype:

```
virtual void unbuild();
```

Description:

`unbuild()` will be called by DI-Guy when it is time for a renderer-specific visual object to be destroyed.

In general this function should undo all operations and deallocate all objects created during the `build()` function call.

5.9.7 function diguyAuthorVisualWaypoint::update

Prototype:

```
virtual void update(int update_flags);
```

Description:

`update()` will be called by DI-Guy when this visual has changed state and needs to be updated in some way.

The `update_flags` argument will be a value from the `diguyAuthorVisualUpdateFlag` enumeration:

DIGUY_GRAPHICS_VISUAL_UPDATE_GEOMETRY

The overall geometry of the visual needs to be updated. This means that polygons, lines, and whatever other graphics primitives the visual uses need to be recreated.

DIGUY_GRAPHICS_VISUAL_UPDATE_POSITION

The position and/or orientation of the visual needs to be updated. The overall geometry of the visual (polygons, lines, etc.) remains unchanged, and don't need to be recreated.

If the visual can be relocated via changing the offset of a scene graph node it can save time to not recreate a lot of geometry.

If the visual can't be relocated in such a way, the entire set of geometry used by the visual; essentially, doing the same operation as for [DIGUY_GRAPHICS_VISUAL_UPDATE_GEOMETRY](#).

DIGUY_GRAPHICS_VISUAL_UPDATE_MATERIAL

The material of the visual needs to be updated. This can include color, fill options, line width, etc. The geometry and position of the visual remain the same. The [get_visual_material\(\)](#) function should be called to retrieve the updated materials.

For some renderers it can make more sense to just recreate all of the visual geometry with the updated material settings.

DIGUY_GRAPHICS_VISUAL_UPDATE_ALL

All aspects of the visual need to be updated.

5.9.8 function diguyAuthorVisualWaypoint::draw

Prototype:

```
virtual void draw();
```

Description:

[draw\(\)](#) will be called by DI-Guy when it is time for a renderer-specific visual object to be drawn.

Immediate Mode:

Immediate mode renderers almost always override this function, to draw the visual.

Scene Graph:

Scene graph renderers usually do not override this function, as draw operations are usually taken care of automatically by the scene graph.

5.9.9 function diguyAuthorVisualWaypoint::show

Prototype:

```
virtual void show();
```

Description:

`show( )` will be called by DI-Guy if the visibility status of this visual has changed to be visible.

5.9.10 function diguyAuthorVisualWaypoint::hide

Prototype:

```
virtual void hide();
```

Description:

`hide( )` will be called by DI-Guy if the visibility status of this visual has changed to be not visible.

5.10 diguyChainSettings

diguyChainSettings.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

- ◆ General Functions
- ◆ Simulation Constants Functions
- ◆ Chain Visual Functions
- ◆ Coil-Specific Functions

5.10.1 class diguyChainSettings

```
class BDI_DECLSPEC_diguy diguyChainSettings
{
public:
    const char* get_name();
    int set_name(const char* name);
    void set_num_masses(int num_masses);
    int get_num_masses();
    void set_index_of_chain_mass_connected_to_character(int chain_mass_index);
    int get_index_of_chain_mass_connected_to_character();
    void set_simulation_dt(float simulation_dt);
    float get_simulation_dt();
    void set_mass(float mass);
    float get_mass();
    void set_spring_length(float spring_length);
    float get_spring_length();
    void set_spring_k(float spring_k);
    float get_spring_k();
    void set_spring_b(float spring_b);
    float get_spring_b();
    void set_neighbor2_spring_length(float neighbor2_spring_length);
    float get_neighbor2_spring_length();
    void set_neighbor2_spring_k(float neighbor2_spring_k);
    float get_neighbor2_spring_k();
    void set_neighbor2_spring_b(float neighbor2_spring_b);
    float get_neighbor2_spring_b();
    void set_ground_k(float ground_k);
    float get_ground_k();
    void set_ground_b(float ground_b);
    float get_ground_b();
    void set_ground_lateral_b(float ground_lateral_b);
    float get_ground_lateral_b();
    void set_ground_stiction_engage_vel(float ground_stiction_engage_vel);
    float get_ground_stiction_engage_vel();
    void set_ground_stiction_release_force(float ground_stiction_release_force);
    float get_ground_stiction_release_force();
    void set_ground_height(float ground_height);
    float get_ground_height();
```

```

void set_air_b(float air_b);
float get_air_b();
void set_gravity(float x, float y, float z);
void get_gravity(float* x, float* y, float* z);
void set_radius(float radius);
float get_radius();
void set_ambient_color(float r, float g, float b, float a);
void get_ambient_color(float* r, float* g, float* b, float* a);
void set_diffuse_color(float r, float g, float b, float a);
void get_diffuse_color(float* r, float* g, float* b, float* a);
void set_specular_color(float r, float g, float b, float a);
void get_specular_color(float* r, float* g, float* b, float* a);
void set_shininess(float shininess);
float get_shininess();
void set_texture_filename(const char* texture_filename);
const char* get_texture_filename();
void set_num_texture_repetitions(int num_texture_repetitions);
int get_num_texture_repetitions();
void set_cross_section(diguyChainCrossSection cs);
diguyChainCrossSection get_cross_section();
void set_appearance_begin_cap(const char* appearance_begin_cap);
const char* get_appearance_begin_cap();
void set_appearance_end_cap(const char* appearance_end_cap);
const char* get_appearance_end_cap();
void set_num_coils(int num_coils);
int get_num_coils();
void set_coil_thickness(float coil_thickness);
float get_coil_thickness();
void set_coil_taper_begin_mass_count(int coil_taper_begin_mass_count);
int get_coil_taper_begin_mass_count();
void set_coil_taper_end_mass_count(int coil_taper_end_mass_count);
int get_coil_taper_end_mass_count();

```

General Functions

5.10.2 function diguyChainSettings::get_name

Prototype:

```
const char* get_name();
```

Returns:

name of this object; this value will never be NULL

Callable From:

- ◇ C++
- ◇ Script

5.10.3 function diguyChainSettings::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Simulation Constants Functions

5.10.4 function diguyChainSettings::set_num_masses

Prototype:

```
void set_num_masses(int num_masses);
```

Description:

This function sets how many masses will in a chain using these settings.

The default is 80 masses.

Arguments:

num_masses number of masses

Callable From:

- ◇ C++
- ◇ Script

5.10.5 function diguyChainSettings::get_num_masses

Prototype:

```
int get_num_masses();
```

Returns:

number of masses; see [set_num_masses\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.6 function**diguyChainSettings::set_index_of_chain_mass_connected_to_ch****Prototype:**

```
void set_index_of_chain_mass_connected_to_character(int chain_mass_index);
```

Description:

This function sets which chain mass will be connected to the chain's associated character when a chain using these settings is created or reset.

Pass -1 for no mass to be connected.

The default is chain mass 0.

Arguments:

`chain_mass_index` index of chain mass

Callable From:

- ◇ C++
- ◇ Script

5.10.7 function**diguyChainSettings::get_index_of_chain_mass_connected_to_ch****Prototype:**

```
int get_index_of_chain_mass_connected_to_character();
```

Returns:

which mass to connect on reset; see

[set_index_of_chain_mass_connected_to_character\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.8 function diguyChainSettings::set_simulation_dt

Prototype:

```
void set_simulation_dt(float simulation_dt);
```

Description:

This function sets the dt of the chain simulation using these settings.

Higher dt values will provide faster simulations at the risk of instabilities. Conversely, lower dt values result in more simulation computational overhead, but the simulation will be more stable.

The default is 0.002 seconds.

Arguments:

`simulation_dt` dt of each simulation time step, in seconds

Callable From:

- ◇ C++
- ◇ Script

5.10.9 function diguyChainSettings::get_simulation_dt

Prototype:

```
float get_simulation_dt();
```

Returns:

dt of chain simulation; see [set_simulation_dt\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.10 function diguyChainSettings::set_mass**Prototype:**

```
void set_mass(float mass);
```

Description:

This function sets the mass of each chain mass of the chain simulation using these settings.

Heavier mass will result in chain masses that have more momentum when moving.

The default is 0.5 kg.

Arguments:

mass	mass of each chain mass, in kg

Callable From:

- ◇ C++
- ◇ Script

5.10.11 function diguyChainSettings::get_mass**Prototype:**

```
float get_mass();
```

Returns:

mass of each chain mass; see [set_mass\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.12 function diguyChainSettings::set_spring_length

Prototype:

```
void set_spring_length(float spring_length);
```

Description:

This function sets the nominal length of the springs connecting adjacent chain masses of the chain simulation using these settings.

Adjacent chain masses will try to keep this distance between them. A longer spring length will result in a longer chain.

The default is 0.2 meters

Arguments:

`spring_length` nominal length between adjacent chain masses, in meters

Callable From:

◇ C++

◇ Script

5.10.13 function diguyChainSettings::get_spring_length

Prototype:

```
float get_spring_length();
```

Returns:

nominal length between adjacent chain masses; see
[set_spring_length\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.14 function diguyChainSettings::set_spring_k

Prototype:

```
void set_spring_k(float spring_k);
```

Description:

This function sets the spring constant of the springs connecting adjacent chain masses of the chain simulation using these settings.

A higher spring constant will result in a more rigid chain, as adjacent masses will try to maintain their nominal spring length between them, as set by [set_spring_length\(\)](#).

A hose would have a high spring constant, while a "slinky" would have a low spring constant.

This spring constant only applies between adjacent masses.

The default is 10000.

Arguments:

`spring_k` spring constant of springs between adjacent masses

Callable From:

- ◇ C++
- ◇ Script

5.10.15 function `diguyChainSettings::get_spring_k`

Prototype:

```
float get_spring_k();
```

Returns:

spring constant of springs between adjacent masses; see [set_spring_k\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.16 function diguyChainSettings::set_spring_b**Prototype:**

```
void set_spring_b(float spring_b);
```

Description:

This function sets the damping constant of the springs connecting adjacent chain masses of the chain simulation using these settings.

A higher damping constant will result in a chain that doesn't oscillate as much; movement between masses will quickly die down.

A hose would have a high damping constant, while a "slinky" would have a low damping constant.

The default is 15.

damping constant of springs between adjacent masses

Callable From:

◇ C++

◇ Script

5.10.17 function diguyChainSettings::get_spring_b**Prototype:**

```
float get_spring_b();
```

Returns:

damping constant of springs between adjacent masses; see [set_spring_b\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.18 function diguyChainSettings::set_neighbor2_spring_length**Prototype:**

```
void set_neighbor2_spring_length(float neighbor2_spring_length);
```

Description:

This function sets the nominal length of the springs connecting '2nd neighbor' chain masses of the chain simulation using these settings.

2nd neighbor chain masses are two masses apart instead of one. (Adjacent masses are one mass apart.)

A longer 2nd neighbor spring length will result in a chain that is stiffer and doesn't droop. A lower value will result in a chain that can droop a lot from mass to mass.

A garden hose would have a high 2nd neighbor spring length, while a metal chain made of linked chain links would have a low 2nd neighbor spring length.

The default is 0.8 meters

Arguments:

`spring_length` nominal length between adjacent chain masses, in meters

Callable From:

◇ C++
◇ Script

5.10.19 function `diguyChainSettings::get_neighbor2_spring_length`

Prototype:

```
float get_neighbor2_spring_length();
```

Returns:

nominal distance between each mass and its '2nd neighbor'; see [`set\_neighbor2\_spring\_length\(\)`](#)

Callable From:

◇ C++
◇ Script

5.10.20 function diguyChainSettings::set_neighbor2_spring_k**Prototype:**

```
void set_neighbor2_spring_k(float neighbor2_spring_k);
```

Description:

This function sets the spring constant of springs connecting 2nd neighbor chain masses of the chain simulation using these settings.

See [set_spring_k\(\)](#) for a discussion of what this value means.

The default is 600.

Arguments:

`neighbor2_spring_k` spring constant of springs between
2nd neighbor masses

Callable From:

- ◇ C++
- ◇ Script

5.10.21 function diguyChainSettings::get_neighbor2_spring_k**Prototype:**

```
float get_neighbor2_spring_k();
```

Returns:

spring constant of springs between 2nd neighbor masses; see [set_neighbor2_spring_k\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.22 function diguyChainSettings::set_neighbor2_spring_b**Prototype:**

```
void set_neighbor2_spring_b(float neighbor2_spring_b);
```

Description:

This function sets the damping constant of springs connecting 2nd neighbor chain masses of the chain simulation using these settings.

See [set_spring_b\(\)](#) for a discussion of what this value means.

The default is 5.

Arguments:

`neighbor2_spring_b` damping constant of springs between
2nd neighbor masses

Callable From:

◇ C++
◇ Script

5.10.23 function diguyChainSettings::get_neighbor2_spring_b

Prototype:

```
float get_neighbor2_spring_b();
```

Returns:

damping constant of springs between 2nd neighbor masses; see
[set_neighbor2_spring_b\(\)](#)

Callable From:

◇ C++
◇ Script

5.10.24 function diguyChainSettings::set_ground_k**Prototype:**

```
void set_ground_k(float ground_k);
```

Description:

This function sets the spring constant of the ground model.

A higher spring constant will result in a harder ground that bounces chain masses higher when they come in contact with the ground. Also, chain masses will not sink as far into the ground when they come to rest.

A tile floor would have a high spring constant, while a mattress would have a low spring constant.

The default is 5000.

Arguments:

ground_k	spring constant of ground model

Callable From:

- ◇ C++
- ◇ Script

5.10.25 function diguyChainSettings::get_ground_k

Prototype:

```
float get_ground_k();
```

Returns:

spring constant of ground model; see [set_ground_k\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.26 function diguyChainSettings::set_ground_b

Prototype:

```
void set_ground_b(float ground_b);
```

Description:

This function sets the damping constant of the ground model.

A higher damping constant will result in a ground that absorbs impacts and doesn't bounce masses as much when they hit the ground.

A tile floor would have a low damping constant, while an exercise mat would have a high damping constant.

The default is 100.

Arguments:

`ground_b` vertical damping constant of ground model

Callable From:

- ◇ C++
- ◇ Script

5.10.27 function diguyChainSettings::get_ground_b

Prototype:

```
float get_ground_b();
```

Returns:

vertical damping constant of ground model; see [set_ground_b\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.28 function diguyChainSettings::set_ground_lateral_b**Prototype:**

```
void set_ground_lateral_b(float ground_lateral_b);
```

Description:

This function sets how much the ground model slows down masses that are sliding on it.

A higher lateral damping constant will quickly slow down masses sliding along the ground.

Dirt or grass ground would have a high lateral damping constant. Ice would have a very low damping constant.

The default is 0.5.

Arguments:

`ground_lateral_b` lateral damping constant of ground model

Callable From:

◇ C++

◇ Script

5.10.29 function diguyChainSettings::get_ground_lateral_b**Prototype:**

```
float get_ground_lateral_b();
```

Returns:

lateral damping constant of ground model; see
[set_ground_lateral_b\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.30 function diguyChainSettings::set_ground_stiction_engage_vel**Prototype:**

```
void set_ground_stiction_engage_vel(float ground_stiction_engage_vel);
```

Description:

This function sets how slow a mass in contact with the ground must go in order for the mass to stop moving and "stick" to the ground. This is a simple form of "stiction".

A mass will unstick if the force being applied to it exceeds the stiction release force as set by [set_ground_stiction_release_force\(\)](#).

The default is 0.05 m/s.

Arguments:

`ground_stiction_engage_vel` velocity at stiction will engage

Callable From:

- ◇ C++
- ◇ Script

5.10.31 function diguyChainSettings::get_ground_stiction_engage_vel

Prototype:

```
float get_ground_stiction_engage_vel();
```

Returns:

velocity at stiction will engage; see
[set_ground_stiction_engage_vel\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.32 function diguyChainSettings::set_ground_stiction_release_force

Prototype:

```
void set_ground_stiction_release_force(float ground_stiction_release_force);
```

Description:

This function sets how much force is required to cause a mass that has stuck to the ground because of stiction to be released.

The default is 10 newtons.

Arguments:

ground_stiction_release_force	force at which masses will release from stiction

Callable From:

- ◇ C++
- ◇ Script

5.10.33 function diguyChainSettings::get_ground_stiction_release_force

Prototype:

```
float get_ground_stiction_release_force();
```

Returns:

force at which masses will release from stiction; see
[set_ground_stiction_release_force\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.34 function diguyChainSettings::set_ground_height**Prototype:**

```
void set_ground_height(float ground_height);
```

Description:

This function sets the height of the ground model. Chain masses below this height will be repelled upward based on their mass and the ground spring and damping constants.

Currently the ground is a level plane.

The default is 0 meters.

Arguments:

`ground_height` height of the ground model, in meters

Callable From:

- ◇ C++
- ◇ Script

5.10.35 function diguyChainSettings::get_ground_height**Prototype:**

```
float get_ground_height();
```

Returns:

height of the ground model; see [set_ground_height\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.36 function diguyChainSettings::set_air_b**Prototype:**

```
void set_air_b(float air_b);
```

Description:

This function sets the damping constant caused by air resistance.

A higher air damping constant will result in chains that don't swing as much. A low air damping constant will result in chains that swing freely for a long time.

Water would have a very high "air" damping constant. A clock pendulum would have a very low air damping constant.

The default is 0.1.

Arguments:

<code>air_b</code>	damping constant of air, or air resistance

Callable From:

- ◇ C++
- ◇ Script

5.10.37 function diguyChainSettings::get_air_b**Prototype:**

```
float get_air_b();
```

Returns:

damping constant of air; see [set_air_b\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.38 function diguyChainSettings::set_gravity**Prototype:**

```
void set_gravity(float x, float y, float z);
```

Description:

This function sets how strongly link masses will be accelerated in the gravity direction, typically downward.

The default is (0, 0, -9.8), in meters per second squared.

Arguments:

`x, y, z` accelerations due to gravity

Callable From:

- ◇ C++
- ◇ Script

5.10.39 function diguyChainSettings::get_gravity**Prototype:**

```
void get_gravity(float* x, float* y, float* z);
```

Returns:

x, y, and z components of gravity; see [set_gravity\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

Chain Visual Functions

5.10.40 function diguyChainSettings::set_radius

Prototype:

```
void set_radius(float radius);
```

Description:

This function sets the radius of the chain visual.

The default is 0.05 meters.

Arguments:

radius	radius of chain visual, in meters

Callable From:

- ◇ C++
- ◇ Script

5.10.41 function diguyChainSettings::get_radius

Prototype:

```
float get_radius();
```

Returns:

radius of chain visual; see [set_radius\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.42 function diguyChainSettings::set_ambient_color**Prototype:**

```
void set_ambient_color(float r, float g, float b, float a);
```

Description:

This function sets the ambient color of the chain visual. Passed values should be between 0 and 1.

The default values are (0.4, 0.4, 0.4, 1).

Arguments:

r	red component
g	blue component
b	green component
a	alpha component

Callable From:

- ◇ C++
- ◇ Script

5.10.43 function diguyChainSettings::get_ambient_color**Prototype:**

```
void get_ambient_color(float* r, float* g, float* b, float* a);
```

Description:

This function gets the ambient color of the chain visual. Values are returned via the passed pointers.

Arguments:

r	red component
g	blue component
b	green component
a	alpha component

Callable From:

- ◇ C++
- ◇ Script

5.10.44 function diguyChainSettings::set_diffuse_color**Prototype:**

```
void set_diffuse_color(float r, float g, float b, float a);
```

Description:

This function sets the diffuse color of the chain visual. Passed values should be between 0 and 1.

The default values are (1, 1, 1, 1).

Arguments:

r	red component
g	blue component
b	green component
a	alpha component

Callable From:

◇ C++
◇ Script

5.10.45 function diguyChainSettings::get_diffuse_color

Prototype:

```
void get_diffuse_color(float* r, float* g, float* b, float* a);
```

Description:

This function gets the diffuse color of the chain visual. Values are returned via the passed pointers.

Arguments:

r	red component
g	blue component
b	green component
a	alpha component

Callable From:

◇ C++

◇ Script

5.10.46 function diguyChainSettings::set_specular_color**Prototype:**

```
void set_specular_color(float r, float g, float b, float a);
```

Description:

This function sets the specular color of the chain visual. Passed values should be between 0 and 1.

The default values are (1, 1, 1, 1).

Arguments:

r	red component
g	blue component
b	green component
a	alpha component

Callable From:

◇ C++
◇ Script

5.10.47 function diguyChainSettings::get_specular_color

Prototype:

```
void get_specular_color(float* r, float* g, float* b, float* a);
```

Description:

This function gets the specular color of the chain visual. Values are returned via the passed pointers.

Arguments:

r	red component
g	blue component
b	green component
a	alpha component

Callable From:

◇ C++

◇ Script

5.10.48 function diguyChainSettings::set_shininess

Prototype:

```
void set_shininess(float shininess);
```

Description:

This function sets the shininess of the chain visual.

The default is 20.

Arguments:

`shininess` the shininess of the chain visual

Callable From:

◇ C++

◇ Script

5.10.49 function diguyChainSettings::get_shininess**Prototype:**

```
float get_shininess();
```

Returns:

the shininess of the chain visual; see [set_shininess\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.50 function diguyChainSettings::set_texture_filename**Prototype:**

```
void set_texture_filename(const char* texture_filename);
```

Description:

This function sets the filename of the texture that will be used for the chain visual.

Common textures are:

- ◇ "chain_link.rgba"
- ◇ "firefighter_hose2.rgb"

The default is "" (no texture).

Arguments:

`texture_filename` filename of texture of chain visual

Callable From:

- ◇ C++
- ◇ Script

5.10.51 function diguyChainSettings::get_texture_filename

Prototype:

```
const char* get_texture_filename();
```

Returns:

filename of texture of chain visual; see [set_texture_filename\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.52 function diguyChainSettings::set_num_texture_repetitions

Prototype:

```
void set_num_texture_repetitions(int num_texture_repetitions);
```

Description:

This function sets how many times the texture will be repeated along the length of the chain.

The default is 1 rep.

Arguments:

num_texture_repetitions how many times texture will be repeated
on chain visual

Callable From:

- ◇ C++
- ◇ Script

5.10.53 function diguyChainSettings::get_num_texture_repetitions

Prototype:

```
int get_num_texture_repetitions();
```

Returns:

how many times texture will be repeated on chain visual; see
[set_num_texture_repetitions\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.54 function `diguyChainSettings::set_cross_section`

Prototype:

```
void set_cross_section(diguyChainCrossSection cs);
```

Description:

This function sets what the "cross section" of the chain visual will be.

See the documentation for [diguyChainCrossSection](#) for information on what the different cross sections mean.

The default is `DIGUY_CHAIN_CROSS_SECTION_SQUARE`.

Arguments:

<code>CS</code>	cross section of the chain visual

Callable From:

- ◇ C++
- ◇ Script

5.10.55 function `diguyChainSettings::get_cross_section`

Prototype:

```
diguyChainCrossSection get_cross_section();
```

Returns:

cross section type of chain visual; see [set_cross_section\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.56 function diguyChainSettings::set_appearance_begin_cap**Prototype:**

```
void set_appearance_begin_cap(const char* appearance_begin_cap);
```

Description:

Set the shape that will be attached to the beginning end of the chain. Options are:

- ◇ "chain_cap_firehose_nozzle"
- ◇ "chain_cap_fuel_hose_nozzle"
- ◇ "chain_cap_taser_probe"
- ◇ "chain_cap_chain_hook"

The default is "" (no begin cap).

Arguments:

appearance_begin_cap name of begin cap

Callable From:

- ◇ C++
- ◇ Script

5.10.57 function diguyChainSettings::get_appearance_begin_cap

Prototype:

```
const char* get_appearance_begin_cap();
```

Returns:

name of cap appearance at beginning of chain; see [set_appearance_begin_cap\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.58 function diguyChainSettings::set_appearance_end_cap

Prototype:

```
void set_appearance_end_cap(const char* appearance_end_cap);
```

Description:

Sets the shape that will be attached to the end of the chain. See [set_appearance_begin_cap\(\)](#).

The default is "" (no end cap).

Arguments:

appearance_end_cap name of end cap

Callable From:

- ◇ C++
- ◇ Script

5.10.59 function diguyChainSettings::get_appearance_end_cap**Prototype:**

```
const char* get_appearance_end_cap();
```

Returns:

name of cap appearance at end of chain; see
[set_appearance_end_cap\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

Coil-Specific Functions

5.10.60 function diguyChainSettings::set_num_coils

Prototype:

```
void set_num_coils(int num_coils);
```

Description:

For chains with the [DIGUY_CHAIN_CROSS_SECTION_COIL](#) cross section, sets the number of coil windings of the chain.

The default is 4 coils.

Arguments:

`num_coils` number of coil windings

Callable From:

- ◇ C++
- ◇ Script

5.10.61 function diguyChainSettings::get_num_coils

Prototype:

```
int get_num_coils();
```

Returns:

number of coils for chain visual of type
[DIGUY_CHAIN_CROSS_SECTION_COIL](#); see [set_num_coils\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.62 function diguyChainSettings::set_coil_thickness**Prototype:**

```
void set_coil_thickness(float coil_thickness);
```

Description:

Sets how thick the coil will be.

The default is 0.02 meters.

Arguments:

`coil_thickness` thickness of coil

Callable From:

- ◇ C++
- ◇ Script

5.10.63 function diguyChainSettings::get_coil_thickness**Prototype:**

```
float get_coil_thickness();
```

Returns:

thickness of coils for chain visual of type
[DIGUY_CHAIN_CROSS_SECTION_COIL](#); see
[set_coil_thickness\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.64 function diguyChainSettings::set_coil_taper_begin_mass_count

Prototype:

```
void set_coil_taper_begin_mass_count(int coil_taper_begin_mass_count);
```

Description:

Sets at which mass in the coil the thickness of the coil will reach its full radius.

The default is 6.

Arguments:

`coil_taper_begin_mass_count` chain mass at which to begin taper

Callable From:

- ◇ C++
- ◇ Script

5.10.65 function diguyChainSettings::get_coil_taper_begin_mass_count

Prototype:

```
int get_coil_taper_begin_mass_count();
```

Returns:

mass at which the coil will reach its full radius at the beginning of chain; see [set_coil_taper_begin_mass_count\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.10.66 function diguyChainSettings::set_coil_taper_end_mass_count

Prototype:

```
void set_coil_taper_end_mass_count(int coil_taper_end_mass_count);
```

Description:

Similar to [set_coil_taper_begin_mass_count\(\)](#), but for the end of the chain.

The default is 0.

Arguments:

`coil_taper_end_mass_count` chain mass at which to end taper

Callable From:

- ◇ C++
- ◇ Script

5.10.67 function diguyChainSettings::get_coil_taper_end_mass_count

Prototype:

```
int get_coil_taper_end_mass_count();
```

Returns:

mass at which the coil will reach its full radius at end of chain; see [set_coil_taper_end_mass_count\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.11 diguyChainSimulation

diguyChainSimulation.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

5.11.1 class diguyChainSimulation

```
class BDI_DECLSPEC_diguy diguyChainSimulation
{
public:
    diguyChainSettings* get_settings();
    int update_settings(const char* settings);
    int do_stabilization_time_runoff(float time_runoff);
    void set_is_frozen(int is_frozen);
    int get_is_frozen();
    int set_chain_mass_connection_character(int chain_mass_index,
        const char* character_name,
        const char* link_name = NULL,
        float link_offset_x = 0.0f,
        float link_offset_y = 0.0f,
        float link_offset_z = 0.0f,
        float connection_smoothing_period = 0.1f);
    int set_chain_mass_position(int chain_mass_index,
        float mass_x,
        float mass_y,
        float mass_z,
        int fix_mass_position);
    int set_chain_mass_positions_along_path_shape(diguyPathShape* path_shape,
        int stretch_to_fit = 1);
    int set_chain_mass_position_is_fixed(int chain_mass_index,
        int is_fixed);
    int get_chain_mass_position_is_fixed(int mass_index);
    float get_length();
```

5.11.2 function diguyChainSimulation::get_settings

Prototype:

```
diguyChainSettings* get_settings();
```

Description:

This function returns the chain settings this chain simulation is using.

Returns:

object of type [diguyChainSettings](#); will never be NULL

Callable From:

- ◇ C++
- ◇ Script

5.11.3 function diguyChainSimulation::update_settings

Prototype:

```
int update_settings(const char* settings);
```

Description:

This function sets the settings this chain simulation will use.

Arguments:

`settings` name of new settings to use

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.11.4 function

diguyChainSimulation::do_stabilization_time_runoff

Prototype:

```
int do_stabilization_time_runoff(float time_runoff);
```

Description:

This function runs the simulation forward the specified number of seconds, to let the chain settle into a stable configuration.

Note that the chain will appear to achieve the new configuration instantly.

Specifying a large amount of time can result in a frame rate hitch.

Arguments:

`time_runoff` how much time to run off, in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.11.5 function diguyChainSimulation::set_is_frozen

Prototype:

```
void set_is_frozen(int is_frozen);
```

Description:

This function can 'freeze' or 'unfreeze' a chain.

If a chain is frozen, all chain masses will remain in their current positions and will not move until the chain is unfrozen.

A frozen chain requires no simulation time.

Arguments:

`is_frozen` pass 1 to freeze the chain; 0 to unfreeze

Callable From:

- ◇ C++
- ◇ Script

5.11.6 function diguyChainSimulation::get_is_frozen

Prototype:

```
int get_is_frozen();
```

Returns:

1 if chain is frozen, 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.11.7 function**diguyChainSimulation::set_chain_mass_connection_character****Prototype:**

```
int set_chain_mass_connection_character(int chain_mass_index,
    const char* character_name,
    const char* link_name = NULL,
    float link_offset_x = 0.0f,
    float link_offset_y = 0.0f,
    float link_offset_z = 0.0f,
    float connection_smoothing_period = 0.1f);
```

Description:

This function connects a chain mass to a character, or disconnects a chain mass if the passed `character_name` is NULL.

As the character moves the chain mass will move with it, dragging adjacent masses along. This allows a firefighter to pick up one end of a hose and drag the hose to a new location, for example.

In each time-step of the scenario a new position for the specified chain mass will be computed based on the position of the character.

When the chain sim is created, and when the scenario is reset, one chain mass will be automatically connected to the chain's character. The index of the mass that will be connected is set by a call to

[`diguyChainSettings::set\_index\_of\_chain\_mass\_connected\_to\_character\(\)`](#).

Chain masses can be connected to a specific link of the connection character by passing the link's name. If NULL is passed, the connection will be made to the character's "position" link. Note that a non-NULL link name must be used for the link offset arguments to have an effect.

The chain mass can gradually be shifted from its current position to the connection position using the `connection_smoothing_period` argument. Passing in 0.5, for example, will cause the link to shift to its new connection point within approximately 0.5 seconds, avoiding an abrupt and potentially ugly impulse to the simulation.

Arguments:

<code>chain_mass_index</code>	index of chain mass
<code>character_name</code>	name of character to attach to; must not be NULL
<code>link_name</code>	name of link to attach to; can be NULL; if so, will attach to character's base position

<code>link_offset_x</code>	link attachment offset in X
<code>link_offset_y</code>	link attachment offset in Y
<code>link_offset_z</code>	link attachment offset in Z
<code>connection_smoothing_period</code>	how much time to take, in seconds, to gradually shift the chain mass to its new connection point

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.11.8 function `diguyChainSimulation::set_chain_mass_position`

Prototype:

```
int set_chain_mass_position(int chain_mass_index,
    float mass_x,
    float mass_y,
    float mass_z,
    int fix_mass_position);
```

Description:

This function sets the position of one chain mass, and optionally fixes it there.

The passed position is always relative to the chain character's parent, if there is one.

If the `fix_mass_position` argument is 1, the chain mass will remain fixed in the new position, as if `set_chain_mass_position_is_fixed(1)` were called.

Arguments:

<code>chain_mass_index</code>	index of chain mass; pass -1 to affect all chain masses
<code>mass_x</code>	position x
<code>mass_y</code>	position y
<code>mass_z</code>	position z
<code>fix_mass_position</code>	pass 1 to fix chain mass in place; 0 to leave it free to move

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.11.9 function `diguyChainSimulation::set_chain_mass_positions_along_path_shape`

Prototype:

```
int set_chain_mass_positions_along_path_shape(diguyPathShape* path_shape,
int stretch_to_fit = 1);
```

Description:

This function sets the positions of all chain masses to lie along the passed path shape.

If the `stretch_to_fit` argument is 1, chain masses will be spaced along the entire length of the path shape. If the path shape is longer than the nominal length of the chain, the chain will be stretched out. If the path shape is shorter, the chain will be compressed.

Arguments:

<code>path_shape</code>	path shape for new chain mass positions
<code>stretch_to_fit</code>	

pass 1 to stretch chain to take up entire length of path_shape; pass 0 to leave chain at its nominal length

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.11.10 function diguyChainSimulation::set_chain_mass_position_is_fixed

Prototype:

```
int set_chain_mass_position_is_fixed(int chain_mass_index,
    int is_fixed);
```

Description:

This function can fix or unfix a chain mass in place. If it is fixed the chain mass will remain in its current position until it is unfixed.

Fixed chain masses do not require simulation time, so the more masses in a chain that are fixed, the lower the computational load of the chain simulation.

Arguments:

chain_mass_index	index of chain mass; pass -1 to affect all chain masses
is_fixed	pass 1 to fix chain mass position; 0 to unfix it

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.11.11 function `diguyChainSimulation::get_chain_mass_position_is_fixed`

Prototype:

```
int get_chain_mass_position_is_fixed(int mass_index);
```

Returns:

1 if chain mass is fixed, 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.11.12 function diguyChainSimulation::get_length**Prototype:**

```
float get_length();
```

Returns:

length of the chain

Callable From:

- ◇ C++
- ◇ Script

5.12 diguyCharacter

diguyCharacter.h

DI-Guy SDK Version 12.5.2

Includes: [declspec_diguy.h](#) | [diguy_constants.h](#) | [diguy_typedefs.h](#) | [diguy_vector_classes.h](#) | [diguyCharacterPoseOverride.h](#) | [diguyCharacterTypeMap.h](#) | [diguyMotionDirection.h](#) | [diguyMotionPosture.h](#) | [diguyMotionVariant.h](#)

Contents:

- ◆ General Functions
- ◆ C++ Only Functions
- ◆ Action Functions
- ◆ Position and Orientation Functions
- ◆ Appearance Functions
- ◆ Performance Functions
- ◆ Callback Functions
 - Callback Enums:
 - DI-Guy Scenario UI Callback enums:
 - DI-Guy AI Agent Callback enums:
- ◆ Event Handler Functions
- ◆ Path Functions
- ◆ Link and Shape Functions
- ◆ Altitude Functions
- ◆ Pose Functions
 - ◇ Pose Override Stages
 - ◇ Pose Override C++ Example
- ◆ Gaze and Point Functions
- ◆ Aim Functions
 - ◇ Aim Algorithm 8 Parameters
- ◆ Weapon Functions
- ◆ Face Functions
- ◆ Gesture Functions
 - ◇ Gesture Stages
 - ◇ Gesture Channels
- ◆ Sound Functions
- ◆ Network Functions
- ◆ Formation Functions
- ◆ Guide Functions
 - ◇ Guide Algorithms
 - Exact
 - Follow1
 - Follow2
 - Drift1
- ◆ Motion Texture Functions
- ◆ History Functions
- ◆ Meta-Action Functions
- ◆ Variable Functions

- ◆ Label Functions
- ◆ Impact Functions
- ◆ Appearance Effect Functions
- ◆ Chain Functions
- ◆ Vehicle Functions
- ◆ Shader Functions
- ◆ DI-Guy AI and Agent Functions
 - ◇ Attack Functions
 - ◇ Flee Functions
 - ◇ Mingle Functions
 - ◇ Wander Functions
 - ◇ Pursue Functions
 - ◇ Travel Functions
 - ◇ Path Follow Functions
 - ◇ Other Agent Functions
 - ◇ User-Defined Feelers
- ◆ Lua Mind Functions
- ◆ Visibility Functions
 - ◇ Visibility Tests
 - ◇ AI Callback Functions
- ◆ Simulation Functions
- ◆ Experimental Functions
- ◆ Deprecated Functions

Link against: libdiguy

5.12.1 class diguyCharacter

```
class BDI_DECLSPEC_diguy diguyCharacter
{
public:
    const char* get_name();
    int set_name(const char* name);
    const char* get_type_name();
    long get_uid();
    int get_index();
    int update(float t, int full_update = 1);
    void set_character_type(const char* type);
    const char* get_character_type_string();
    const char* get_character_class();
    diguyScenario* get_scenario();
    int set_enabled(int enabled);
    int get_enabled();
    int set_drawn_by_scenario_flag(int flag);
    int get_drawn_by_scenario_flag();
    void set_invisible_flag(int invisible_flag);
    int get_invisible_flag();
    int get_is_active();
    int get_is_drawn();
    int get_is_temporary();
    int get_is_recycled();
    int set_t_controlled_by_scenario_t(int t_controlled_by_scenario_t);
    int get_t_controlled_by_scenario_t();
```

```

int set_initial_tin(float tin, int override_time_warning = 0);
int set_initial_tout(float tout);
int set_current_tin(float tin);
int set_current_tin_to_now();
float get_current_tin();
int set_current_tout(float tout,
    int disable_character_at_tout = 0);
int set_current_tout_to_now();
float get_current_tout();
void set_tout_enabled(int tout_enabled);
int get_tout_enabled();
int set_careful_interpolation_threshold(float threshold);
float get_careful_interpolation_threshold();
int set_motion_interpolation_flag(int flag);
int get_motion_interpolation_flag();
int set_scale(float scale_x,
    float scale_y,
    float scale_z);
int get_scale(float* scale_x,
    float* scale_y,
    float* scale_z);
int set_parent(const char* parent_name,
    const char* parent_link_name = "");
diguyCharacter* get_parent();
const char* get_parent_link_name();
int unset_parent();
int get_num_children();
diguyCharacter* get_child_at_index(int index);
int get_is_scene_object();
int set_is_scene_object(int is_scene_object);
float get_bounding_radius();
float get_default_bounding_radius();
void set_bounding_radius(float f);
float get_time_to_transition();
diguyCharacterMode get_action_mode();
diguyCharacterMode get_position_mode();
int set_t_scale_factor(float t_scale_factor);
int unset_t_scale_factor();
int is_within_distance_n_of_character(const char* character_name,
    float distance);
int is_within_distance_n_of_member_of_group(const char* group_name,
    float distance);
int is_group_member(const char* group_name);
int get_num_group_memberships();
diguyCharacterGroup* get_group_membership_at_index(int index);
diguyCharacter* get_nearest_active_character(int check_visibility,
    int xy_distance = 0,
    float max_distance_to_check = 100000);
diguyCharacter* get_nearest_active_character_in_group(const char* group_name,
    int check_visibility = 1,
    int xy_distance = 0,
    float max_distance_to_check = 100000);
diguyCharacter* get_random_active_character(float max_distance = 10.0f,
    int check_visibility = 1,
    int xy_distance = 0);
diguyCharacter* get_random_active_character_in_group(const char* group_name,
    float max_distance = 10,
    int check_visibility = 1,
    int xy_distance = 0);
float get_distance_to_character(diguyCharacter* character);
float get_distance_xy_to_character(diguyCharacter* character);

```

```

float get_distance_to_impact(diguyImpact* impact);
float get_random_factor();
int draw();
int draw_pass1();
int draw_pass2();
#ifdef CPLUSPLUS_ONLY
void* get_graphics_ptr();
int set_graphics_ptr(void* graphics_ptr);
void set_graphics_api_node_ptr(void* node_ptr);
void* get_graphics_api_node_ptr();
int set_user_data(void* user_data);
void* get_user_data();
#endif
int set_desired_action(const char* action_name,
    float speed = DIGUY_DEFAULT_FLOAT,
    int retain_path_shape = 0);
const char* get_desired_action();
const char* get_current_action();
int force_action(const char* action_name,
    float speed = DIGUY_DEFAULT_FLOAT,
    int include_transition_arc = 1,
    float max_rampdown_interval = 0.5f,
    int retain_path_shape = 0,
    float t_offset_into_new_action = 0.0f);
int force_action_with_duration(const char* action_name,
    float duration,
    int include_transition_arc = 1,
    float max_rampdown_interval = 0.5f);
int force_action_and_path_shape(const char* action_name,
    const char* path_shape_name,
    const char* waypoint_name = NULL,
    float distance_into_path = 0.0f);
const char* get_most_recent_forced_action();
float get_most_recent_forced_action_t();
float get_most_recent_forced_action_t_offset();
int add_pending_desired_action(const char* action,
    float scenario_t,
    int remove_existing_pending_actions = 0,
    float speed = DIGUY_DEFAULT_FLOAT,
    int retain_path_shape = 1);
int add_pending_force_action(const char* action,
    float scenario_t,
    int remove_existing_pending_actions = 0,
    float speed = DIGUY_DEFAULT_FLOAT,
    int include_transition_arc = 1,
    float max_rampdown_interval = 0.5f,
    int retain_path_shape = 1,
    float first_arc_time_shift = 0.0f);
int is_valid_action(const char* action);
int die_now(const char* preferred_dead_action_name = "(default)");
int revive_now(const char* preferred_revive_action_name = "(default)");
int get_dead();
int set_speed(float speed);
float get_speed();
int set_position(float tx, float ty, float tz);
int set_position_double(double tx, double ty, double tz);
int get_position(float* tx, float* ty, float* tz);
int get_position_double(double* tx, double* ty, double* tz);
int set_position_relative_to_parent(float tx, float ty, float tz);
int get_position_relative_to_parent(float* tx, float* ty, float* tz);
int set_position_to_other_character_position(const char* other_character_name,

```

```

const char* other_link_name = NULL,
float offset_x = 0.0f, float offset_y = 0.0f, float offset_z = 0.0f,
int also_set_orientation = 1);
int set_desired_position(float tx, float ty, float tz,
int force_guide_unacquired = 0);
int set_desired_position_to_current_position(int also_set_orientation = 1);
int set_desired_position_to_waypoint(diguyWaypoint* waypoint,
float offset_x = 0.0f, float offset_y = 0.0f, float offset_z = 0.0f,
int offset_in_world_coords = 0);
int set_desired_position_to_path_waypoint(const char* path_name,
const char* waypoint_name);
int set_desired_position_to_path_shape_waypoint(const char* path_shape_name,
const char* waypoint_name);
int get_desired_position(float* tx, float* ty, float* tz);
int get_delta_to_desired_position(float* tx, float* ty, float* tz);
int set_initial_position(float tx, float ty, float tz);
int get_initial_position(float* tx, float* ty, float* tz);
int set_orientation(float rz, float rx, float ry);
int get_orientation(float* rz, float* rx, float* ry);
int set_orientation_relative_to_parent(float rz, float rx, float ry);
int get_orientation_relative_to_parent(float* rz, float* rx, float* ry);
int set_initial_up_vector(char axis);
char get_initial_up_vector();
int set_up_vector(char axis);
char get_up_vector();
int set_custom_orientation_rx_and_ry(float rx,
float ry);
int set_desired_orientation(float rz, float rx, float ry,
int force_guide_unacquired = 0);
int get_desired_orientation(float* rz, float* rx, float* ry);
int get_delta_to_desired_orientation(float* rz, float* rx, float* ry);
int set_desired_orientation_to_current_orientation();
int set_desired_orientation_towards_position(float x, float y, float z,
int force_guide_unacquired = 0);
int set_initial_orientation(float rz, float rx, float ry);
int get_initial_orientation(float* rz, float* rx, float* ry);
int get_link_position(const char* link_name,
float* tx, float* ty, float* tz,
float* rz, float* rx, float* ry);
int get_link_position_double(const char* link_name,
double* tx, double* ty, double* tz,
double* rz, double* rx, double* ry);
int get_link_position_with_offset(const char* link_name,
float offset_tx, float offset_ty, float offset_tz,
float* tx, float* ty, float* tz,
float* rz, float* rx, float* ry);
int get_link_position_with_offset_double(const char* link_name,
double offset_tx, double offset_ty, double offset_tz,
double* tx, double* ty, double* tz,
double* rz, double* rx, double* ry);
int get_link_relative_position(const char* beginning_link_name,
const char* ending_link_name,
float* tx, float* ty, float* tz,
float* rz, float* rx, float* ry);
int get_link_relative_position_with_offset(const char* beginning_link_name,
const char* ending_link_name,
float offset_tx, float offset_ty, float offset_tz,
float* tx, float* ty, float* tz,
float* rz, float* rx, float* ry);
void set_apply_actor_scale_to_action_bead_travel(int apply_scale_flag);
int get_apply_actor_scale_to_action_bead_travel();

```

```

int local_to_global(float x, float y, float z,
    float* res_x, float* res_y, float* res_z);
int global_to_local(float x, float y, float z,
    float* res_x, float* res_y, float* res_z);
void get_velocity(float* vel_x, float* vel_y, float* vel_z);
void set_desired_velocity(float vel_x, float vel_y, float vel_z);
void get_angular_velocity(float* vel_rz);
const char* get_appearance();
int set_appearance(const char* appearance);
int set_current_appearance(const char* appearance,
    int allow_actor_change = 1);
const char* get_current_appearance();
int get_current_appearance_is(const char* appearance);
int set_current_head_appearance(const char* head_appearance);
const char* get_current_head_appearance();
int get_current_head_appearance_is(const char* head_appearance);
int get_num_appearances_of_type(diguyCharacterAppearanceTypes appearance_type);
const char* get_appearance_name_at_index(diguyCharacterAppearanceTypes appearance_type,
    int index);
int set_lod_ranges(float* lod_ranges);
int set_graphics_lod(int lod);
int get_graphics_lod();

int set_shader_lod(int lod);
int get_shader_lod();

int set_motion_lod(int motion_lod);
int get_motion_lod();
int get_current_motion_lod();
int set_minimum_cpl(int minimum_cpl);
int get_minimum_cpl();
int get_maximum_possible_current_cpl();
int maximize_current_cpl();
int set_automatic_cpl_switching_enabled(int enabled);
int get_automatic_cpl_switching_enabled();
int set_current_cpl(int current_cpl);
int get_current_cpl();
int set_position_update_rate(int update_rate);
int get_position_update_rate();
int set_pose_update_rate(int update_rate);
int get_pose_update_rate();
int set_minimum_full_update_period(float min_period);
float get_minimum_full_update_period();
int set_position_accumulation_disabled(int disabled_flag);
int get_position_accumulation_disabled();
int set_all_blends_disabled(int disabled_flag);
int get_all_blends_disabled();
int set_shape_switches_disabled(int disabled_flag);
int get_shape_switches_disabled();
int set_shape_callbacks_disabled(int disabled_flag);
int get_shape_callbacks_disabled();
int set_graphics_api_shape_update_disabled(int disabled_flag);
int get_graphics_api_shape_update_disabled();
int optimized_update(float t);
int set_is_load_managed(int is_load_managed);
int get_is_load_managed();
enum {
    CALLBACK_ID_CREATE = 1,
    CALLBACK_ID_DESTROY,
    CALLBACK_ID_PRE_CREATE,
    CALLBACK_ID_CURRENT_APPEARANCE_CHANGED,
    CALLBACK_ID_CURRENT_HEAD_APPEARANCE_CHANGED,

```

CALLBACK_ID_PRE_UPDATE,
 CALLBACK_ID_POST_UPDATE,
 CALLBACK_ID_DONE_SPEAKING,
 CALLBACK_ID_GAZE_STATUS,
 CALLBACK_ID_LPOINT_STATUS,
 CALLBACK_ID_SHOW,
 CALLBACK_ID_HIDE,
 CALLBACK_ID_PRE_DIE,
 CALLBACK_ID_POST_DIE,
 CALLBACK_ID_PRE_FIRE_WEAPON,
 CALLBACK_ID_POST_FIRE_WEAPON,
 CALLBACK_ID_IMPACT,
 CALLBACK_ID_FIRE_WEAPON_SUCCESS,
 CALLBACK_ID_CURRENT_ACTION_CHANGED,
 CALLBACK_ID_DESIRED_ACTION_CHANGED,
 CALLBACK_ID_DESIRED_ACTION_REACHED,
 CALLBACK_ID_MANUALLY_INVOKED,
 CALLBACK_ID_USER_SELECTED,
 CALLBACK_ID_USER_UNSELECTED,
 CALLBACK_ID_IGUY_INTERACT,
 CALLBACK_ID_IGUY_INTERACT_WITH_SUBJECT,
 CALLBACK_ID_GUIDE_POSITION_ACQUIRED,
 CALLBACK_ID_GUIDE_POSITION_UNACQUIRED,
 CALLBACK_ID_GUIDE_ORIENTATION_ACQUIRED,
 CALLBACK_ID_GUIDE_ORIENTATION_UNACQUIRED,
 CALLBACK_ID_GUIDE_ALTITUDE_ACQUIRED,
 CALLBACK_ID_GUIDE_ALTITUDE_UNACQUIRED,
 CALLBACK_ID_CURRENT_TOUT_REACHED,
 CALLBACK_ID_POST_CREATE_GEOMETRY,
 CALLBACK_ID_PRE_DESTROY_GEOMETRY,
 CALLBACK_ID_END_OF_PATH_REACHED,
 CALLBACK_ID_UI_PRE_TRANSLATION,
 CALLBACK_ID_UI_POST_TRANSLATION,
 CALLBACK_ID_UI_MOUSE_DOWN,
 CALLBACK_ID_UI_MOUSE_UP,
 CALLBACK_ID_UI_RIGHT_MOUSE_DOWN,
 CALLBACK_ID_UI_RIGHT_MOUSE_UP,
 CALLBACK_ID_AGENT_NEW_BEHAVIOR,
 CALLBACK_ID_AGENT_NEW_PATH_SHAPE,
 CALLBACK_ID_AGENT_NEW_FOCUS_CHARACTER,
 CALLBACK_ID_AGENT_ATTACK_OUT_OF_TARGETS,
 CALLBACK_ID_AGENT_ATTACK_NO_VISIBLE_TARGETS,
 CALLBACK_ID_AGENT_PURSUE_TARGET_REACHED,
 CALLBACK_ID_AGENT_PURSUE_TARGET_LOST,
 CALLBACK_ID_AGENT_MINGLE_TARGET_REACHED,
 CALLBACK_ID_AGENT_MINGLE_TARGET_LOST,
 CALLBACK_ID_AGENT_WANDER_TARGET_REACHED,
 CALLBACK_ID_AGENT_WANDER_TARGET_LOST,
 CALLBACK_ID_AGENT_FLEE_AREA_LEFT,
 CALLBACK_ID_AGENT_FLEE_AREA_ENTERED,
 CALLBACK_ID_AGENT_TRAVEL_WAYPOINT_REACHED,
 CALLBACK_ID_AGENT_TRAVEL_FORWARD_DEST_REACHED,
 CALLBACK_ID_AGENT_TRAVEL_BACKWARD_DEST_REACHED,
 CALLBACK_ID_AGENT_CROWD_MEMBER_KILLED,
 CALLBACK_ID_AGENT_CROWD_MEMBER_IMPACT,
 CALLBACK_ID_AGENT_NEARBY_SCENE_OBJECT_IMPACT,
 CALLBACK_ID_AGENT_NEARBY_WEAPON_FIRED,
 CALLBACK_ID_AGENT_VEHICLE_COLLISION,
 CALLBACK_ID_AGENT_VEHICLE_NEAR_COLLISION,
 CALLBACK_ID_AGENT_PATH_PLAN_COMPLETED,
 CALLBACK_ID_AGENT_MIND_CURRENT_STATE_CHANGED,

```

        CALLBACK_ID_AGENT_NON_ZERO_REPULSION,
        CALLBACK_ID_AGENT_ZERO_REPULSION
    };
#ifdef CPLUSPLUS_ONLY
    int add_callback(int callback_id,
        diguyCharacterCallback* callback,
        void* callback_params = 0,
        void* callback_user_data = 0);
    int remove_callback(int callback_id,
        diguyCharacterCallback* callback);
    int remove_callback_with_user_data(int callback_id,
        void* callback_user_data);
#endif
    int add_callback_script(int callback_id,
        const char* callback_script,
        const char* callback_script_type = NULL);
    int remove_callback_script(int callback_id,
        const char* callback_script,
        const char* callback_script_type = NULL);
    int map_event_handler_to_callback_id(int callback_id,
        const char* handler_name);
    int unmap_event_handler_from_callback_id(int callback_id,
        const char* handler_name,
        int unmap_all_matches = 0);
    diguyCallbackReturn manually_invoke_event_handler(const char* handler_name);
    diguyCharacterPath* create_path(const char* name);
    diguyCharacterPath* create_simple_path(const char* name,
        diguyWaypoint* waypoint0,
        diguyWaypoint* waypoint1,
        const char* transition_from_action_name,
        const char* fill_action_name,
        const char* transition_to_action_name);
    diguyCharacterPath* create_path_from_path_shape(const char* name,
        diguyPathShape* source_path_shape,
        const char* transition_from_action_name,
        const char* fill_action_name,
        const char* transition_to_action_name);
    int create_and_force_bridge_path(const char* destination_path_name,
        const char* destination_path_action_bead_name,
        const char* via_action_name);
    int create_and_force_bridge_path_to_point(float x, float y, float z,
        float path_offset,
        const char* via_action_name,
        const char* next_local_path = NULL,
        float orientation = 0.0f);
    int destroy_path(diguyCharacterPath* path);
    int get_num_paths();
    diguyCharacterPath* get_path_at_index(int index);
    const char* get_path_name_at_index(int index);
    int get_current_path_index();
    diguyCharacterPath* get_current_traveled_path();
    int get_current_traveled_path_is(const char* path_name);
    int push_path(const char* path_name);
    int force_path(const char* path_name,
        float max_rampdown_interval = 0.5f);
    int force_partial_path(const char* path_name,
        const char* beginning_action_bead_name,
        float max_rampdown_interval = 0.5f);
    int resume_interrupted_path(float max_rampdown_interval = 0.5f);
    diguyCharacterPath* force_local_path(const char* path_name,
        int use_character_orientation,

```



```

        float rotation = 0.0f,
        int clamp_path = 1);
int set_initial_path(const char* name);
const char* get_initial_path();
void translate_all_paths(float tx, float ty, float tz,
    int reclamp_waypoints = 1);
int rotate_all_paths_about_point(float rz, float rx, float ry,
    float rotation_pt_x, float rotation_pt_y, float rotation_pt_z,
    int reclamp_waypoints = 1);
int rotate_all_paths_about_current_waypoint(float rz, float rx, float ry,
    int reclamp_waypoints = 1);
int jump_to_action_bead(const char* bead_name,
    float max_rampdown_interval = 0.5f);
diguyCharacterPath* find_path(const char* path_name);
int apply_aim_bead(diguyCharacterPathAimBead* bead,
    float how_far = 0.0f);
int apply_decision_bead(diguyCharacterPathDecisionBead* bead,
    float how_far = 0.0f);
int apply_gaze_bead(diguyCharacterPathGazeBead* bead,
    float how_far = 0.0f);
int apply_script_bead(diguyCharacterPathScriptBead* bead,
    float how_far = 0.0f);
int set_distance_along_path(float distance);
float get_distance_along_path();
int leave_path();
int get_nearest_waypoint_index_in_path(const char* path_name);
float get_distance_to_path(const char* path_name,
    const char* action_bead_name = NULL);
diguyGraphicsLink* get_position_link();
int get_num_links();
diguyGraphicsLink* get_link_at_index(int index);
diguyGraphicsLink* find_link(const char* name);
const char* get_link_name_at_index(int link_index);
int get_num_link_shapes(const char* link_name);
const char* get_link_shape_name_at_index(const char* link_name,
    int shape_index);
#ifdef CPLUSPLUS_ONLY
    int set_altitude_function(diguyAltitudeFunction* altitude_function);
    diguyAltitudeFunction* get_altitude_function();
#endif
int set_altitude_function_enabled_in_free_position_mode(int enabled);
int get_altitude_function_enabled_in_free_position_mode();
int set_altitude_function_enabled_in_path_position_mode(int value);
int get_altitude_function_enabled_in_path_position_mode();
int set_altitude_update_rate(int update_rate);
int get_altitude_update_rate();
void set_altitude_interpolation_rate(float interpolation_rate);
float get_altitude_interpolation_rate();
int get_pose_array_size();
#ifdef CPLUSPLUS_ONLY
    int get_pose_in_radians(float* pose_array);
#endif
diguyCharacterPoseOverride* create_pose_override(int override_stage = 3);
int destroy_pose_override(diguyCharacterPoseOverride* po);
int get_num_pose_overrides();
diguyCharacterPoseOverride* get_pose_override_at_index(int index);
diguyCharacterPoseOverride* create_complete_pose_override();
void destroy_complete_pose_override();
int gaze_at_angle(float azimuth,
    float elevation,
    float distance,

```

```

    int is_new_gaze = 1);
int gaze_at_angle_local(float azimuth,
    float elevation,
    float distance = 10000.0f,
    int is_new_gaze = 1);
int gaze_at_point(float tx,
    float ty,
    float tz,
    int is_new_gaze = 1);
int gaze_at_point_local(float tx,
    float ty,
    float tz,
    int is_new_gaze = 1);
int gaze_at_character(const char* target_character_name,
    const char* target_character_link_name = NULL,
    float offset_tx = 0.0f,
    float offset_ty = 0.0f,
    float offset_tz = 0.0f);
int get_gaze_is_active();
int get_gaze_point(float* x, float* y, float* z);
int get_gaze_is_acquired();
int get_gaze_is_steady();
int end_gaze();
int lpoint_at_angle(float azimuth,
    float elevation,
    float distance,
    int is_new_lpoint = 1,
    const char* base_gesture_name = 0);
int lpoint_at_angle_local(float azimuth,
    float elevation,
    float distance = 10000.0f,
    int is_new_lpoint = 1,
    const char* base_gesture_name = 0);
int lpoint_at_point(float tx,
    float ty,
    float tz,
    int is_new_lpoint = 1,
    const char* base_gesture_name = 0);
int lpoint_at_point_local(float tx,
    float ty,
    float tz,
    int is_new_lpoint = 1,
    const char* base_gesture_name = 0);
int lpoint_at_character(const char* target_character_name,
    const char* target_character_link_name = NULL,
    float offset_tx = 0.0f,
    float offset_ty = 0.0f,
    float offset_tz = 0.0f);
int get_lpoint_point(float* x, float* y, float* z);
const char* get_lpoint_base_gesture_name();
int get_lpoint_is_active();
int get_lpoint_is_acquired();
int get_lpoint_is_steady();
int end_lpoint();
int set_gaze_params_to_defaults();
int set_gaze_param_desired_locate_time(float t);
float get_gaze_param_desired_locate_time();
int set_gaze_link_uses_desired_locate_time(const char* link_name,
    int uses_desired_locate_time);
int get_gaze_link_uses_desired_locate_time(const char* link_name);
int set_gaze_link_azimuth_limits(const char* link_name, float min, float max);

```

```

int get_gaze_link_azimuth_limits(const char* link_name,
    float* min,
    float* max);
int set_gaze_link_elevation_limits(const char* link_name,
    float min,
    float max);
int get_gaze_link_elevation_limits(const char* link_name,
    float* min,
    float* max);
int set_gaze_link_azimuth_tolerance(const char* link_name,
    float min,
    float max);
int get_gaze_link_azimuth_tolerance(const char* link_name,
    float* min,
    float* max);
int set_gaze_link_elevation_tolerance(const char* link_name,
    float min,
    float max);
int get_gaze_link_elevation_tolerance(const char* link_name,
    float* min,
    float* max);
int set_gaze_link_max_angular_vel(const char* link_name, float angular_vel);
int get_gaze_link_max_angular_vel(const char* link_name, float* angular_vel);
int set_gaze_link_scale_factor(const char* link_name, float scale_factor);
int get_gaze_link_scale_factor(const char* link_name, float* scale_factor);
int set_link_enabled_for_gaze(const char* link_name, int enabled);
int get_link_enabled_for_gaze(const char* link_name);
int set_gaze_can_change_posture(int enabled);
int get_gaze_can_change_posture();
int set_gaze_can_change_variant(int enabled);
int get_gaze_can_change_variant();
int set_link_enabled_for_lpoint(const char* link_name, int enabled);
int get_link_enabled_for_lpoint(const char* link_name);
int set_nod(float azimuth,
    float elevation,
    float duration = 0.5f);
int nod_head(float duration,
    float nod_count,
    float nod_elevation0 = 15.0f,
    float nod_elevation1 = 0.0f,
    float nod_elevation_out = 0.0f);
int shake_head(float duration,
    float shake_count,
    float shake_azimuth0 = -20.0f,
    float shake_azimuth1 = 20.0f,
    float shake_azimuth_out = 0.0f);
float get_nod_azimuth();
float get_nod_elevation();
int set_aim(float azimuth, float elevation);
int aim_at_angle(float azimuth,
    float elevation,
    float distance,
    int is_new_aim = 1);
int aim_at_angle_local(float azimuth,
    float elevation,
    float distance,
    int is_new_aim = 1);
int aim_at_point(float x, float y, float z, int is_new_aim = 1);
int aim_at_point_local(float x, float y, float z, int is_new_aim = 1);
int aim_at_character(const char* target_character_name,
    const char* target_character_link_name = NULL,

```

```

        float offset_tx = 0.0f,
        float offset_ty = 0.0f,
        float offset_tz = 0.0f);
void end_aim();
int aim_converge(int max_iterations = 30);
float get_aim_azimuth();
int get_aim_trajectory(float* muzzle_x, float* muzzle_y, float* muzzle_z,
    float* far_x, float* far_y, float* far_z);
float get_aim_elevation();
int get_aim_is_possible();
int get_aim_is_acquired();
int get_aim_is_steady();
void set_aim_algorithm(int aim_algorithm);
int get_aim_algorithm();
int get_is_aiming();
void set_aim_param_acquire_factor(float factor);
float get_aim_param_acquire_factor();
void set_aim_param_new_aim_factor(float factor);
float get_aim_param_new_aim_factor();
void set_aim_param_new_aim_duration(float duration);
float get_aim_param_new_aim_duration();
void set_aim_param_max_per_frame_adj_factor(float factor);
float get_aim_param_max_per_frame_adj_factor();
void set_aim_param_azimuth_acquired_range(float range);
float get_aim_param_azimuth_acquired_range();
void set_aim_param_elevation_acquired_range(float range);
float get_aim_param_elevation_acquired_range();
void set_aim_converge_every_frame(int max_iterations = 30);
int get_aim_converge_every_frame();
int fire_weapon(diguyMuzzle muzzle = DIGUY_MUZZLE_DEFAULT);
int fire_weapon_n_times(int count,
    float dt = 0.2f,
    const char* override_sound_name = NULL,
    diguyMuzzle muzzle = DIGUY_MUZZLE_DEFAULT);
int fire_screen_space_bullet(diguyView* view, float x, float y);
int get_weapon_fire_count(diguyMuzzle muzzle = DIGUY_MUZZLE_DEFAULT);
int set_default_weapon_muzzle(diguyMuzzle muzzle);
diguyMuzzle get_default_weapon_muzzle();
const char* get_weapon_muzzle_munition_type(diguyMuzzle muzzle);
const char* get_default_muzzle_munition_type();
int set_weapon_munition_type_override(const char* munition_type);
const char* get_weapon_munition_type_override();
int set_weapon_fires_live_rounds(int weapon_fires_live_rounds);
int get_weapon_fires_live_rounds();
int set_weapon_sound_override(const char* sound_name);
const char* get_weapon_sound_override();
int set_weapon_sound_gain_override(float gain);
float get_weapon_sound_gain_override();
void set_weapon_spread(float spread);
void set_weapon_near_range(float near_range);
void set_weapon_far_range(float far_range);
void set_weapon_hit_ratio(float hit_ratio);
void set_weapon_max_impacts(int val);
int get_accessory1_state();
int get_rifle_state();
int set_weapon_projectile_is_visible(diguyMuzzle muzzle, int is_visible);
int get_weapon_projectile_is_visible(diguyMuzzle muzzle);
int get_weapon_aim_vector(float* muzzle_tx, float* muzzle_ty, float* muzzle_tz,
    float* end_pt_tx = NULL, float* end_pt_ty = NULL, float* end_pt_tz = NULL,
    float* muzzle_rz = NULL, float* muzzle_rx = NULL, float* muzzle_ry = NULL,
    int in_character_local_coordinates = 0,

```

```

    diguyMuzzle muzzle = DIGUY_MUZZLE_DEFAULT);
const char* get_muzzle_link(diguyMuzzle muzzle = DIGUY_MUZZLE_DEFAULT);
int get_muzzle_offset(diguyMuzzle muzzle = DIGUY_MUZZLE_DEFAULT,
    float* offset_x = NULL,
    float* offset_y = NULL,
    float* offset_z = NULL);
int set_weapon_max_azimuth_error(float h_error);
float get_weapon_max_azimuth_error();
int set_weapon_max_elevation_error(float v_error);
float get_weapon_max_elevation_error();
int set_weapon_tracers_enabled(int enabled);
int get_weapon_tracers_enabled();
int get_weapon_rounds_until_tracer(diguyMuzzle muzzle);
int set_weapon_shell_eject_enabled(int enabled);
int get_weapon_shell_eject_enabled();
int set_weapon_smoke_enabled(int enabled);
int get_weapon_smoke_enabled();
int trigger_detonation_on_character(const char* munition_name,
    const char* attacker_name = NULL,
    float radius_override = -1.0f,
    int broadcast_on_network = 1);
int set_face_expression(const char* facex_name, float blend_duration, float strength);
int set_orientation_eyes(float azimuth,
    float elevation,
    float blend_time = 0.0f);
int get_orientation_eyes(float* azimuth, float* elevation);
int set_eye_droop(float droop_amount, float blend_time = 0.0f);
float get_eye_droop();
int blink(float duration_in = 0.1f,
    float duration_out = 0.2f,
    float target_in = 1.0f,
    float target_out = 0.0f,
    float tin = DIGUY_DEFAULT_FLOAT);
int set_automatic_blinks_enabled(int enabled,
    float period = 6.0f,
    float period_variation = 3.0f);
int get_automatic_blinks_enabled();
void set_automatic_blink_duration_in(float duration_in);
float get_automatic_blink_duration_in();
void set_automatic_blink_duration_out(float duration_out);
float get_automatic_blink_duration_out();
void set_automatic_blink_target_in(float target_in);
float get_automatic_blink_target_in();
void set_automatic_blink_target_out(float target_out);
float get_automatic_blink_target_out();
int is_speaking();
int set_current_facefx_animation_group(const char* anim_group);
const char* get_current_facefx_animation_group();
const char* get_facefx_actor();
int facefx_mount_animset(const char* file_name);
int play_facefx_animation(const char* animation,
    float magnitude_scale = 1.0f,
    float duration_scale = 1.0f,
    float blendin_time = 0.1f,
    float blendout_time = 0.1f,
    int loop = 0);
int stop_facefx_animations(float blend_time = 0.25f, int stop_sounds = 1);
int get_num_facefx_nodes();
const char* get_facefx_node_name(int index);
int find_facefx_node_index(const char* node_name);
int set_facefx_node_value(int node_index,

```

```

        float value,
        float blend_time = 0.25f);
float get_facefx_node_value(int node_index);
diguyCharacterGesture* create_gesture(const char* gesture_name);
int destroy_gesture(diguyCharacterGesture* gesture);
int execute_gesture(const char* gesture_name,
    int reps = DIGUY_DEFAULT_INT,
    float overall_duration = DIGUY_DEFAULT_FLOAT,
    float channel_A_weight = DIGUY_DEFAULT_FLOAT);
int execute_1stage_gesture(const char* gesture_name,
    int reps = DIGUY_DEFAULT_INT,
    float stage1_duration = DIGUY_DEFAULT_FLOAT,
    float channel_A_weight = DIGUY_DEFAULT_FLOAT);
int execute_2stage_gesture(const char* gesture_name,
    int reps = DIGUY_DEFAULT_INT,
    float stage1_duration = DIGUY_DEFAULT_FLOAT,
    float stage2_duration = DIGUY_DEFAULT_FLOAT,
    float channel_A_weight = DIGUY_DEFAULT_FLOAT);
int execute_3stage_gesture(const char* gesture_name,
    int reps = DIGUY_DEFAULT_INT,
    float stage1_duration = DIGUY_DEFAULT_FLOAT,
    float stage2_duration = DIGUY_DEFAULT_FLOAT,
    float stage3_duration = DIGUY_DEFAULT_FLOAT,
    float channel_A_weight = DIGUY_DEFAULT_FLOAT);
int execute_table_gesture(const char* gesture_name,
    int reps = DIGUY_DEFAULT_INT,
    const char* loop_action = NULL,
    const char* begin_action = NULL,
    const char* end_action = NULL);
int set_executing_table_gesture_desired_action(const char* gesture_name,
    const char* desired_action);
int end_executing_gesture(const char* gesture_name);
int abort_executing_gesture(const char* gesture_name,
    float max_rampdown_interval);
int end_all_gestures();
int abort_all_gestures(float max_rampdown_interval);
int play_sound(const char* sound_name,
    float sound_gain = 1.0f,
    int sound_loops = 0);
int play_3d_sound(const char* sound_name,
    float sound_gain = 1.0f,
    int sound_loops = 0);
diguySoundInstance* create_sound_instance(const char* sound_name);
int destroy_sound_instance(diguySoundInstance* sound_instance);
int stop_all_sounds(float rampdown_time = 0.0f);
int stop_sound(const char* sound_name, float rampdown_time = 0.0f);
int is_playing_sound(const char* sound_name);
int set_current_voice_actor(const char* voice_actor);
const char* get_current_voice_actor();
int get_DIS_lifeform_state();
int get_DIS_primary_weapon_position();
int get_DIS_secondary_weapon_position();
int set_network_marking(const char* network_marking);
const char* get_network_marking();
int set_is_network_published(int is_network_published);
int get_is_network_published();
int get_is_network_reflected();
int get_is_network_paused();
int get_network_entity_number();
const char* get_action_from_entity_state(int damage_state,
    int life_form_state,

```

```

        int primary_weapon_state,
        float speed,
        int forward);
void set_DIS_damage_state(int state);
int get_DIS_damage_state();
void set_DIS_smoke_state(int state);
int get_DIS_smoke_state();
void set_DIS_flames_state(int state);
int get_DIS_flames_state();
void set_DIS_power_plant_state(int state);
int get_DIS_power_plant_state();
void set_DIS_trailing_effect_state(int state);
int get_DIS_trailing_effect_state();
int call_formation(const char* formation_name,
    int snap_to_position_flag = 0,
    int break_existing_formations_flag = 1);
int break_formation(const char* formation_name = NULL);
diguyCharacterGuide* add_guide(const char* guide_name,
    int automatically_disable_when_acquired = 0);
diguyCharacterGuide* create_guide(const char* guide_name,
    const char* guide_algorithm,
    int automatically_disable_when_acquired = 0);
diguyCharacterGuide* find_or_create_guide(const char* guide_name,
    const char* guide_algorithm,
    int automatically_disable_when_acquired = 0);
diguyCharacterGuide* find_guide(const char* guide_name);
int remove_guide_with_name(const char* guide_name);
int remove_guide(diguyCharacterGuide* guide);
int remove_all_guides();
int remove_all_disabled_guides();
int add_motion_texture(const char* motex_name = "default",
    float texture_gain = 1.0f,
    int end_previous_motex = 1,
    float rampup_interval = 0.2f);
int end_motion_texture(float rampdown_interval = 0.2f);
int set_initial_motion_texture(const char* motex_name);
const char* get_initial_motion_texture();
int set_initial_motion_texture_gain(float motex_gain);
float get_initial_motion_texture_gain();
int set_history_type(diguyHistoryType history_type);
diguyHistoryType get_history_type();
int set_history_size_t(float t);
float get_history_size_t();
int set_history_size_increment_t(float t);
float get_history_size_increment_t();
void set_desired_posture(diguyMotionPosture desired_posture);
diguyMotionPosture get_desired_posture();
void set_desired_variant(diguyMotionVariant desired_variant);
diguyMotionVariant get_desired_variant();
const char* get_action_from_description(diguyMotionDirection direction,
    diguyMotionPosture posture,
    diguyMotionVariant variant,
    float speed,
    diguyParameterComparison comparison = DIGUY_PARAMETER_COMPARISON_NEAREST,
    float comparison_tolerance = DIGUY_DEFAULT_FLOAT,
    int match_posture_before_variant = 1,
    diguyMotionPosture posture_alternate1 = DIGUY_MOTION_POSTURE_UNKNOWN,
    diguyMotionPosture posture_alternate2 = DIGUY_MOTION_POSTURE_UNKNOWN,
    diguyMotionVariant variant_alternate1 = DIGUY_MOTION_VARIANT_UNKNOWN,
    diguyMotionVariant variant_alternate2 = DIGUY_MOTION_VARIANT_UNKNOWN,
    int return_random_match = 0);

```



```

const char* get_turning_action_from_description(diguyMotionDirection direction,
    diguyMotionPosture posture,
    diguyMotionVariant variant,
    float turn_amount,
    diguyParameterComparison comparison = DIGUY_PARAMETER_COMPARISON_NEAREST,
    float comparison_tolerance = DIGUY_DEFAULT_FLOAT,
    int match_posture_before_variant = 1,
    diguyMotionPosture posture_alternate1 = DIGUY_MOTION_POSTURE_UNKNOWN,
    diguyMotionPosture posture_alternate2 = DIGUY_MOTION_POSTURE_UNKNOWN,
    diguyMotionVariant variant_alternate1 = DIGUY_MOTION_VARIANT_UNKNOWN,
    diguyMotionVariant variant_alternate2 = DIGUY_MOTION_VARIANT_UNKNOWN);

int get_num_variables();
diguyVariable* get_variable_at_index(int index);
diguyVariable* find_variable(const char* name);
diguyVariable* find_or_create_variable(const char* name);
int destroy_variable(diguyVariable* variable);
int variable_equal_to_int(const char* varname, int value);
int variable_greater_than_int(const char* varname, int value);
int variable_less_than_int(const char* varname, int value);
int variable_equal_to_float(const char* varname, float value);
int variable_greater_than_float(const char* varname, float value);
int variable_less_than_float(const char* varname, float value);
int variable_equal_to_string(const char* varname, const char* value);
int variable_greater_than_string(const char* varname, const char* value);
int variable_less_than_string(const char* varname, const char* value);
int variable_decrement_value_as_int(const char* varname, int value = 1);
int variable_increment_value_as_int(const char* varname, int value = 1);
int variable_set_value_as_int(const char* varname,
    int value,
    int create_if_doesnt_exist = 1);
int set_ivar(const char* varname,
    int value,
    int create_if_doesnt_exist = 1);
int get_ivar(const char* varname,
    int create_if_doesnt_exist = 1);
int variable_set_value_as_float(const char* varname,
    float value,
    int create_if_doesnt_exist = 1);
int set_fvar(const char* varname,
    float value,
    int create_if_doesnt_exist = 1);
float get_fvar(const char* varname,
    int create_if_doesnt_exist = 1);
int variable_set_value_as_string(const char* varname,
    const char* value,
    int create_if_doesnt_exist = 1);
int set_svar(const char* varname,
    const char* value,
    int create_if_doesnt_exist = 1);
const char* get_svar(const char* varname,
    int create_if_doesnt_exist = 1);
diguyViewLabel* get_name_label();
int set_label_text(const char* label_text);
const char* get_label_text();
int set_label_is_visible(int is_visible);
int get_label_is_visible();
int set_label_text_color(float r, float g, float b, float a);
int set_label_text_color_vec4(diguyVec4f color);
int get_label_text_color(float* r, float* g, float* b, float* a);
diguyVec4f get_label_text_color_vec4();
int set_label_text_shadow_color(float r, float g, float b, float a);

```



```

int set_label_text_shadow_color_vec4(diguyVec4f color);
int get_label_text_shadow_color(float* r, float* g, float* b, float* a);
diguyVec4f get_label_text_shadow_color_vec4();
int set_label_text_shadow_is_visible(int is_visible);
int get_label_text_shadow_is_visible();
int set_label_background_color(float r, float g, float b, float a);
int set_label_background_color_vec4(diguyVec4f color);
int get_label_background_color(float* r, float* g, float* b, float* a);
diguyVec4f get_label_background_color_vec4();
void set_label_link_offset_override(float offset);
float get_label_link_offset_override();
int set_label_shows_character_state(int show_state);
int get_label_shows_character_state();
diguyImpact* get_last_impact_record();
int get_last_impact_number();
int get_last_impact_been_processed();
void set_last_impact_been_processed();
void set_max_unprocessed_impacts(int max_unprocessed_impacts);
int get_max_unprocessed_impacts();
const char* get_last_impact_link();
int get_last_impact_link_is(const char* link_name);
int get_last_impact_is_from_vehicle();
int get_last_impact_speed_is_greater_than(float impact_speed);
void set_is_immune_to_impacts(int immune);
int get_is_immune_to_impacts();
int begin_appearance_effect(const char* effect,
    const char* override_link = NULL,
    float override_scale = DIGUY_DEFAULT_FLOAT,
    float override_offset_x = DIGUY_DEFAULT_FLOAT,
    float override_offset_y = DIGUY_DEFAULT_FLOAT,
    float override_offset_z = DIGUY_DEFAULT_FLOAT,
    float override_duration = DIGUY_DEFAULT_FLOAT);
int end_appearance_effect(const char* effect,
    const char* override_link = NULL);
int stop_appearance_effect(const char* effect,
    const char* override_link = NULL);
int stop_all_appearance_effects();
int appearance_effect_is_active(const char* effect,
    const char* override_link = NULL);
diguyChainSimulation* get_chain_simulation();
diguyVehicleController* get_vehicle_controller();
int get_vehicle_body_orientation(float* rz, float* rx, float* ry);
int set_render_mode_shader(const char* shader_name,
    const char* render_mode = NULL);
void reset_render_mode_shader_to_default(const char* render_mode = NULL);
const char* get_render_mode_shader(const char* render_mode = NULL);
diguyGraphicsShaderProgram* get_shader_program();
int set_temporary_current_shader(const char* shader_name);
void reset_current_shader_to_default();
void trigger_shader_refresh();
diguyGraphicsShaderInstance* get_shader_instance();
void reset_shader_instance_variables_to_initial_values();
void reset_shader_instance_variables_to_program_defaults();
int agent_attack_character(const char* character_name);
int agent_attack_group(const char* group_name);
int agent_set_attack_zone_enabled(int use_attack_zone);
int agent_get_attack_zone_enabled();
int agent_set_attack_zone(float x, float y, float z, float radius);
int agent_get_attack_zone(float* x, float* y, float* z, float* radius);
int agent_flee_character(const char* character_name,
    float distance,

```

```

    int danger_level = 50);
int agent_flee_group(const char* group_name,
    float distance,
    int danger_level = 50);
int agent_flee_crowd(const char* crowd_name,
    float distance,
    int danger_level = 50);
int agent_flee_location(float x, float y, float z,
    float distance,
    int danger_level = 50);
void agent_remove_all_flee_objects();
int agent_mingle_in_region(const char* region_name = "(current)",
    const char* subregion_name = "base");
int agent_wander_region(const char* region_name = "(current)",
    const char* subregion_name = "base");
int agent_pursue_character(const char* character_name);
int agent_pursue_character_with_offset(const char* character_name,
    float offset_x,
    float offset_y);
int agent_pursue_group(const char* group_name);
int agent_call_formation(const char* formation_name,
    float target_x,
    float target_y);
int agent_travel_behavior_path(const char* direction = NULL,
    const char* at_end = NULL,
    int waypoint_index = -1,
    float offset_y = 0.5f);
int agent_travel_path_shape(const char* path_shape_name,
    const char* direction = NULL,
    const char* at_end = NULL,
    int waypoint_index = -1,
    float offset_y = 0.5f);
int agent_travel_region_border(const char* region_name = NULL,
    const char* subregion_name = "base");
int agent_move_to_point(float x, float y, float z,
    const char* via_region = NULL,
    const char* subregion_name = "base");
int agent_move_to_point_bg(float x, float y, float z,
    const char* via_region = NULL,
    const char* subregion_name = "base");
int agent_move_to_point_via_subregions(float x, float y, float z,
    const char* via_region = NULL,
    int preferred_subregions_mask = DIGUY_SUBREGION_MASK_NONE,
    float cost_bias_for_preferred_regions = 0.25f,
    float cost_bias_for_neutral_regions = 1.1f,
    int repulsed_regions_mask = DIGUY_SUBREGION_MASK_NONE,
    float cost_bias_for_repulsed_regions = 10.0f,
    int cost_rule = 0);
int agent_move_to_point_via_subregions_bg(float x, float y, float z,
    const char* via_region = NULL,
    int preferred_subregions_mask = DIGUY_SUBREGION_MASK_NONE,
    float cost_bias_for_preferred_regions = 0.25f,
    float cost_bias_for_neutral_regions = 1.1f,
    int repulsed_regions_mask = DIGUY_SUBREGION_MASK_NONE,
    float cost_bias_for_repulsed_regions = 10.0f,
    int cost_rule = 0);
int get_path_planning_result();
int get_is_path_planning();
int agent_move_to_region(const char* region,
    const char* subregion_name = "base",
    const char* via_region = NULL);

```

```

int agent_move_to_region_via_subregions(const char* region_name,
    const char* subregion_name = "base",
    const char* via_region = NULL,
    float cost_bias_for_preferred_regions = 0.25f,
    int preferred_subregions_mask = DIGUY_SUBREGION_MASK_NONE,
    float cost_bias_for_neutral_regions = 1.1f,
    int repulsed_regions_mask = DIGUY_SUBREGION_MASK_NONE,
    float cost_bias_for_repulsed_regions = 10.0f);
int agent_path_follow(const char* path_name);
int agent_resume_path_follow();
diguyCharacterPath* agent_get_path_follow_path();
float agent_get_path_follow_path_t();
int agent_get_path_follow_path_t_paused();
diguyCrowd* agent_get_crowd();
int agent_begin_behavior(const char* behavior_name);
int agent_stop_behavior();
int agent_set_current_focus_character(const char* character_name);
diguyCharacter* agent_get_current_focus_character();
int agent_set_current_focus_group(const char* group_name);
diguyAgentParams* agent_get_initial_params();
diguyAgentParams* agent_get_current_params();
const char* agent_get_current_params_profile_name();
int agent_set_current_params_from_profile(const char* profile_name,
    int retain_focus_objects = 1,
    int retain_navigation_values = 1,
    int retain_behavior = 0);
int agent_set_initial_params_from_profile(const char* profile_name);
void agent_set_current_params_from_initial_params();
void agent_set_current_behavior_region_border_is_solid(int is_solid);
void agent_move_on();

int agent_set_auto_variant_selection(int selection_enabled);
int agent_get_auto_variant_selection();
#ifdef CPLUSPLUS_ONLY
int agent_set_feeler_function(diguyFeelerFunction* feeler_function);
diguyFeelerFunction* agent_get_feeler_function();
#endif
int create_lua_mind(const char* base_class, const char* init_arg1 = NULL,
    const char* init_arg2 = NULL);
const char* get_mind_name();
const char* get_mind_current_state_name();
void do_mind_current_state_changed_callback();
int set_sleep_timer(float sleep_length);
float get_sleep_timer();
int set_heartbeat_dt(float heartbeat_dt);
int add_secondary_heartbeat(const char* function_name,
    float heartbeat_dt);
int remove_secondary_heartbeat(const char* function_name);
int set_secondary_heartbeat_dt(const char* function_name, float heartbeat_dt);
int add_wakeup_callback(int callback);
int remove_wakeup_callback(int callback);
int remove_all_wakeup_callbacks();
int has_wakeup_callback(int callback);
int agent_accept_message(const char* sender,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);
int agent_broadcast_message(float radius,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);

```

```

int agent_broadcast_message_to_group(const char* group_name,
    float radius,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);
const char* get_lua_object_name();
void set_mind_field_as_string(const char* field_name,
    const char* field_value);
const char* get_mind_field_as_string(const char* field_name,
    int warn_if_no_field = 1);
void set_mind_field_as_float(const char* field_name, float field_value);
float get_mind_field_as_float(const char* field_name,
    int warn_if_no_field = 1,
    int* found_field = NULL);
void set_mind_field_as_int(const char* field_name, int field_value);
int get_mind_field_as_int(const char* field_name,
    int warn_if_no_field = 1,
    int* found_field = NULL);
const char* evaluate_mind_function(const char* function_name,
    const char* argument = NULL,
    int has_return_string = 0);
const char* get_ai_state_string(int include_stack = 1, int include_locals = 1);
int check_character_visibility(diguyCharacter* character,
    int los_check_type = 0);
int check_point_visibility(float x, float y, float z);
#ifdef CPLUSPLUS_ONLY
int set_point_los_function(diguyPointLOSFunction* point_los_function);
diguyPointLOSFunction* get_point_los_function();
int set_character_los_function(diguyCharacterLOSFunction* character_los_function);
diguyCharacterLOSFunction* get_character_los_function();
#endif
int set_initial_character_simulator(diguyCharacterSimulator character_simulator);
diguyCharacterSimulator get_initial_character_simulator();
int set_current_character_simulator(diguyCharacterSimulator character_simulator);
diguyCharacterSimulator get_current_character_simulator();
int set_initial_physics_simulation_mode(diguyPhysicsSimulationMode physics_simulation);
diguyPhysicsSimulationMode get_initial_physics_simulation_mode();
int set_current_physics_simulation_mode(diguyPhysicsSimulationMode physics_simulation);
diguyPhysicsSimulationMode get_current_physics_simulation_mode();
int set_physics_collision_group(diguyPhysicsCollisionGroup collision_group);
diguyPhysicsCollisionGroup get_physics_collision_group();
void set_stop_physics_sim_at_t(float t);
void set_stop_physics_sim_in_t_seconds(float relative_t);
float get_stop_physics_sim_at_t();
void apply_impulse_to_physics_sim(const char* link,
    float impulse_magnitude_x,
    float impulse_magnitude_y,
    float impulse_magnitude_z,
    float link_offset_x = 0.0f,
    float link_offset_y = 0.0f,
    float link_offset_z = 0.0f,
    int set_simulation_drives_character = 1);
void apply_impulse_to_physics_sim_from_world_position(const char* link,
    float impulse_magnitude_x,
    float impulse_magnitude_y,
    float impulse_magnitude_z,
    float impulse_generator_position_x,
    float impulse_generator_position_y,
    float impulse_generator_position_z,
    float max_distance = 20.0f,
    float impulse_rolloff = 0.0f,

```

```

        float link_offset_x = 0.0f,
        float link_offset_y = 0.0f,
        float link_offset_z = 0.0f,
        int set_simulation_drives_character = 1);
static int set_texture_load_behavior(diguyGraphicsTextureLoadingBehavior mode);
static diguyGraphicsTextureLoadingBehavior get_texture_load_behavior();
int get_has_bump_maps();
int get_action_num_channels(const char* action_name);
int set_action_channel_weight(const char* action_name,
    char channel,
    float weight);
float get_action_channel_weight(const char* action_name,
    char channel);
float get_time_to_reach_desired_action();
const char* get_type_map_field(diguyCharacterTypeMapField field);
const char* get_type_map_field_using_string(const char* field);
int merge_object(const char* object);
int is_active_character_in_group_visible(const char* group_name,
    float max_distance_to_check = 100000.0f);
int get_local_space_bounding_box(float* x1, float* y1, float* z1,
    float* x2, float* y2, float* z2);
int get_geometry_tag(int tag,
    int world_space,
    float* x1, float* y1, float* z1, float* radius);
void set_mind_always_updates(int force_update);
int get_mind_always_updates();
float get_angle_to_path(const char* path_name,
    const char* action_bead_name = NULL);
const char* get_aim_at_link();
int get_aim_at_link_offset(float* offset_x, float* offset_y, float* offset_z);
const char* get_gaze_at_link();
int get_gaze_at_link_offset(float* offset_x, float* offset_y, float* offset_z);
void set_scene_graph_mask(unsigned int mask);
unsigned int get_scene_graph_mask();
int set_use_texture_variations(int value);
int get_use_texture_variations();
int set_emit_light(int emit_light);
int get_emit_light();
static int set_max_shader_quality_level(int quality_level);
static int get_max_shader_quality_level();
diguyGraphicsLink* get_base_link();

void set_apply_actor_scale_to_action_bead_xy_travel(int apply_scale_flag);
int get_apply_actor_scale_to_action_bead_xy_travel();
int set_default_weapon_sound(const char* sound_name);
const char* get_default_weapon_sound();
int set_default_weapon_sound_gain(float gain);
float get_default_weapon_sound_gain();
int set_weapon_munition_type(const char* munition_type);
const char* get_weapon_munition_type();
int set_weapon_horizontal_error(float h_error);
float get_weapon_horizontal_error();
int set_weapon_vertical_error(float v_error);
float get_weapon_vertical_error();
int begin_legacy_appearance_effect(const char* effect,
    const char* override_link = "",
    float override_scale = DIGUY_DEFAULT_FLOAT,
    float override_offset_x = DIGUY_DEFAULT_FLOAT,
    float override_offset_y = DIGUY_DEFAULT_FLOAT,
    float override_offset_z = DIGUY_DEFAULT_FLOAT);
int end_legacy_appearance_effect(const char* effect);

```

```
int legacy_appearance_effect_is_active(const char* effect);
int merge_asset(const char* asset);
void set_DIS_smoke_trail_state(int state);
int get_DIS_smoke_trail_state();
int speak(const char* string);
int set_viseme(const char* viseme_name, float weight);
float get_viseme(const char* viseme_name);
int set_orientation_eye_left(float azimuth, float elevation);
int set_orientation_eye_right(float azimuth, float elevation);
float get_azimuth_eye_left();
float get_elevation_eye_left();
float get_azimuth_eye_right();
float get_elevation_eye_right();
```

General Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.2 function diguyCharacter::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

5.12.3 function diguyCharacter::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

5.12.4 function diguyCharacter::get_type_name

Prototype:

```
const char* get_type_name();
```

Description:

Returns the type name of the object. This pointer will never be NULL.

5.12.5 function diguyCharacter::get_uid

Prototype:

```
long get_uid();
```

Description:

All characters are assigned a unique identifier, or uid. This function returns this character's uid.

Note: unique identifiers will change between DI-Guy runs!

5.12.6 function diguyCharacter::get_index

Prototype:

```
int get_index();
```

Description:

Returns the index of the object. This index may change if characters are created or destroyed, so it should not be considered a unique identifier for the character. See [get_uid\(\)](#).

5.12.7 function diguyCharacter::update

Prototype:

```
int update(float t, int full_update = 1);
```

Description:

This function updates the character. The character's position and pose will be set to be what it will be (or was) at the specified t.

If the scenario has control of t (return value of [get_t_controlled_by_scenario_t\(\)](#) is 0), then this function's effects will be overridden the next time [diguyScenario::update\(\)](#) is called with a new t.

Note: [diguyScenario::update\(\)](#) must be called as or more frequently than [diguyCharacter::update\(\)](#) for proper DI-Guy operation. [diguyCharacter::update\(\)](#) is intended for load management use, where the user intends to update far away or out-of-frustum characters less frequently than the overall scenario. This is done in conjunction with [set_t_controlled_by_scenario_t\(0\)](#) for the character. [diguyCharacter::update\(\)](#) is not intended as a replacement for

[diguyScenario::update\(\)](#).

Returns:

0 on success, -1 on failure

Arguments:

`t` new time for the character
`full_update` set to 1 to cause full update of character

5.12.8 function `diguyCharacter::set_character_type`

Prototype:

```
void set_character_type(const char* type);
```

Description:

This function sets the current type of this character.

Arguments:

`type` name of the new desired type

See Also:

[get_character_type_string\(\)](#)

5.12.9 function diguyCharacter::get_character_type_string

Prototype:

```
const char* get_character_type_string();
```

Description:

Returns the type of the character (never NULL).

Character types specify the types of motions the character will be able to perform. Some characters can move like soldiers, others can move like civilians, still others can move like specific animals.

The character type is explicitly specified in the call to `diguyScenario::create_character()`.

Examples of character types are:

- soldier
- male_pedestrian
- horse
- prop

5.12.10 function diguyCharacter::get_character_class

Prototype:

```
const char* get_character_class();
```

Description:

Returns the class of the character (never NULL).

Character classes are a higher level description than character types, and allow for more generalized logic. For example, instead of having something happen when one of a large set of specific character types enters a region, something can happen when any human enters the region.

Character classes are not specified directly, but are instead derived from the character type. (In DI-Guy, all soldiers are humans, for example.)

Examples of character classes are:

- human
- vehicle
- object

· animal

5.12.11 function diguyCharacter::get_scenario

Prototype:

```
diguyScenario* get_scenario();
```

Description:

Returns the scenario that this character is a part of.

5.12.12 function diguyCharacter::set_enabled

Prototype:

```
int set_enabled(int enabled);
```

Description:

This is the top-level on/off switch for the character. A character that is not enabled has very little processing overhead: it is neither updated during an `update()` call, nor is it drawn during a `draw()` call. A disabled character will remain disabled until `set_enabled(1)` is called on it; nothing else implicitly re-enables the character.

There are several other `diguyCharacter` functions that have somewhat similar effects that may be more appropriate at times:

- ◇ `set_current_tin()` and `set_current_tout()`: These functions show and animate the character starting at time tin ("T In"), and ending at time tout ("T Out")
- ◇ `set_invisible_flag()`: This function disables drawing of the character, but the character still updates and changes position.

Refer to the documentation of these functions for more information.

Returns:

always return 0

5.12.13 function diguyCharacter::get_enabled**Prototype:**

```
int get_enabled();
```

Description:

Returns whether the character is enabled, as set by the `set_enabled()` call. New characters are enabled by default.

Returns:

1 if the character is enabled, 0 if not

5.12.14 function diguyCharacter::set_drawn_by_scenario_flag**Prototype:**

```
int set_drawn_by_scenario_flag(int flag);
```

Description:

Sets whether this character is drawn automatically by the scenario.

OpenGL Version:

Direct3D Version:

If this flag is set to 0, it is up to the user to call the character `draw_pass1()` and `draw_pass2()` calls at the appropriate time.

Arguments:

<code>flag</code>	pass 1 for character drawing to automatically be handled by scenario (this is the default); 0 for it not to be

Returns:

0 on success, -1 on failure

5.12.15 function diguyCharacter::get_drawn_by_scenario_flag

Prototype:

```
int get_drawn_by_scenario_flag();
```

Description:

Returns whether character drawing is done automatically by the scenario.

Returns:

1 if character drawing is done by the scenario; 0 if not

See Also:

[set_drawn_by_scenario_flag\(\)](#)

5.12.16 function diguyCharacter::set_invisible_flag

Prototype:

```
void set_invisible_flag(int invisible_flag);
```

Description:

This function makes the character invisible; i.e., `draw( )` calls for the character will have no effect. Note that the character will still update its position and pose during `update( )` calls. To more thoroughly disable a character, consider the functions `set_enabled( )` and `set_current_tout( )`.

Arguments:

`invisible_flag` pass 1 to make character invisible, 0 to make it visible

Callable from:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.17 function diguyCharacter::get_invisible_flag

Prototype:

```
int get_invisible_flag();
```

Returns:

the character's invisible flag, as set by `set_invisible_flag( )`

5.12.18 function diguyCharacter::get_is_active

Prototype:

```
int get_is_active();
```

Description:

Returns whether the character is active. A character is active if:

- ◇ it is enabled as per the `set_enabled()` call
- ◇ it is within its tin/tout interval as set by the `set_current_tin()` and `set_current_tout()` calls
- ◇ it is not "network paused" (`get_is_network_paused()` returns 0)

Returns:

1 if the character is active, 0 if not

Callable from:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.19 function diguyCharacter::get_is_drawn

Prototype:

```
int get_is_drawn();
```

Description:

Returns whether the character will be drawn during scenario and character `draw()` calls. A character will be drawn if:

- ◇ it is enabled as per the `set_enabled()` call
- ◇ it is within its tin/tout interval as set by the `set_current_tin()` and `set_current_tout()` calls
- ◇ its invisible flag is 0, as set by the `set_invisible_flag()` call

There is no corresponding `set_is_drawn()` function; use `set_invisible_flag()` to explicitly make a character not be drawn.

Returns:

1 if the character will be drawn, 0 if not

5.12.20 function diguyCharacter::get_is_temporary

Prototype:

```
int get_is_temporary();
```

Returns:

1 if the character is temporary (created by `diguyScenario::create_temporary_character()` call; 0 if not

5.12.21 function diguyCharacter::get_is_recycled

Prototype:

```
int get_is_recycled();
```

Description:

This function returns 1 if the character has been recycled from the character recycle bin, 0 if not.

See `diguyScenario::send_character_to_recycle_bin()` and `diguyScenario::retrieve_character_from_recycle_bin()`.

5.12.22 function diguyCharacter::set_t_controlled_by_scenario_t**Prototype:**

```
int set_t_controlled_by_scenario_t(int t_controlled_by_scenario_t);
```

Description:

This function sets how the character's t (time) is controlled. If 1, the character's t will be set by the scenario on each call to `diguyScenario::update()`. If 0, the character should be updated by calls to `update()`.

Arguments:

`t_controlled_by_scenario_t` 1 for controlled by scenario, 0 for manual update

Returns:

0 on success, -1 on failure

5.12.23 function diguyCharacter::get_t_controlled_by_scenario_t**Prototype:**

```
int get_t_controlled_by_scenario_t();
```

Returns:

whether character's t is controlled by the scenario; see `set_t_controlled_by_scenario_t()`

5.12.24 function diguyCharacter::set_initial_tin

Prototype:

```
int set_initial_tin(float tin, int override_time_warning = 0);
```

Description:

This function sets the initial tin ("T In") time of the character. See [set_current_tin\(\)](#) for a discussion of how tin and tout times affect the character.

When the scenario is reset or loaded from a .dss file, the current tin value will be set to this initial tin value.

Note that this function cannot be called once the scenario has begun running ([diguyScenario::get_t\(\)](#) returns > 0).

Note that both tin and tout times are scenario-relative times, not character-relative.

This call makes an implicit call to [set_current_tin\(\)](#) with the same value.

Arguments:

<code>tin</code>	new value of initial tin in seconds

Returns:

0 on success, -1 on failure

5.12.25 function diguyCharacter::set_initial_tout

Prototype:

```
int set_initial_tout(float tout);
```

Description:

This function sets the initial tout ("T Out") time of the character. See [set_current_tin\(\)](#) for a discussion of how tin and tout times affect the character.

When the scenario is reset or loaded from a .dss file, the current tout value will be set to this initial tout value.

Note that this function cannot be called once the scenario has begun running ([diguyScenario::get_t\(\)](#) returns > 0).

Note that both tin and tout times are scenario-relative times, not character-relative.

This call makes an implicit call to [set_current_tout\(\)](#) with the same value.

Arguments:

tout	new value of initial tout in seconds

Returns:

0 on success, -1 on failure

5.12.26 function diguyCharacter::set_current_tin

Prototype:

```
int set_current_tin(float tin);
```

Description:

This function sets the current tin ("T In") time of the character. The character will not be visible nor will it start moving or animating until the scenario reaches this time.

There are two ways the tin and tout times of the character can be used:

The first is to set them to their desired values when the scenario begins. If, for example, it is known that this character should only be visible and active from 60 seconds to 80 seconds, the tin and tout times can be set to 60 and 80, respectively. The scenario will then spend very little time for character updates and draws outside of these limits.

The second is to use them to quickly "trigger" a character to become visible and start moving. This is done by setting the initial values of tin and tout to very high values, and then changing them to lower values when the character should become active. Consider the following example: This character is waiting just inside a doorway, and should emerge when another character comes close. If the tin time of the character is very high, very little time will be taken for this character's updates and draws. When the other character does come close, this character's tin time can be set to "now" (the scenario's current t, as returned by `diguyScenario::get_t()`). This character will then become visible and start moving out of the doorway.

The default current tin time is the current time of the scenario when the character is created.

Note that both tin and tout times are scenario-relative times, not character-relative.

Arguments:

<code>tin</code>	new value of current tin in seconds

Returns:

0 on success, -1 on failure

5.12.27 function diguyCharacter::set_current_tin_to_now

Prototype:

```
int set_current_tin_to_now();
```

Description:

Same as [set_current_tin\(\)](#), but time is set to current scenario time.

Returns:

0 on success, -1 on failure

Callable from:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.28 function diguyCharacter::get_current_tin

Prototype:

```
float get_current_tin();
```

Description:

This function returns the current tin time of the character. See [set_current_tin\(\)](#).

Returns:

Current tin time in seconds.

5.12.29 function diguyCharacter::set_current_tout**Prototype:**

```
int set_current_tout(float tout,
    int disable_character_at_tout = 0);
```

Description:

This function sets the current tout ("T Out") time of the character. If tout is enabled the character will become invisible and will stop moving and animating when the scenario reaches this time.

Note that both tin and tout times are scenario-relative times, not character-relative.

Default tout time is 60000 seconds.

Arguments:

tout	new value of current tout in seconds
disable_character_at_tout	this argument is present to keep the function prototype will remain backward compatible; it has no effect

5.12.30 function diguyCharacter::set_current_tout_to_now**Prototype:**

```
int set_current_tout_to_now();
```

Description:

Same as `set_current_tout()`, but time is set to current scenario time.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.31 function diguyCharacter::get_current_tout**Prototype:**

```
float get_current_tout();
```

Description:

This function returns the current tout ("T Out") time of the character in seconds. See [set_current_tout\(\)](#).

5.12.32 function diguyCharacter::set_tout_enabled**Prototype:**

```
void set_tout_enabled(int tout_enabled);
```

Description:

This function sets what will happen when the character reaches its tout time. If set to 0, the character will ignore the tout time and continue indefinitely.

5.12.33 function diguyCharacter::get_tout_enabled

Prototype:

```
int get_tout_enabled();
```

Description:

This function returns the whether tout is enabled.

See [set_tout_enabled\(\)](#).

Returns:

1 if tout enabled, 0 if not

5.12.34 function diguyCharacter::set_careful_interpolation_threshold

Prototype:

```
int set_careful_interpolation_threshold(float threshold);
```

Description:

When the motion data of a character is interpolated a quick interpolation method can be used which may yield incorrect results, or an expensive interpolation method can be used which will yield correct results at the expense of time. The threshold at which the interpolation method changes is specified by this function.

Arguments:

`threshold` float between 0 (careful never) and 1 (careful always);

default is 0.7

DI-Guy compares the careful interpolation threshold to the cosine of an angle. Both the threshold and the cosine are dimensionless and thus have no units.

Returns:

0 on success, -1 on failure

5.12.35 function `diguyCharacter::get_careful_interpolation_threshold`

Prototype:

```
float get_careful_interpolation_threshold();
```

Returns:

most recent setting of the careful interpolation threshold; see
[`set\_careful\_interpolation\_threshold\(\)`](#)

5.12.36 function diguyCharacter::set_motion_interpolation_flag**Prototype:**

```
int set_motion_interpolation_flag(int flag);
```

Description:

Sets whether the motion data of the character is interpolated. Interpolation is disabled by default for better performance. For applications achieving high frame rates, smoother motion may be achieved by turning interpolation on.

Arguments:

flag	0 by default

Returns:

0 on success, -1 on failure

5.12.37 function diguyCharacter::get_motion_interpolation_flag**Prototype:**

```
int get_motion_interpolation_flag();
```

Returns:

most recent setting of the motion interpolation flag; see [set_motion_interpolation_flag\(\)](#)

5.12.38 function diguyCharacter::set_scale**Prototype:**

```
int set_scale(float scale_x,
             float scale_y,
             float scale_z);
```

Description:

Sets the scale of the character on each of the three major axes.

Arguments:

scale_x, the factor by which to scale along the given axis
scale_y,
scale_z

Returns:

0 on success, -1 on failure

5.12.39 function diguyCharacter::get_scale**Prototype:**

```
int get_scale(float* scale_x,
             float* scale_y,
             float* scale_z);
```

Description:

Returns the scale of the character.

Arguments:

SX, SY, scale of the character along each axis
SZ

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.40 function diguyCharacter::set_parent**Prototype:**

```
int set_parent(const char* parent_name,
               const char* parent_link_name = "");
```

Description:

Sets the parent of this character to be the passed character. This means that the coordinate system of this character will be local to the parent character rather than the world coordinate system.

Arguments:

parent_name	character to which this character should be attached; pass NULL to have no parent and be attached to the world
parent_link_name	name of the link to which this character's position link should attach; pass "" to attach to the parent's position link

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.41 function diguyCharacter::get_parent

Prototype:

```
diguyCharacter* get_parent();
```

Description:

Gets the parent of this character.

Returns:

pointer to type [diguyCharacter](#); NULL if no parent

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.42 function diguyCharacter::get_parent_link_name

Prototype:

```
const char* get_parent_link_name();
```

Description:

Gets the link name that the character is parented to, if any.

Returns:

name of link, NULL if no parent

5.12.43 function diguyCharacter::unset_parent

Prototype:

```
int unset_parent();
```

Description:

Detaches a character from its parent and reattaches it to the world. This is equivalent to `diguyCharacter::set_parent(NULL)`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.44 function diguyCharacter::get_num_children

Prototype:

```
int get_num_children();
```

Description:

Gets the number of children the character has.

5.12.45 function diguyCharacter::get_child_at_index**Prototype:**

```
diguyCharacter* get_child_at_index(int index);
```

Description:

This function returns a pointer to the nth child of the character.

Returns:

pointer of type [diguyCharacter](#); NULL if no child at the specified index

Arguments:

index	index of the child; indices start at 0

5.12.46 function diguyCharacter::get_is_scene_object**Prototype:**

```
int get_is_scene_object();
```

Description:

Gets if a character is a static object, by default any blitzed . in props are.

Returns:

1 if true, 0 if false

5.12.47 function diguyCharacter::set_is_scene_object**Prototype:**

```
int set_is_scene_object(int is_scene_object);
```

Description:

Sets if a character is a static object. By default any blitzed in props are.

Arguments:

`is_scene_object` pass 1 to make the object a scene object, 0 to not

Returns:

0 on success, -1 on failure

5.12.48 function diguyCharacter::get_bounding_radius**Prototype:**

```
float get_bounding_radius();
```

Description:

Gets the bounding radius currently used for this character for purposes of culling it from the draw operation.

Returns:

The current draw-culling bounding radius for the character, in meters.

5.12.49 function diguyCharacter::get_default_bounding_radius**Prototype:**

```
float get_default_bounding_radius();
```

Description:

Gets the default draw-culling bounding radius for this character, as specified in its actor cfg file. See actor_vehicle.cfg for an example.

Returns:

The default draw-culling bounding radius for the character.

5.12.50 function diguyCharacter::set_bounding_radius**Prototype:**

```
void set_bounding_radius(float f);
```

Description:

Sets the current draw-culling bounding radius to be used for this character.

5.12.51 function diguyCharacter::get_time_to_transition

Prototype:

```
float get_time_to_transition();
```

Description:

This function returns the number of seconds before the blend into the next motion begins. If the character is already in a blend between motions the function returns 0.

Note that this function is of limited practical use; in most cases [get_time_to_reach_desired_action\(\)](#) returns information that is more useful.

Returns:

time to next blend, in seconds

Mode Restrictions:

◇ This function can only be called in free action mode.

5.12.52 function diguyCharacter::get_action_mode

Prototype:

```
diguyCharacterMode get_action_mode();
```

Returns:

the action mode of the character

5.12.53 function diguyCharacter::get_position_mode**Prototype:**

```
diguyCharacterMode get_position_mode();
```

Returns:

the position mode of the character

5.12.54 function diguyCharacter::set_t_scale_factor**Prototype:**

```
int set_t_scale_factor(float t_scale_factor);
```

Description:

This function speeds up or slows down the character. Setting this value greater than 1 will cause character actions to be played back at a slower pace, setting to less than 1 will cause character actions to be played at a faster pace.

This will override other factors that may scale the time scale factor of a character. (For example, smaller scale characters will generally move faster.)

Call [unset_t_scale_factor\(\)](#) to allow other factors such as scale or desired speed to determine the time scale factor.

Mode Restrictions:

◇ This function can only be called in free action mode.

Arguments:

`t_scale_factor` time scale factor

Returns:

0 on success, -1 on failure

5.12.55 function `diguyCharacter::unset_t_scale_factor`

Prototype:

```
int unset_t_scale_factor();
```

Description:

This function removes the manual setting of the time scale factor as set by `set_t_scale_factor()`.

Mode Restrictions:

◇ This function can only be called in free action mode.

Returns:

0 on success, -1 on failure

5.12.56 function**diguyCharacter::is_within_distance_n_of_character****Prototype:**

```
int is_within_distance_n_of_character(const char* character_name,
    float distance);
```

Description:

This function determines whether this character is within the specified distance of another.

Arguments:

character	character to be checked
distance	distance to character in meters

Returns:

1 if within distance; 0 if not

5.12.57 function**diguyCharacter::is_within_distance_n_of_member_of_group****Prototype:**

```
int is_within_distance_n_of_member_of_group(const char* group_name,
    float distance);
```

Description:

This function determines whether this character is within the specified distance of any members of the specified group.

Arguments:

group_name	name of group to be checked
distance	distance in meters

Returns:

1 if member of group is within distance; 0 if not

5.12.58 function diguyCharacter::is_group_member**Prototype:**

```
int is_group_member(const char* group_name);
```

Description:

This function determines whether a character is a member of a particular group.

Also see [diguyCharacterGroup::is_member\(\)](#).

Arguments:

`group_name` name of group to be checked

Returns:

1 if character is in group, 0 if not

5.12.59 function diguyCharacter::get_num_group_memberships**Prototype:**

```
int get_num_group_memberships();
```

Description:

This function returns the number of groups that a character is a member of.

Also see [diguyCharacterGroup::is_member\(\)](#).

5.12.60 function `diguyCharacter::get_group_membership_at_index`

Prototype:

```
diguyCharacterGroup* get_group_membership_at_index(int index);
```

Description:

This function returns a pointer to the nth group that this character is a member of.

Also see `diguyCharacterGroup::is_member()`.

Returns:

pointer of type `diguyCharacterGroup`; NULL if no group at the specified index

Arguments:

<code>index</code>	index of the group; indices start at 0

5.12.61 function `diguyCharacter::get_nearest_active_character`

Prototype:

```
diguyCharacter* get_nearest_active_character(int check_visibility,
      int xy_distance = 0,
      float max_distance_to_check = 100000);
```

Description:

This function will check all characters in the scenario and return the nearest character that is both alive and enabled. An optional argument specifies whether to skip characters that are invisible to the caller because a scene object is in the way. Checking visibility is only an option if an octtree is generated from scene objects. Typically only DI-Guy Scenario does this.

Arguments:

<code>check_visibility</code>	whether to check that characters are visible to calling character; pass 1 to check, 0 to not check
<code>xy_distance</code>	whether to include the z component in distance calculations; pass 1 to check only XY distance, 0 to

check XYZ distance

Returns:

pointer of type `diguyCharacter`; may be NULL

5.12.62 function

`diguyCharacter::get_nearest_active_character_in_group`

Prototype:

```
diguyCharacter* get_nearest_active_character_in_group(const char* group_name,
    int check_visibility = 1,
    int xy_distance = 0,
    float max_distance_to_check = 100000);
```

Description:

This function will check all characters in a group and return the nearest character that is both alive and enabled. An optional argument specifies whether to skip characters that are invisible to the caller because a scene object is in the way. Checking visibility is only an option if an octtree is generated from scene objects. Typically only DI-Guy Scenario does this.

Arguments:

<code>group_name</code>	name of group to check
<code>check_visibility</code>	whether to check that characters are visible to

	calling character; pass 1 to check, 0 to not check
<code>xy_distance</code>	whether to include the z component in distance calculations; pass 1 to check only XY distance, 0 to check XYZ distance

Returns:

pointer of type `diguyCharacter`; may be NULL

5.12.63 function `diguyCharacter::get_random_active_character`

Prototype:

```
diguyCharacter* get_random_active_character(float max_distance = 10.0f,
    int check_visibility = 1,
    int xy_distance = 0);
```

Description:

This function will check all characters in the scenario and return a random character within `max_distance` that is both alive and enabled. An optional argument specifies whether to skip characters that are invisible to the caller because a scene object is in the way. Checking visibility is only an option if an octtree is generated from scene objects. Typically only DI-Guy Scenario does this.

Arguments:

<code>check_visibility</code>	whether to check that characters are visible to calling character; pass 1 to check, 0 to not check
<code>max_distance</code>	how far away the search cut off should be pass in

	-1.0f to use all characters
<code>xy_distance</code>	whether to include the z component in distance calculations; pass 1 to check only XY distance, 0 to check XYZ distance

Returns:

pointer of type `diguyCharacter`; may be NULL

5.12.64 function `diguyCharacter::get_random_active_character_in_group`

Prototype:

```
diguyCharacter* get_random_active_character_in_group(const char* group_name,
    float max_distance = 10,
    int check_visibility = 1,
    int xy_distance = 0);
```

Description:

This function will check all characters in a group and return a random character within `max_distance` that is both alive and enabled. An optional argument specifies whether to skip characters that are invisible to the caller because a scene object is in the way. Checking visibility is only an option if an octree is generated from scene objects. Typically only DI-Guy Scenario does this.

Arguments:

<code>group_name</code>	name of group to check

<code>max_distance</code>	how far away the search cut off should be, pass in -1.0f to use all characters
<code>check_visibility</code>	whether to check that characters are visible to calling character; pass 1 to check, 0 to not check
<code>xy_distance</code>	whether to include the z component in distance calculations; pass 1 to check only XY distance, 0 to check XYZ distance

Returns:

pointer of type `diguyCharacter`; may be NULL

5.12.65 function `diguyCharacter::get_distance_to_character`

Prototype:

```
float get_distance_to_character(diguyCharacter* character);
```

Description:

This function returns the 3D distance from this character to the specified character, in meters. The distance is calculated from the characters' idealized positions.

5.12.66 function diguyCharacter::get_distance_xy_to_character**Prototype:**

```
float get_distance_xy_to_character(diguyCharacter* character);
```

Description:

This function returns the 2D distance in X and Y coordinates only from this character to the specified character, in meters. The distance is calculated from the characters' idealized positions.

5.12.67 function diguyCharacter::get_distance_to_impact**Prototype:**

```
float get_distance_to_impact(diguyImpact* impact);
```

Description:

This function returns the 3D distance from this character to the specified impact, in meters. The distance is calculated from the characters' idealized positions.

5.12.68 function diguyCharacter::get_random_factor**Prototype:**

```
float get_random_factor();
```

Description:

Each character has a "random factor" between 0 and 1. This value is persistent with the character; i.e., the random factor will stay constant for the entire lifetime of the character.

This random factor can be used to add some variability between different characters.

Though each character has a random factor that has a very good chance of being different from all other characters' in the scenario, this is not guaranteed and so should not be used as a unique identifier.

An example of use: Instead of all characters beginning an action or behavior at the same time, have each one begin (`random_factor * 2.0`) seconds from the current time. This will result in more natural looking behavior as characters one by one begin the new behavior in the next two seconds, rather than all beginning at once.

Returns:

the character's random factor, a value between 0 and 1

C++ Only Functions

5.12.69 function diguyCharacter::draw

Prototype:

```
int draw();
```

Description:

This function draws this character in immediate mode graphics environments (see below). This is the equivalent of calling `draw_pass1( )` immediately followed by `draw_pass2( )`.

Returns:

0 on success, -1 on failure

OpenGL Version:

This function immediately draws this character. Either this function or `draw_pass1( )` and `draw_pass2( )` should be called once per frame.

Callable From:

◇ C++

5.12.70 function diguyCharacter::draw_pass1

Prototype:

```
int draw_pass1();
```

Description:

This function, along with [draw_pass2\(\)](#), allows the drawing of opaque and transparent polygons to be separated. This function draws all opaque polygons of this character.

Returns:

0 on success, -1 on failure

OpenGL Version:

This function immediately draws the opaque character parts. Either this function or [draw\(\)](#) should be called once per frame.

Callable From:

◇ C++

5.12.71 function diguyCharacter::draw_pass2

Prototype:

```
int draw_pass2();
```

Description:

Same as [draw_pass1\(\)](#), but draws transparent character parts.

Callable From:

◇ C++

5.12.72 function diguyCharacter::get_graphics_ptr

Prototype:

```
void* get_graphics_ptr();
```

Returns:

A pointer to the character's graphics.

Vega Prime Version:

The return pointer may be cast to point to a `vpDiguyCharacter`, which is derived from `vpObject`.

OpenGL Version:

This function returns `NULL`.

DirectX 9 Version:

This function returns `NULL`.

DI-Guy Graphics API:

This function returns the pointer set from the most recent `set_graphics_ptr()` call.

Callable From:

◇ C++

5.12.73 function diguyCharacter::set_graphics_ptr**Prototype:**

```
int set_graphics_ptr(void* graphics_ptr);
```

Description:

Sets a graphics environment-specific pointer to graphics data. The pointer can be retrieved using `get_graphics_ptr()`.

Arguments:

`graphics_ptr` pointer to graphics data

Returns:

0 on success, -1 on failure

Vega Prime Version:

OpenGL Version:

DirectX 9 Version:

This function should not be called for these environments.

DI-Guy Graphics API Version:

This function allows graphics environment-specific data to be stored for later retrieval by `get_graphics_ptr()`. DI-Guy does not directly use the pointer, beyond returning it using the `get_graphics_ptr()` function.

Callable From:

◇ C++

5.12.74 function diguyCharacter::set_graphics_api_node_ptr**Prototype:**

```
void set_graphics_api_node_ptr(void* node_ptr);
```

Description:

This function sets a generic node pointer that can later be retrieved by the [get_graphics_api_node_ptr \(\)](#) call. The pointer is otherwise not used.

Callback Info:

This function can be safely be called in the
CALLBACK_ID_SET_GRAPHICS_API_NODE_PTR callback.

Arguments:

node_ptr generic void* pointer

Callable From:

- ◇ C++
- ◇ Script

5.12.75 function diguyCharacter::get_graphics_api_node_ptr**Prototype:**

```
void* get_graphics_api_node_ptr();
```

Returns:

pointer set by most recent call to set_graphics_api_node_ptr

Callable From:

◇ C++

5.12.76 function diguyCharacter::set_user_data**Prototype:**

```
int set_user_data(void* user_data);
```

Description:

This function stores a pointer to user data.

Arguments:

<code>user_data</code>	pointer for user's own use; DI-Guy will do nothing to the contents of this pointer beyond passing it back when requested by <code>get_user_data()</code>

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.12.77 function diguyCharacter::get_user_data

Prototype:

```
void* get_user_data();
```

Returns:

The user data pointer set by [set_user_data\(\)](#)

Callable From:

◇ C++

Action Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.12.78 function diguyCharacter::set_desired_action

Prototype:

```
int set_desired_action(const char* action_name,
    float speed = DIGUY_DEFAULT_FLOAT,
    int retain_path_shape = 0);
```

Description:

This function sets the desired action of the character.

This function will put the character into free action mode. If the character was in path action mode, actions will no longer be set by action beads on the path.

If the character is in path position mode and 1 is passed for retain_path_shape, the character will stay in path position mode. If 0 is passed for retain_path_shape, the character will be put into free position mode.

If the character is in free position mode the retain_path_shape argument will have no effect.

If this call is interrupting a character in path action mode and path position mode, the character will not be able to resume the interrupted path unless `retain_path_shape` is set to 1. See [resume_interrupted_path\(\)](#).

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Mode Effects:

- ◇ This function will put the character into free action mode.
- ◇ This function may change the position mode; see Description.

Arguments:

<code>action_name</code>	name of the desired action to be performed by the character
<code>speed</code>	the speed at which the character should travel while performing the action; pass <code>DIGUY_DEFAULT_FLOAT</code> for the optimal speed to be used
<code>retain_path_shape</code>	pass 1 to remain in path position mode; pass 0 to change to free position mode

Returns:

0 on success, -1 on failure

5.12.79 function diguyCharacter::get_desired_action**Prototype:**

```
const char* get_desired_action();
```

Returns:

name of desired action; see [set_desired_action\(\)](#)

Mode Restrictions:

◇ This function can only be called in free action mode.

5.12.80 function diguyCharacter::get_current_action**Prototype:**

```
const char* get_current_action();
```

Returns:

name of current action being performed by the character

5.12.81 function diguyCharacter::force_action

Prototype:

```
int force_action(const char* action_name,
float speed = DIGUY_DEFAULT_FLOAT,
int include_transition_arc = 1,
float max_rampdown_interval = 0.5f,
int retain_path_shape = 0,
float t_offset_into_new_action = 0.0f);
```

Description:

This function forces the current action of the character to be the action identified by `action_name`.

See `set_desired_action()` for a discussion of the effects on position mode.

This function will have no effect on a dead character (see `die_now()` and `get_dead()`).

By passing a value greater than 0 for `t_offset_into_new_action`, the character can be made to begin the action partway in. This is useful when forcing actions for a lot of characters at the same time; without the offset, the characters would move in lockstep.

The most recent forced action, the time at which it was forced, and any time offset can be queried using the functions `get_most_recent_forced_action()`, `get_most_recent_forced_action_t()`, and `get_most_recent_forced_action_t_offset()`.

Mode Effects:

- ◇ This function will put the character into free action mode.
- ◇ This function may change the position mode; see Description.

Arguments:

<code>action_name</code>	name of the action to be performed by the character
<code>speed</code>	the speed at which the character should travel while performing the action; pass <code>DIGUY_DEFAULT_FLOAT</code> for the optimal speed to be used
<code>include_transition_arc</code>	flag for whether the transition motion from the current action to the new action is included; pass 1 for best chance of good looking transition; pass 0 for fastest

	response
<code>max_rampdown_interval</code>	max amount of time spent trying to smooth over any motion seams; set to 0.5 for a smooth transition, set to 0 for a potentially rough transition
<code>retain_path_shape</code>	pass 1 to remain in path position mode; pass 0 to change to free position mode
<code>t_offset_into_new_action</code>	how much to shift time forward into new action; must be ≥ 0 ; see comment above for more info

Returns:

0 on success, -1 on failure

5.12.82 function `diguyCharacter::force_action_with_duration`

Prototype:

```
int force_action_with_duration(const char* action_name,
    float duration,
    int include_transition_arc = 1,
    float max_rampdown_interval = 0.5f);
```

Description:

This function is similar to `force_action()`, but allows the specification of how long the interruption should last.

This function will have no effect on a dead character (see `die_now()` and `get_dead()`).

Mode Restrictions:

◇ This function can only be called if the character is in path action mode **and** path position mode.

Mode Effects:

- ◇ This function will put the character into free action mode.
- ◇ This function will not change the position mode.

Arguments:

duration how long the action should be performed before an automatic `resume_interrupted_path()` call is made

Returns:

0 on success, -1 on failure

5.12.83 function `diguyCharacter::force_action_and_path_shape`

Prototype:

```
int force_action_and_path_shape(const char* action_name,
                               const char* path_shape_name,
                               const char* waypoint_name = NULL,
                               float distance_into_path = 0.0f);
```

Description:

This function forces the current action of the character to be the action identified by `action_name`, to be executed along the path shape specified by `path_shape_name`.

This function will have no effect on a dead character (see `die_now( )` and `get_dead( )`).

Mode Effects:

- ◇ This function will put the character into free action mode.
- ◇ This function will put the character into path position mode.

Arguments:

<code>action_name</code>	name of the action to be performed by the character
<code>path_shape_name</code>	name of the path shape on which action should be performed
<code>waypoint_name</code>	name of waypoint on path shape to begin at; default of NULL will start at first waypoint
<code>distance_into_path</code>	distance into path shape to start at, if <code>waypoint_name</code> isn't specified

Returns:

0 on success, -1 on failure

5.12.84 function `diguyCharacter::get_most_recent_forced_action`

Prototype:

```
const char* get_most_recent_forced_action();
```

Description:

This function returns the most recently forced action as set by by `force_action()`, `force_action_with_duration()`, or `force_action_and_path_shape()`.

Returns:

the name of the most recent forced action, or NULL if no action has been forced

5.12.85 function**diguyCharacter::get_most_recent_forced_action_t****Prototype:**

```
float get_most_recent_forced_action_t();
```

Description:

This function returns the time at which the most recently forced action occurred.

Returns:

time at which force action occurred; -1.0 if no action has been forced

5.12.86 function**diguyCharacter::get_most_recent_forced_action_t_offset****Prototype:**

```
float get_most_recent_forced_action_t_offset();
```

Description:

This function returns the action time offset of the most recently forced action. This is typically value of the `t_offset_into_new_action` argument of the `force_action()` call.

Returns:

time at which force action occurred; -1.0 if no action has been forced

5.12.87 function diguyCharacter::add_pending_desired_action**Prototype:**

```
int add_pending_desired_action(const char* action,
    float scenario_t,
    int    remove_existing_pending_actions = 0,
    float speed = DIGUY_DEFAULT_FLOAT,
    int    retain_path_shape = 1);
```

Description:

This function adds a "pending action" to the character. This is essentially a delayed `set_desired_action()` call. The desired action will be set at the passed scenario t, just as if `set_desired_action()` is called at that time.

Note: Any call to `set_desired_action()` or `force_action()`, or their related variants, will clear any pending actions.

Arguments:

action	name of pending desired action
scenario_t	time at which to set desired action
remove_existing_pending_actions	pass 1 to remove any previously added pending actions, 0 to leave them
speed	as passed to <code>set_desired_action()</code>
retain_path_shape	as passed to <code>set_desired_action()</code>

Returns:

0 on success, -1 on failure

5.12.88 function diguyCharacter::add_pending_force_action**Prototype:**

```
int add_pending_force_action(const char* action,
    float scenario_t,
    int    remove_existing_pending_actions = 0,
    float speed = DIGUY_DEFAULT_FLOAT,
    int    include_transition_arc = 1,
    float max_rampdown_interval = 0.5f,
    int    retain_path_shape = 1,
    float first_arc_time_shift = 0.0f);
```

Description:

Similar to [add_pending_desired_action\(\)](#), but will do a [force_action\(\)](#) at the passed scenario_t instead of a [set_desired_action\(\)](#).

Note: Any call to [set_desired_action\(\)](#) or [force_action\(\)](#), or their related variants, will clear any pending actions.

Returns:

0 on success, -1 on failure

5.12.89 function diguyCharacter::is_valid_action**Prototype:**

```
int is_valid_action(const char* action);
```

Returns:

1 if the passed action name is an action available to this character, 0 if not

5.12.90 function diguyCharacter::die_now**Prototype:**

```
int die_now(const char* preferred_dead_action_name = "(default)");
```

Description:

This function sends this character a signal to die as soon as possible. The character will transition to a dead action.

The following side-effects will also happen:

- all aiming is ended
- all gazing is ended
- all pointing is ended
- head nodding and shaking are stopped
- all gestures are aborted
- all sounds originated by the character are stopped

Many function calls will have no effect on dead characters; see individual function descriptions for limitations.

Use the [get_dead\(\)](#) call to retrieve the dead state of a character.

Mode Effects:

- ◇ This function will put the character into free action mode.
- ◇ This function will not change the position mode.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.91 function diguyCharacter::revive_now**Prototype:**

```
int revive_now(const char* preferred_revive_action_name = "(default)");
```

Description:

This function revives a dead character. The limitations of dead characters will be lifted from the revived character; e.g., the character will be able to gaze, execute gestures, etc.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.92 function diguyCharacter::get_dead**Prototype:**

```
int get_dead();
```

Description:

This function returns whether or not the character is dead. Characters can be killed by [die_now\(\)](#) function calls, as well as by being hit by weapon fire.

Returns:

1 if character is dead ([die_now\(\)](#) function has been called), 0 if not.

5.12.93 function diguyCharacter::set_speed

Prototype:

```
int set_speed(float speed);
```

Description:

Sets the speed the character should attempt to move, in meters per second.

This setting takes effect immediately. To set a desired action and a desired speed at the same time, use the speed argument of the [set_desired_action\(\)](#) call.

This desired speed setting can be undone in a couple of ways:

1. by calling [set_speed\(\)](#) with an argument of [DIGUY_DEFAULT_FLOAT](#), or
2. by calling [set_desired_action\(\)](#), or
3. by calling one of the [force_action\(\)](#) functions

Arguments:

speed	desired speed in meters per second

Returns:

0 on success, -1 on failure

5.12.94 function diguyCharacter::get_speed

Prototype:

```
float get_speed();
```

Returns:

the approximate speed at which the character is moving, in meters per second

Position and Orientation Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.12.95 function diguyCharacter::set_position

Prototype:

```
int set_position(float tx, float ty, float tz);
```

Description:

Sets the position of the character relative to the origin of the *Boston Dynamics* global coordinate system.

The new settings will take effect immediately, possibly causing the character to "teleport" if the new values are significantly different than the old.

The *Boston Dynamics* global coordinate system is right-handed, with X forward, Z up, and Y to the left. Rotation directions follow standard right-handed coordinate system conventions:

- ◇ positive rotations about X cause a counter-clockwise roll
- ◇ position rotations about Y cause a forward pitch
- ◇ positive rotations about Z cause a yaw to the left

Note that if the character has an altitude function (see [set_altitude_function\(\)](#)), the tz argument will effectively be ignored as the altitude function will override it.

If the magnitude of the numbers is large (say 32000 or higher), the function [set_position_double\(\)](#) should be used instead.

Mode Effects:

- ◇ This function will put the character into free position mode.
- ◇ This function will put the character into free action mode.

Arguments:

tx, ty, position in meters from the origin
tz

Returns:

0 on success, -1 on failure

5.12.96 function diguyCharacter::set_position_double**Prototype:**

```
int set_position_double(double tx, double ty, double tz);
```

Description:

Similar to `set_position()`, but using double-precision rather than single-precision numbers. If the magnitude of the numbers is large (say 32000 or higher), this function should be used.

Mode Effects:

- ◇ This function will put the character into free position mode.
- ◇ This function will put the character into free action mode.

Arguments:

tx, ty, position in meters from the origin
tz

Returns:

0 on success, -1 on failure

5.12.97 function diguyCharacter::get_position**Prototype:**

```
int get_position(float* tx, float* ty, float* tz);
```

Description:

Retrieves the position of the character in the *Boston Dynamics* global coordinate system.

See [set_position\(\)](#) for a description of the coordinate system.

Arguments:

tx, ty, position in meters from the origin
tz

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.98 function diguyCharacter::get_position_double**Prototype:**

```
int get_position_double(double* tx, double* ty, double* tz);
```

Description:

Retrieves the position of the character in the *Boston Dynamics* global coordinate system.

See [set_position\(\)](#) for a description of the coordinate system.

Unlike [get_position\(\)](#), this function returns higher-precision values for the position, which is important when the character is far from the coordinate system origin.

Arguments:

tx, ty, position in meters from the origin
tz

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.99 function diguyCharacter::set_position_relative_to_parent

Prototype:

```
int set_position_relative_to_parent(float tx, float ty, float tz);
```

Description:

Sets the position of the character relative to its parent or the origin of the *Boston Dynamics* global coordinate system if this character is not parented.

See [set_position\(\)](#) for a description of the coordinate system.

The new settings will take effect immediately, possibly causing the character to "teleport" if the new values are significantly different than the old.

Mode Effects:

- ◇ This function will put the character into free position mode.
- ◇ This function will put the character into free action mode.

Arguments:

tx, ty, position in meters from the origin
tz

Returns:

0 on success, -1 on failure

5.12.100 function diguyCharacter::get_position_relative_to_parent

Prototype:

```
int get_position_relative_to_parent(float* tx, float* ty, float* tz);
```

Description:

Retrieves the position of the character relative to its parent, or relative to the origin of the *Boston Dynamics* global coordinate system if this character is not parented.

See [set_position\(\)](#) for a description of the coordinate system.

Arguments:

tx, ty, position in meters from the origin
tz

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.101 function**diguyCharacter::set_position_to_other_character_position****Prototype:**

```
int set_position_to_other_character_position(const char* other_character_name,
    const char* other_link_name = NULL,
    float offset_x = 0.0f, float offset_y = 0.0f, float offset_z = 0.0f,
    int also_set_orientation = 1);
```

Description:

Similar to [set_position_relative_to_parent\(\)](#), but the position is either from the other character's overall position (if other_link_name is NULL), or the position of the other character's link (if other_link_name specifies a link on the other character).

The position that is set is relative to this character's parent.

Mode Effects:

- ◊ This function will put the character into free position mode.
- ◊ This function will put the character into free action mode.

Arguments:

<code>other_character_name</code>	other character from which to read position
<code>other_link_name</code>	optional link on other character; if NULL, other character's base position is used
<code>offset_x, offset_y, offset_z</code>	offset to apply to position read from other character
<code>also_set_orientation</code>	pass 1 to also set this character's orientation from the other character's orientation

Returns:

0 on success, -1 on failure

5.12.102 function `diguyCharacter::set_desired_position`

Prototype:

```
int set_desired_position(float tx, float ty, float tz,
    int force_guide_unacquired = 0);
```

Description:

Sets the desired position of the character. How the character moves toward the desired position is determined by its current guides; see [add_guide\(\)](#) and [create_guide\(\)](#).

Mode Effects:

- ◇ This function will put the character into free position mode.
- ◇ This function will put the character into free action mode.

Arguments:

<code>tx, ty, tz</code>	desired position in meters from the origin
<code>force_guide_unacquired</code>	

this optional argument will set any guides
the character has to be unacquired

Returns:

0 on success, -1 on failure

5.12.103 function `diguyCharacter::set_desired_position_to_current_position`

Prototype:

```
int set_desired_position_to_current_position(int also_set_orientation = 1);
```

Description:

Like `set_desired_position()`, but uses the character's current position as the desired position.

Mode Effects:

- ◇ This function will put the character into free position mode.
- ◇ This function will put the character into free action mode.

Arguments:

`also_set_orientation` pass 1 to also set this character's orientation to its current orientation

Returns:

0 on success, -1 on failure

5.12.104 function diguyCharacter::set_desired_position_to_waypoint

Prototype:

```
int set_desired_position_to_waypoint(diguyWaypoint* waypoint,
    float offset_x = 0.0f, float offset_y = 0.0f, float offset_z = 0.0f,
    int offset_in_world_coords = 0);
```

Description:

Like [set_desired_position\(\)](#), but sets the character's desired position to passed waypoint's position.

The passed waypoint can be one created by [diguyScenario::create_waypoint](#), a waypoint from a character's [diguyCharacterPath](#), or waypoint from a [diguyPathShape](#).

Mode Effects:

- ◊ This function will put the character into free position mode.
- ◊ This function will put the character into free action mode.

Arguments:

waypoint	pointer to diguyWaypoint object
offset_x, offset_y, offset_z	offset to apply to waypoint position
offset_in_world_coords	1 if offset is to be in world coordinates, 0 if it is to be in waypoint-local coordinates

Returns:

0 on success, -1 on failure

5.12.105 function `diguyCharacter::set_desired_position_to_path_waypoint`

Prototype:

```
int set_desired_position_to_path_waypoint(const char* path_name,
    const char* waypoint_name);
```

Description:

Like [set_desired_position\(\)](#), but finds the specified waypoint on the specified path and sets the character's desired position to the waypoint's position.

Mode Effects:

- ◊ This function will put the character into free position mode.
- ◊ This function will put the character into free action mode.

Arguments:

path_name	name of path on which to find waypoint

`waypoint_name` name of waypoint

Returns:

0 on success, -1 on failure

5.12.106 function

`diguyCharacter::set_desired_position_to_path_shape_waypoint`

Prototype:

```
int set_desired_position_to_path_shape_waypoint(const char* path_shape_name,
const char* waypoint_name);
```

Description:

Like [set_desired_position\(\)](#), but finds the specified waypoint on the specified path shape and sets the character's desired position to the waypoint's position.

Mode Effects:

- ◇ This function will put the character into free position mode.
- ◇ This function will put the character into free action mode.

Arguments:

`path_shape_name` name of path shape on which to find waypoint

waypoint_name name of waypoint

Returns:

0 on success, -1 on failure

5.12.107 function diguyCharacter::get_desired_position

Prototype:

```
int get_desired_position(float* tx, float* ty, float* tz);
```

Description:

Retrieves the desired position of the character.

Mode Restrictions:

◊ This function can only be called in free position mode.

Arguments:

tx, ty, desired position in meters from the origin
tz

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.108 function diguyCharacter::get_delta_to_desired_position

Prototype:

```
int get_delta_to_desired_position(float* tx, float* ty, float* tz);
```

Description:

Retrieves the delta vector from current position to desired one.

Mode Restrictions:

◇ This function can only be called in free position mode.

Arguments:

tx, ty, tz delta in meters

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.109 function diguyCharacter::set_initial_position

Prototype:

```
int set_initial_position(float tx, float ty, float tz);
```

Description:

Sets the initial position the character should move to on a call to [diguyScenario::reset\(\)](#).

If the character has an initial path, that path, not this function call, will determine its initial position.

Arguments:

tx, ty, tz initial position in meters from the origin

Returns:

0 on success, -1 on failure

5.12.110 function diguyCharacter::get_initial_position**Prototype:**

```
int get_initial_position(float* tx, float* ty, float* tz);
```

Description:

Retrieves the initial position of the character.

If the character is in free position mode or has no initial path, this will be the position set by [set_initial_position\(\)](#).

Otherwise this will be the position of the first waypoint of the initial path.

Arguments:

tx, ty, tz initial position in meters from the origin

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.111 function diguyCharacter::set_orientation**Prototype:**

```
int set_orientation(float rz, float rx, float ry);
```

Description:

Sets the orientation of the character relative to the origin of the *Boston Dynamics* global coordinate system.

See [set_position\(\)](#) for a description of the coordinate system.

The new settings will take effect immediately, possibly causing the character to "teleport" if the new values are significantly different than the old.

Note that if the character has an up vector of 'z' or 'n' or even sometimes 'd' (as set by the [set_up_vector\(\)](#) call) the rx and ry values will not have any effect. To be able to set rx and ry values, set the up vector of the character to 'c', for "custom".

Mode Effects:

- ◇ This function will put the character into free position mode.
- ◇ This function will put the character into free action mode.

Arguments:

rz, rx, ry orientations in degrees

Returns:

0 on success, -1 on failure

5.12.112 function diguyCharacter::get_orientation**Prototype:**

```
int get_orientation(float* rz, float* rx, float* ry);
```

Description:

Retrieves the orientation of the character in the *Boston Dynamics* global coordinate system. See [set_position\(\)](#) for a description of the coordinate system.

Note:

For vehicles, you may want to use the function [get_vehicle_body_orientation\(\)](#). See that function for more information.

Arguments:

rz, rx,	orientations in degrees counter-clockwise from the positive
ry	X axis

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.113 function diguyCharacter::set_orientation_relative_to_parent

Prototype:

```
int set_orientation_relative_to_parent(float rz, float rx, float ry);
```

Description:

Sets the orientation of the character relative to its parent or the origin of the *Boston Dynamics* global coordinate system if this character is not parented.

See [set_position\(\)](#) for a description of the coordinate system.

The new settings will take effect immediately, possibly causing the character to "teleport" if the new values are significantly different than the old.

Mode Effects:

- ◇ This function will put the character into free position mode.
- ◇ This function will put the character into free action mode.

Arguments:

rz, rx, ry orientations in degrees

Returns:

0 on success, -1 on failure

5.12.114 function diguyCharacter::get_orientation_relative_to_parent

Prototype:

```
int get_orientation_relative_to_parent(float* rz, float* rx, float* ry);
```

Description:

Retrieves the orientation of the character relative to its parent, or the origin of the *Boston Dynamics* global coordinate system if this character does not have a parent.

See [set_position\(\)](#) for a description of the coordinate system.

Arguments:

rz, rx,	orientations in degrees counter-clockwise from the positive
ry	X axis

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.115 function diguyCharacter::set_initial_up_vector**Prototype:**

```
int set_initial_up_vector(char axis);
```

Description:

This function sets how the character should be oriented relative to its surroundings.

This function changes the current up vector, and the up vector that will be set when the character is reset. To change only the current up vector (a temporary change that goes away on reset), call [set_up_vector\(\)](#).

Arguments:

axis the up vector the character should use; legal values shown below:

- ◇ 'd' – use default settings; character will change up vector depending on current action and path shape parameters
- ◇ 'z' – Z axis is up; character will be oriented so that positive Z in character coordinates is always oriented along the positive world Z axis
- ◇ 'n' – normal to path; character will be oriented so that positive Z in character coordinates is normal to the path shape the character is on
- ◇ 'g' – ground-clamped; the vehicle will attempt to cling to the terrain, in an approximation of real rigid body physics
- ◇ 'c' – custom; character's rx and ry orientation values will not be modified

Having an up vector of 'z' effectively zeroes out the rx and ry values of the character's orientation.

Having an up vector of 'n' may override rx and ry values of the character's orientation with values derived from the character's path shape.

Having an up vector of 'c' allows the pitch and roll of the character to be changed without taking it off the path it is on by calling [set_custom_orientation_rx_and_ry\(\)](#). Calling [set_orientation\(\)](#) has the side-effect of putting the character into free position mode, taking it off of any path it's on.

Returns:

0 on success, -1 on failure

5.12.116 function diguyCharacter::get_initial_up_vector**Prototype:**

```
char get_initial_up_vector();
```

Returns:

current up vector; see [set_initial_up_vector\(\)](#)

5.12.117 function diguyCharacter::set_up_vector**Prototype:**

```
int set_up_vector(char axis);
```

Description:

This function sets how the character should be oriented relative to its surroundings.

This function changes only the current up vector. To change the up vector that will be used when the character is reset, call [set_initial_up_vector\(\)](#).

Arguments:

<code>axis</code>	the up vector the character should use; see set_initial_up_vector() for description

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.118 function diguyCharacter::get_up_vector

Prototype:

```
char get_up_vector();
```

Returns:

current up vector; see [set_up_vector\(\)](#)

5.12.119 function**diguyCharacter::set_custom_orientation_rx_and_ry****Prototype:**

```
int set_custom_orientation_rx_and_ry(float rx,
float ry);
```

Description:

This function sets the rx and ry components of the character's orientation. Unlike [set_orientation\(\)](#), this function does not put the character into free position mode if it is currently in path position mode.

Note that if the character has an up vector of 'z' or 'n' or even sometimes 'd' (as set by the [set_up_vector\(\)](#) call) this function will not have any effect. To be able to set rx and ry values, set the up vector of the character to 'c', for "custom".

The passed rx and ry values are in character-local coordinates, not parent coordinates if the character has a parent.

Mode Effects:

◇ None.

Arguments:

rx, ry orientations in degrees

Returns:

0 on success, -1 on failure

5.12.120 function diguyCharacter::set_desired_orientation**Prototype:**

```
int set_desired_orientation(float rz, float rx, float ry,
                           int force_guide_unacquired = 0);
```

Description:

Sets the desired orientation of the character. See [set_desired_position\(\)](#).

Mode Effects:

- ◇ This function will put the character into free position mode.
- ◇ This function will put the character into free action mode.

Arguments:

rz, rx,	orientations in degrees counter-clockwise from the positive X axis
ry	

Returns:

0 on success, -1 on failure

5.12.121 function diguyCharacter::get_desired_orientation**Prototype:**

```
int get_desired_orientation(float* rz, float* rx, float* ry);
```

Description:

Retrieves the desired orientation of the character.

Mode Restrictions:

- ◇ This function can only be called in free position mode.

Arguments:

rz, rx,	orientations in degrees counter-clockwise from the positive X axis
ry	

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.122 function diguyCharacter::get_delta_to_desired_orientation

Prototype:

```
int get_delta_to_desired_orientation(float* rz, float* rx, float* ry);
```

Description:

Retrieves the delta from current orientation of the character to the desired one.

Mode Restrictions:

◊ This function can only be called in free position mode.

Arguments:

<code>rz, rx,</code>	orientation deltas in degrees counter-clockwise from the
<code>ry</code>	positive X axis

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.123 function diguyCharacter::set_desired_orientation_to_current_orientation

Prototype:

```
int set_desired_orientation_to_current_orientation();
```

Description:

Like [set_desired_orientation\(\)](#), but uses the character's current orientation as the desired orientation.

Mode Effects:

- ◇ This function will put the character into free position mode.
- ◇ This function will put the character into free action mode.

Returns:

0 on success, -1 on failure

5.12.124 function**diguyCharacter::set_desired_orientation_towards_position****Prototype:**

```
int set_desired_orientation_towards_position(float x, float y, float z,
int force_guide_unacquired = 0);
```

Description:

Like `set_desired_orientation()`, but orients the character towards the position specified.

Mode Effects:

- ◇ This function will put the character into free position mode.
- ◇ This function will put the character into free action mode.

Returns:

0 on success, -1 on failure

5.12.125 function diguyCharacter::set_initial_orientation**Prototype:**

```
int set_initial_orientation(float rz, float rx, float ry);
```

Description:

Sets the initial orientation the character should move to on a call to `diguyScenario::reset()`.

If the character is on a path the path, not this function call, will determine its initial orientation.

Arguments:

<code>rz, rx, ry</code>	orientations in degrees counter-clockwise from the positive X axis

Returns:

0 on success, -1 on failure

5.12.126 function diguyCharacter::get_initial_orientation**Prototype:**

```
int get_initial_orientation(float* rz, float* rx, float* ry);
```

Description:

Retrieves the initial orientation of the character.

If the character is in free position mode or has no initial path, this will be the orientation set by [set_initial_orientation\(\)](#).

Otherwise this will be the orientation of the first waypoint of the initial path.

Arguments:

<code>rz, rx,</code>	orientations in degrees counter-clockwise from the positive
<code>ry</code>	X axis

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.127 function diguyCharacter::get_link_position**Prototype:**

```
int get_link_position(const char* link_name,
    float* tx, float* ty, float* tz,
    float* rz, float* rx, float* ry);
```

Description:

Returns the location and orientation of a specific link of the character.

Arguments:

link_name	name of the link; links are named after their inbound joint name
tx, ty, tz	position in meters from the origin
rz, rx, ry	orientations in degrees counter-clockwise from the positive X axis

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.128 function diguyCharacter::get_link_position_double**Prototype:**

```
int get_link_position_double(const char* link_name,
    double* tx, double* ty, double* tz,
    double* rz, double* rx, double* ry);
```

Description:

Returns the location and orientation of a specific link of the character.

Arguments:

link_name	name of the link; links are named after their inbound joint name
tx, ty, tz	position in meters from the origin
rz, rx, ry	orientations in degrees counter-clockwise from the positive X axis

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.129 function diguyCharacter::get_link_position_with_offset**Prototype:**

```
int get_link_position_with_offset(const char* link_name,
    float offset_tx, float offset_ty, float offset_tz,
    float* tx, float* ty, float* tz,
    float* rz, float* rx, float* ry);
```

Description:

Similar to [get_link_position\(\)](#), but allows specification of offset into ending link.

Arguments:

link_name	name of the link
offset_tx, offset_ty, offset_tz	offset in link
tx, ty, tz	position in meters from the origin
rz, rx, ry	orientations in degrees counter-clockwise from the positive X axis

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.130 function**diguyCharacter::get_link_position_with_offset_double****Prototype:**

```
int get_link_position_with_offset_double(const char* link_name,
    double offset_tx, double offset_ty, double offset_tz,
    double* tx, double* ty, double* tz,
    double* rz, double* rx, double* ry);
```

Description:

Similar to [get_link_position\(\)](#), but allows specification of offset into ending link.

Arguments:

link_name	name of the link
offset_tx, offset_ty, offset_tz	offset in link
tx, ty, tz	position in meters from the origin
rz, rx, ry	orientations in degrees counter-clockwise from the positive X axis

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.131 function diguyCharacter::get_link_relative_position**Prototype:**

```
int get_link_relative_position(const char* beginning_link_name,
                             const char* ending_link_name,
                             float* tx, float* ty, float* tz,
                             float* rz, float* rx, float* ry);
```

Description:

Returns the location and orientation of a specific link of the character relative to the position of another link.

Arguments:

<code>beginning_link_name</code>	name of the first link; send NULL to to specify the position link
<code>ending_link_name</code>	name of the last link; send NULL to to specify the position link
<code>tx, ty, tz</code>	position in meters from the origin
<code>rz, rx, ry</code>	orientations in degrees counter-clockwise from the positive X axis

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.132 function**diguyCharacter::get_link_relative_position_with_offset****Prototype:**

```
int get_link_relative_position_with_offset(const char* beginning_link_name,
    const char* ending_link_name,
    float offset_tx, float offset_ty, float offset_tz,
    float* tx, float* ty, float* tz,
    float* rz, float* rx, float* ry);
```

Description:

Similar to [get_link_relative_position\(\)](#), but allows specification of offset into ending link.

Arguments:

beginning_link_name	name of the first link; send NULL to to specify the position link
ending_link_name	name of the last link; send NULL to to specify the position link
offset_tx, offset_ty, offset_tz	offset in ending link
tx, ty, tz	position in meters from the origin
rz, rx, ry	orientations in degrees counter-clockwise from the positive X axis

Pass NULL for any values that are not needed.

Returns:

0 on success, -1 on failure

5.12.133 function**diguyCharacter::set_apply_actor_scale_to_action_bead_travel****Prototype:**

```
void set_apply_actor_scale_to_action_bead_travel(int apply_scale_flag);
```

Description:

By default characters on a path will scale their movement speed and velocity based on the scale of the actor the character is based on. For example, if a tall character and short character are both playing the exact same walk motion, the tall character will move further per motion repetition than the short character.

Scaling the travel in this way leads to more realistic looking motion, but can cause different characters using the same actions and underlying motions to move at different speeds. At times it is convenient to not scale the travel.

If actor scaling is disabled, characters of different sizes will all move the same distance, and at the same speed, if they have the same path shapes and action beads. Note that in cases of extreme differences between the actor the motion is based on and the actor the character's current appearance is based on, the characters feet may slip on the ground more than normal.

Arguments:

`apply_scale_flag` pass 1 to apply actor scale to travel; 0 to not

5.12.134 function**diguyCharacter::get_apply_actor_scale_to_action_bead_travel****Prototype:**

```
int get_apply_actor_scale_to_action_bead_travel();
```

Returns:

1 if actor scale is being applied to travel, 0 if not

5.12.135 function diguyCharacter::local_to_global**Prototype:**

```
int local_to_global(float x, float y, float z,  
float* res_x, float* res_y, float* res_z);
```

Description:

This function transforms a point that is in the local space of the character into a location in the world. Useful for things like navigating AIs to a location near a prop. Or moving to a spot in front of someone else.

This code presumes up_axis is 'z' and that there is only rotation about that axis.

5.12.136 function diguyCharacter::global_to_local**Prototype:**

```
int global_to_local(float x, float y, float z,
    float* res_x, float* res_y, float* res_z);
```

Description:

This function transforms a point that is in world space into the local space of a character. Useful for figuring out if something was seen.

This code presumes up_axis is 'z' and that there is only rotation about that axis.

5.12.137 function diguyCharacter::get_velocity**Prototype:**

```
void get_velocity(float* vel_x, float* vel_y, float* vel_z);
```

Description:

Retrieves the current estimated velocity of the character. Note this value may not be valid in environments where characters are transformed independent of DI-Guy function calls. Teleporting characters can also temporarily make this calculation incorrect.

Arguments:

```
vel_x,      meters/sec
vel_y,
vel_z
```

Pass NULL for any values that are not needed.

5.12.138 function diguyCharacter::set_desired_velocity**Prototype:**

```
void set_desired_velocity(float vel_x, float vel_y, float vel_z);
```

Description:

Currently only used by the Adaptive guide

Arguments:

```
vel_x,      meters/sec
vel_y,
vel_z
```

5.12.139 function diguyCharacter::get_angular_velocity**Prototype:**

```
void get_angular_velocity(float* vel_rz);
```

Description:

Retrieves the current estimated angular velocity of the character. As above, changes made to characters independent of DI-Guy function calls will temporarily make this value incorrect.

Currently only the angular velocity in yaw is measured.

Arguments:

```
vel_rz      degrees/sec
```

Pass NULL for any values that are not needed.

Appearance Functions

5.12.140 function diguyCharacter::get_appearance

Prototype:

```
const char* get_appearance();
```

Description:

Returns the base appearance of the character, as passed to the function [diguyScenario::create_character\(\)](#). This pointer will never be NULL.

Note that the current appearance (as returned by [get_current_appearance\(\)](#)) may be different than the base appearance. This base appearance is the starting appearance of the character before any calls to [set_current_appearance\(\)](#) have been made.

Returns:

name of base appearance of the character

5.12.141 function diguyCharacter::set_appearance**Prototype:**

```
int set_appearance(const char* appearance);
```

Description:

This function sets the base appearance of the character. The base appearance is the appearance the character will take if the scenario is reset, undoing any appearance changes made by calls to [set_current_appearance\(\)](#).

Calling this function also makes an implicit call to [set_current_appearance\(\)](#).

Arguments:

`appearance` name of new base appearance

Returns:

0 on success, -1 on failure

5.12.142 function diguyCharacter::set_current_appearance**Prototype:**

```
int set_current_appearance(const char* appearance,
                           int allow_actor_change = 1);
```

Description:

This function sets the current appearance of this character.

Note that different appearances may be based on different actors. Different actors represent people of different sizes, so changing to an appearance that has a different actor can cause a pronounced change in character size.

Also, changing actors can take some time, as more internal character state needs to be updated to support the new size of the character.

Arguments:

`appearance` name of new desired appearance
`allow_actor_change`

pass 1 to allow the appearance change even if
the actor changes, 0 to not

See Also:

[get_current_appearance\(\)](#),
[set_current_head_appearance\(\)](#)

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.143 function diguyCharacter::get_current_appearance

Prototype:

```
const char* get_current_appearance();
```

Returns:

current appearance; see [set_current_appearance\(\)](#)

5.12.144 function diguyCharacter::get_current_appearance_is**Prototype:**

```
int get_current_appearance_is(const char* appearance);
```

Description:

This function can be used to see if the current appearance has the passed name. This is primarily useful for decisions and other limited logic applications.

Returns:

1 if current appearance has passed name; 0 if not

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.145 function diguyCharacter::set_current_head_appearance**Prototype:**

```
int set_current_head_appearance(const char* head_appearance);
```

Description:

This function sets the current head appearance of this character.

Arguments:

`head_appearance` name of the new desired head appearance

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

◇ Decision Bead

5.12.146 function diguyCharacter::get_current_head_appearance

Prototype:

```
const char* get_current_head_appearance();
```

Returns:

current head appearance; see [set_current_head_appearance\(\)](#)

5.12.147 function diguyCharacter::get_current_head_appearance_is

Prototype:

```
int get_current_head_appearance_is(const char* head_appearance);
```

Description:

Similar to [get_current_appearance_is\(\)](#), but for head appearance.

Returns:

1 if current head appearance has passed name; 0 if not

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.148 function diguyCharacter::get_num_appearances_of_type

Prototype:

```
int get_num_appearances_of_type(diguyCharacterAppearanceTypes appearance_type);
```

Description:

A appearance can be a body appearance, head appearance, an expressive head appearance, or a hand item appearance. NOTE: available heads, expressive heads, hands all can change based on the current appearance.

Arguments:

appearance_type a value indicating the type of supplementary appearance of concern

Returns:

The number of available supplementary appearances of given type

Callable From:

- ◇ C++
- ◇ Script

5.12.149 function diguyCharacter::get_appearance_name_at_index

Prototype:

```
const char* get_appearance_name_at_index(diguyCharacterAppearanceTypes appearance_ty
```

Description:

See description for `get_num_supplementary_appearances_of_type( )`

Arguments:

<code>appearance_type</code>	a value indicating the type of appearance of concern
<code>index</code>	a value indicating the type of supplementary appearance of concern

Returns:

The name of the appearance or NULL

Callable From:

- ◇ C++
- ◇ Script

Performance Functions

The functions in this section give control over various settings that can have a big impact on the performance of the character. In this case, better performance means less time to update and/or draw the character each frame.

Better performance almost always comes at the expense of reduced functionality and/or degraded visual appearance.

Many of the parameters below can be automatically managed by a [diguyLoadManager](#), which is returned by `diguyApp::create_load_manager( )`.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.150 function diguyCharacter::set_lod_ranges**Prototype:**

```
int set_lod_ranges(float* lod_ranges);
```

Description:

Sets the graphics level of detail (LOD) ranges for this character. Compare with [diguyScenario::set_default_lod_ranges\(\)](#), which sets the LOD ranges for newly created characters of a particular type. In both cases, ranges must be a static array of 8 floats, each larger than the one before.

This function sets graphics LOD ranges only. Motion levels of detail can be set by calling [set_motion_lod\(\)](#).

Arguments:

`lod_ranges` new lod ranges for this character;

See [diguyScenario::set_default_lod_ranges\(\)](#) for a description of what to pass in `lod_ranges`.

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.12.151 function diguyCharacter::set_graphics_lod**Prototype:**

```
int set_graphics_lod(int lod);
```

Description:

Set the desired graphics level of detail (LOD) for this character. This may be overridden if this character uses automatic LOD switching. See [diguyScenario::set_automatic_graphics_lod_switching\(\)](#).

DI-Guy graphics LODs are numbered 1 (most detail) through 7 (least detail).

Arguments:

lod	an integer between 1 and 7

Returns:

0 on success, -1 on failure

5.12.152 function diguyCharacter::get_graphics_lod**Prototype:**

```
int get_graphics_lod();
```

Returns:

current graphics LOD

5.12.153 function diguyCharacter::set_shader_lod**Prototype:**

```
int set_shader_lod(int lod);
```

Description:

Set the desired shader level of detail (LOD) for this character. By default this is -1 and the shader is automatically picked by distance.

DI-Guy shaders LODs are numbered 1 (most quality) through the number of shaders in the current technique.

Arguments:

lod	an integer between 1 and the number of shaders in the current technique

Returns:

0 on success, -1 on failure

5.12.154 function diguyCharacter::get_shader_lod**Prototype:**

```
int get_shader_lod();
```

Returns:

current shader LOD

5.12.155 function diguyCharacter::set_motion_lod**Prototype:**

```
int set_motion_lod(int motion_lod);
```

Description:

Note: As of DI-Guy 12.5 we strongly recommend using the

multi-threaded pipeline for increasing performance.

Sets the motion level of detail (LOD) for this character. LOD 1 animates all joints of a character. Higher LODs animate fewer joints.

For LOD 6, **nothing** in the pose is updated. The character's pose will remain whatever it was the last time pose data was changed.

LOD	Effect
1	animate all joints
2	stop animating wrists and ankles
3	stop animating elbows and knees
4	stop animating everything but pelvis and position
5	stop animating everything but position
6	stop animating everything but position

Note that characters that are load managed ([get_is_load_managed\(\)](#) returns 1) have this parameter managed automatically. The corresponding function for load managed characters is [diguyLoadManager::set_zone_motion_lod\(\)](#).

Arguments:

`motion_lod` an integer between 1 and 6

Returns:

0 on success, -1 on failure

5.12.156 function diguyCharacter::get_motion_lod**Prototype:**

```
int get_motion_lod();
```

Returns:

character's current motion LOD setting; see [set_motion_lod\(\)](#)

5.12.157 function diguyCharacter::get_current_motion_lod**Prototype:**

```
int get_current_motion_lod();
```

Returns:

character's current motion LOD based on load management and culling; see [set_motion_lod\(\)](#)

5.12.158 function `diguyCharacter::set_minimum_cpl`

Prototype:

```
int set_minimum_cpl(int minimum_cpl);
```

Description:

Note: As of DI-Guy 12.5 we strongly recommend using the

multi-threaded pipeline for increasing performance.

This function sets the minimum CPL (character performance level) of the character.

A higher CPL will limit the types of operations that the character can perform, but will allow for more optimized `update()` calls.

The following CPL limitations are cumulative:

- ◇ CPL 1: all character function calls are available
- ◇ CPL 2: the character cannot save history
- ◇ CPL 3: no advanced pose operations that alter the basic pose of the character, including:
 - gazing
 - pointing
 - aiming
 - gestures
 - head nodding and shaking
 - pose overrides
 - features relying on link matrices
 - wheel rolling and turning on vehicles

Since link matrices are not computed, the DI-Guy Graphics API function

`diguyGraphicsLink::get_transformation_matrix()` will not return valid results.

- ◇ CPL 4: no blends between motions
- ◇ CPL 5: (not currently implemented)

Note that characters that are load managed (`get_is_load_managed()` returns 1) have this parameter managed automatically. The corresponding function for load managed characters is `diguyLoadManager::set_zone_minimum_cpl()`.

The passed `minimum_cpl` may not be achievable if the character is already in the process of doing something not allowed by the new CPL. For example, if the character is already gazing, trying to set the minimum CPL to 3 will fail,

as CPL 3 would disable gazing. In this case the minimum CPL will be set to the current highest possible CPL based on what the character is already doing. In this example the minimum CPL would be set to 2, as gazing is allowed at that CPL.

Conversely, if the minimum CPL has been successfully been set to 3 or higher, any gaze calls such as `diguyCharacter::gaze_at_point()` will fail.

Arguments:

`cpl` minimum CPL allowed for this character

Returns:

0 on success, -1 on failure

5.12.159 function `diguyCharacter::get_minimum_cpl`

Prototype:

```
int get_minimum_cpl();
```

Returns:

most recent setting of `set_minimum_cpl()`

5.12.160 function**diguyCharacter::get_maximum_possible_current_cpl****Prototype:**

```
int get_maximum_possible_current_cpl();
```

Description:

This function returns that maximum possible CPL the character can attain based on what it is currently doing. Advanced pose operations like gazing and pose overrides will limit the maximum CPL that can be achieved.

See [diguyCharacter::set_minimum_cpl\(\)](#) for a description of what is possible under the different CPL levels.

Returns:

maximum possible current CPL

5.12.161 function diguyCharacter::maximize_current_cpl**Prototype:**

```
int maximize_current_cpl();
```

Description:

This function puts the character into the maximum possible CPL that can be achieved based on what the character is currently doing.

This current CPL does not prevent the character from starting operations that will potentially lower the CPL. e.g., if the character starts gazing, the CPL will potentially go down. Use [set_minimum_cpl\(\)](#) to prevent that character from starting such operations.

See [diguyCharacter::set_minimum_cpl\(\)](#) for a description of what is possible under the different CPL levels.

Returns:

0 on success, -1 on failure

5.12.162 function**diguyCharacter::set_automatic_cpl_switching_enabled****Prototype:**

```
int set_automatic_cpl_switching_enabled(int enabled);
```

Description:

This function sets a flag that determines whether the character should raise and lower its CPL automatically. If automatic CPL switching is enabled then any time a potential CPL-changing operation is performed, the character will check the maximum possible current CPL (as per a call to [maximize_current_cpl\(\)](#)) and set its current CPL to the maximum possible.

Arguments:

`enabled` pass 1 to enable automatic CPL switching, 0 to disable it

Returns:

0 on success, -1 on failure

5.12.163 function**diguyCharacter::get_automatic_cpl_switching_enabled****Prototype:**

```
int get_automatic_cpl_switching_enabled();
```

Returns:

most recent setting of [get_automatic_cpl_switching_enabled\(\)](#)

5.12.164 function diguyCharacter::set_current_cpl**Prototype:**

```
int set_current_cpl(int current_cpl);
```

Description:

This function sets the current CPL of the character. Trying to set the CPL lower than the minimum (as set by [set_minimum_cpl\(\)](#)) or higher than the maximum (as returned by [get_maximum_possible_current_cpl\(\)](#)) will fail.

Note that characters that have automatic CPL switching enabled ([get_automatic_cpl_switching_enabled\(\)](#) returns 1) will override this value.

See [diguyCharacter::set_minimum_cpl\(\)](#) for a description of what is possible under the different CPL levels.

Arguments:

cpl	new current CPL

Returns:

0 on success, -1 on failure

5.12.165 function diguyCharacter::get_current_cpl**Prototype:**

```
int get_current_cpl();
```

Returns:

the current CPL the character is using

5.12.166 function diguyCharacter::set_position_update_rate**Prototype:**

```
int set_position_update_rate(int update_rate);
```

Description:

This function sets the position update rate of the character. If the character's position is not updated on a particular update tick, its position will not change for that tick. This can result in a modest performance increase.

The default value for new characters is 100.

This value may be overridden by the minimum full update period, as set by [set_minimum_full_update_period\(\)](#).

Note that if the character's pose is updated, (see [diguyCharacter::set_pose_update_rate\(\)](#)) its position will be updated as well.

Note that characters that are load managed ([get_is_load_managed\(\)](#) returns 1) have this parameter managed automatically. The corresponding

function for load managed characters is

`diguyLoadManager::set_zone_position_update_rate()`.

Arguments:

`update_rate` percentage of scenario ticks for which position should update; can be between 1 and 100

Passing 100 means position will update every tick. Passing 50 means position will update every other tick.

Returns:

0 on success, -1 on failure

5.12.167 function `diguyCharacter::get_position_update_rate`

Prototype:

```
int get_position_update_rate();
```

Returns:

character's position update rate as set by
`set_position_update_rate()`

5.12.168 function diguyCharacter::set_pose_update_rate**Prototype:**

```
int set_pose_update_rate(int update_rate);
```

Description:

This function is very similar to [set_position_update_rate\(\)](#), but sets the percentage of ticks for which the character's pose is updated.

The default value for new characters is 100.

This value may be overridden by the minimum full update period, as set by [set_minimum_full_update_period\(\)](#).

Note that if the character's pose is updated, its position will be updated as well.

Note that characters that are load managed ([get_is_load_managed\(\)](#) returns 1) have this parameter managed automatically. The corresponding function for load managed characters is [diguyLoadManager::set_zone_pose_update_rate\(\)](#).

Arguments:

update_rate percentage of scenario ticks for which pose should update; can be between 1 and 100

Passing 100 means pose will update every tick. Passing 50 means pose will update every other tick.

Returns:

0 on success, -1 on failure

5.12.169 function diguyCharacter::get_pose_update_rate**Prototype:**

```
int get_pose_update_rate();
```

Returns:

character's pose update rate as set by [set_pose_update_rate\(\)](#)

5.12.170 function diguyCharacter::set_minimum_full_update_period**Prototype:**

```
int set_minimum_full_update_period(float min_period);
```

Description:

This function sets the minimum period for position or pose updates. If the passed amount of time passes with no position or pose update (see [set_position_update_rate\(\)](#) and [set_pose_update_rate\(\)](#)), the character's position and potentially pose will be updated.

Note that characters that are load managed ([get_is_load_managed\(\)](#) returns 1) have this parameter managed automatically. The corresponding function for load managed characters is [diguyLoadManager::set_minimum_full_update_period\(\)](#).

The default for new characters is 1.0 seconds.

Arguments:

`min_period` in seconds

Returns:

0 on success, -1 on failure

5.12.171 function `diguyCharacter::get_minimum_full_update_period`

Prototype:

```
float get_minimum_full_update_period();
```

Returns:

character's minimum full update period, as set by
`set_minimum_full_update_period()`

5.12.172 function

`diguyCharacter::set_position_accumulation_disabled`

Prototype:

```
int set_position_accumulation_disabled(int disabled_flag);
```

Description:

This function disables position accumulation due to played actions. The character will stay in whatever position has most recently been set by `set_position()`, regardless of action. Because of this the character will, for example, walk in place. It is up to the user application to correctly set and update the character's position.

This function can be used when a host application provides absolute control over character positions.

Disabling position accumulation will result in a modest performance increase.

Arguments:

`disabled_flag` pass 1 to disable position accumulation, 0 to enable it

Returns:

0 on success, -1 on failure

5.12.173 function diguyCharacter::get_position_accumulation_disabled

Prototype:

```
int get_position_accumulation_disabled();
```

Returns:

most recent setting of `set_position_accumulation_disabled()`

5.12.174 function diguyCharacter::set_all_blends_disabled

Prototype:

```
int set_all_blends_disabled(int disabled_flag);
```

Description:

This function disables all blends between motions of the character. This will speed up performance at the expense of visual quality of the motions. If the character is far from the current camera position, however, the lack of blends may be visually acceptable.

Note that characters that are load managed (`get_is_load_managed()` returns 1) have this parameter managed automatically.

Arguments:

`disabled_flag` pass 1 to disable all blends, 0 to enable them

Returns:

0 on success, -1 on failure

5.12.175 function diguyCharacter::get_all_blends_disabled

Prototype:

```
int get_all_blends_disabled();
```

Returns:

most recent setting of `set_all_blends_disabled()`

5.12.176 function diguyCharacter::set_shape_switches_disabled**Prototype:**

```
int set_shape_switches_disabled(int disabled_flag);
```

Description:

Note: This function is not recommend. There is minimal perf gain.

This function disables all update calls for shape switches.

Switches currently control:

- ◇ muzzle flashes
- ◇ hand switching for selected characters

For DI-Guy Graphics API users this means that most calls to `diguyGraphicsShape::show()` and `diguyGraphicsShape::hide()` will not be made.

Note that characters that are load managed (`get_is_load_managed()` returns 1) have this parameter managed automatically.

Arguments:

`disabled_flag` pass 1 to disable shape switches, 0 to enable them

Returns:

0 on success, -1 on failure

5.12.177 function diguyCharacter::get_shape_switches_disabled**Prototype:**

```
int get_shape_switches_disabled();
```

Returns:

most recent setting of `set_shape_switches_disabled()`

5.12.178 function diguyCharacter::set_shape_callbacks_disabled**Prototype:**

```
int set_shape_callbacks_disabled(int disabled_flag);
```

Description:

Note: This function is not recommend. There is minimal perf gain.

This function disables all shape callbacks. Shape callbacks are used internally for updating some advanced visual effects.

Shape callbacks currently control:

- ◇ expressive faces
- ◇ particle systems

Disabling shape callbacks will make these visual effects work incorrectly but will result in a performance increase for characters that do not use them.

Note that characters that are load managed (`get_is_load_managed()` returns 1) have this parameter managed automatically.

Arguments:

`disabled_flag` pass 1 to disable shape callbacks, 0 to enable them

Returns:

0 on success, -1 on failure

5.12.179 function `diguyCharacter::get_shape_callbacks_disabled`

Prototype:

```
int get_shape_callbacks_disabled();
```

Returns:

most recent setting of `set_shape_callbacks_disabled()`

5.12.180 function**diguyCharacter::set_graphics_api_shape_update_disabled****Prototype:**

```
int set_graphics_api_shape_update_disabled(int disabled_flag);
```

Description:

Note: This function is not recommend. There is minimal perf gain.

This function disables all calls to the virtual
[diguyGraphicsShape::update\(\)](#) function.

Note that characters that are load managed ([get_is_load_managed\(\)](#) returns 1) have this parameter managed automatically.

Arguments:

`disabled_flag` pass 1 to disable update calls, 0 to enable them

Returns:

0 on success, -1 on failure

5.12.181 function**diguyCharacter::get_graphics_api_shape_update_disabled****Prototype:**

```
int get_graphics_api_shape_update_disabled();
```

Returns:

most recent setting of
[set_graphics_api_shape_update_disabled\(\)](#)

5.12.182 function diguyCharacter::optimized_update**Prototype:**

```
int optimized_update(float t);
```

Description:

This function is a stripped-down version of the normal `update()` function. This function is for high performance applications that are willing to sacrifice advanced character functionality in exchange for faster performance.

Characters using `optimized_update()` must not do any of the following:

- ◇ change character types (e.g., `set_character_type()`)
- ◇ play sounds (e.g., `play_sound()`)
- ◇ use history (e.g., `set_history_type()`)
- ◇ use paths (e.g., `push_path()`)
- ◇ weapon firing (e.g., `fire_weapon()`)
- ◇ advanced pose operations (as per CPL 3)
- ◇ be a DI-Guy Network Module character
- ◇ need to use character callbacks
- ◇ use advanced visual effects, such as expressive faces and particle systems

Like `update()`, the scenario must not call the character's `update()` function (return value of `get_t_controlled_by_scenario_t()` is 0) for this function to work properly.

In some cases the regular `update()` call can be used for a time in place of `optimized_update()` if one of the above operations is desired.

Arguments:

`t` new time for the character

Returns:

0 on success, -1 on failure

5.12.183 function diguyCharacter::set_is_load_managed**Prototype:**

```
int set_is_load_managed(int is_load_managed);
```

Description:

This function sets whether many of the performance tuning functions above are automatically managed by a [diguyLoadManager](#) object, which can be obtained by calling `diguyApp::create_load_manager( )`.

Note that this function will have no effect if a load manager has not been enabled.

The default for new characters is to be load managed if the load manager has been enabled.

Arguments:

<code>is_load_managed</code>	pass 1 to have character automatically load managed, 0 for manual management

Returns:

0 on success, -1 on failure

5.12.184 function diguyCharacter::get_is_load_managed**Prototype:**

```
int get_is_load_managed();
```

Returns:

most recent setting of
[set_graphics_api_shape_update_disabled\(\)](#)

Callback Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.185 enumeration

```
enum {
    CALLBACK_ID_CREATE = 1,
    CALLBACK_ID_DESTROY,
    CALLBACK_ID_PRE_CREATE,
    CALLBACK_ID_CURRENT_APPEARANCE_CHANGED,
    CALLBACK_ID_CURRENT_HEAD_APPEARANCE_CHANGED,
    CALLBACK_ID_PRE_UPDATE,
    CALLBACK_ID_POST_UPDATE,
    CALLBACK_ID_DONE_SPEAKING,
    CALLBACK_ID_GAZE_STATUS,
    CALLBACK_ID_LPOINT_STATUS,
    CALLBACK_ID_SHOW,
    CALLBACK_ID_HIDE,
    CALLBACK_ID_PRE_DIE,
    CALLBACK_ID_POST_DIE,
    CALLBACK_ID_PRE_FIRE_WEAPON,
```

DI-Guy SDK Version 12.5.2 Index

```
CALLBACK_ID_POST_FIRE_WEAPON,  
CALLBACK_ID_IMPACT,  
CALLBACK_ID_FIRE_WEAPON_SUCCESS,  
CALLBACK_ID_CURRENT_ACTION_CHANGED,  
CALLBACK_ID_DESIRED_ACTION_CHANGED,  
CALLBACK_ID_DESIRED_ACTION_REACHED,  
CALLBACK_ID_MANUALLY_INVOKED,  
CALLBACK_ID_USER_SELECTED,  
CALLBACK_ID_USER_UNSELECTED,  
CALLBACK_ID_IGUY_INTERACT,  
CALLBACK_ID_IGUY_INTERACT_WITH_SUBJECT,  
CALLBACK_ID_GUIDE_POSITION_ACQUIRED,  
CALLBACK_ID_GUIDE_POSITION_UNACQUIRED,  
CALLBACK_ID_GUIDE_ORIENTATION_ACQUIRED,  
CALLBACK_ID_GUIDE_ORIENTATION_UNACQUIRED,  
CALLBACK_ID_GUIDE_ALTITUDE_ACQUIRED,  
CALLBACK_ID_GUIDE_ALTITUDE_UNACQUIRED,  
CALLBACK_ID_CURRENT_TOUT_REACHED,  
CALLBACK_ID_POST_CREATE_GEOMETRY,  
CALLBACK_ID_PRE_DESTROY_GEOMETRY,  
CALLBACK_ID_END_OF_PATH_REACHED,  
  
// DI-Guy Scenario UI callbacks:  
CALLBACK_ID_UI_PRE_TRANSLATION,  
CALLBACK_ID_UI_POST_TRANSLATION,  
CALLBACK_ID_UI_MOUSE_DOWN,  
CALLBACK_ID_UI_MOUSE_UP,  
CALLBACK_ID_UI_RIGHT_MOUSE_DOWN,  
CALLBACK_ID_UI_RIGHT_MOUSE_UP,  
  
// DI-Guy AI agent callbacks:  
CALLBACK_ID_AGENT_NEW_BEHAVIOR,  
CALLBACK_ID_AGENT_NEW_PATH_SHAPE,  
CALLBACK_ID_AGENT_NEW_FOCUS_CHARACTER,  
CALLBACK_ID_AGENT_ATTACK_OUT_OF_TARGETS,  
CALLBACK_ID_AGENT_ATTACK_NO_VISIBLE_TARGETS,  
CALLBACK_ID_AGENT_PURSUE_TARGET_REACHED,  
CALLBACK_ID_AGENT_PURSUE_TARGET_LOST,  
CALLBACK_ID_AGENT_MINGLE_TARGET_REACHED,  
CALLBACK_ID_AGENT_MINGLE_TARGET_LOST,  
CALLBACK_ID_AGENT_WANDER_TARGET_REACHED,  
CALLBACK_ID_AGENT_WANDER_TARGET_LOST,  
CALLBACK_ID_AGENT_FLEE_AREA_LEFT,  
CALLBACK_ID_AGENT_FLEE_AREA_ENTERED,  
CALLBACK_ID_AGENT_TRAVEL_WAYPOINT_REACHED,  
CALLBACK_ID_AGENT_TRAVEL_FORWARD_DEST_REACHED,  
CALLBACK_ID_AGENT_TRAVEL_BACKWARD_DEST_REACHED,  
CALLBACK_ID_AGENT_CROWD_MEMBER_KILLED,  
CALLBACK_ID_AGENT_CROWD_MEMBER_IMPACT,  
CALLBACK_ID_AGENT_NEARBY_SCENE_OBJECT_IMPACT,  
CALLBACK_ID_AGENT_NEARBY_WEAPON_FIRED,  
CALLBACK_ID_AGENT_VEHICLE_COLLISION,  
CALLBACK_ID_AGENT_VEHICLE_NEAR_COLLISION,  
CALLBACK_ID_AGENT_PATH_PLAN_COMPLETED,  
CALLBACK_ID_AGENT_MIND_CURRENT_STATE_CHANGED,  
CALLBACK_ID_AGENT_NON_ZERO_REPULSION,  
CALLBACK_ID_AGENT_ZERO_REPULSION  
};
```

Description:

This is an enumeration of the different callbacks that can be registered with `add_callback()` and `add_callback_script()`.

Callbacks return a value of type `diguyCallbackReturn`, which will be `DIGUY_CALLBACK_STOP` or `DIGUY_CALLBACK_CONTINUE`. If the callback returns `DIGUY_CALLBACK_STOP`, in some cases the default handler of the function will not be called; the callback is asserting that it has done everything necessary for the function call. If the callback returns `DIGUY_CALLBACK_CONTINUE`, the default handler for the function will be called after the callback.

Usable From:

- ◇ C++
- ◇ Script

Callback Enums:

CALLBACK_ID_CREATE

This callback will be called when a new character is created, **after** the character is fully initialized.

Note that this callback can only be added by calling `diguyScenario::add_default_character_callback()`; adding it with `diguyCharacter::add_callback()` will have no effect, as by that time the character has already been created.

CALLBACK_ID_DESTROY

This callback will be called when a character is destroyed.

CALLBACK_ID_PRE_CREATE

Like `CALLBACK_ID_CREATE`, this callback will be called when a new character is created; however, it is called **before** the character is fully initialized.

NOTE: Care must be taken when using this callback!

As mentioned above, the character is not fully initialized, meaning that most member functions of the `diguyCharacter` object should not be called. Those member functions that are safe to call will be mentioned in the documentation for the function; e.g., `set_graphics_api_node_ptr()`.

Note that this callback can only be added by calling `diguyScenario::add_default_character_callback()`; adding it with `diguyCharacter::add_callback()` will have no effect, as by that time the character has already been created.

CALLBACK_ID_CURRENT_APPEARANCE_CHANGED

This callback will be called when a character's current appearance is changed.

CALLBACK_ID_CURRENT_HEAD_APPEARANCE_CHANGED

This callback will be called when a character's current head appearance is changed.

CALLBACK_ID_PRE_UPDATE

This callback will be called before the character is updated as a result of the `diguyScenario::update()` function.

CALLBACK_ID_POST_UPDATE

This callback will be called after the character is updated as a result of the `diguyScenario::update()` function.

CALLBACK_ID_DONE_SPEAKING

This callback will be called when the character has finished speaking the contents of a `speak()` function call.

CALLBACK_ID_GAZE_STATUS

This callback will be called after the character's gaze has experienced a status change.

CALLBACK_ID_LPOINT_STATUS

This callback will be called after the character's lpoint (left arm pointing) has experienced a status change.

CALLBACK_ID_SHOW

This callback will be called when the character is being shown for any reason.

CALLBACK_ID_HIDE

This callback will be called when the character is being hidden for any reason.

CALLBACK_ID_PRE_DIE

This callback will be called when the character has been told to die, before a die action has been selected and initiated.

If the returned `diguyCallbackReturn` value is `DIGUY_CALLBACK_STOP`, the character will not die.

CALLBACK_ID_POST_DIE

This callback will be called when the character has been told to die, after a die action has been selected and initiated.

CALLBACK_ID_PRE_FIRE_WEAPON

This callback will be called when the character has been told to fire its weapon, before a final decision has been made to fire.

If the returned `diguyCallbackReturn` value is `DIGUY_CALLBACK_STOP`, the character will not fire.

CALLBACK_ID_IMPACT

This callback will be called when the character has been hit. `diguyCharacter::get_last_impact_record()` contains a pointer to the impact information. If a character has this callback the standard behavior (killing the character) is skipped and the system assumes the end user has handled the impact.

Similar behavior results if `diguyCrowd::CALLBACK_ID_CROWD_MEMBER_IMPACT` is registered.

CALLBACK_ID_POST_FIRE_WEAPON

This callback will be called when the character has been told to fire its weapon, after a final decision has been made to fire.

CALLBACK_ID_CURRENT_ACTION_CHANGED

This callback will be called whenever the character's current action changes, typically after a call to `set_desired_action()` or `force_action()`.

CALLBACK_ID_DESIRED_ACTION_CHANGED

This callback will be called whenever the character's desired action changes, typically after a call to `set_desired_action()`.

CALLBACK_ID_DESIRED_ACTION_REACHED

This callback will be called when the character reaches its desired action as set by a call to `set_desired_action()`.

CALLBACK_ID_MANUALLY_INVOKED

This callback id will be supplied to event handlers invoked by a call to `manually_invoke_event_handler()`.

CALLBACK_ID_USER_SELECTED

This callback will be called when the character is selected in DI-Guy Scenario.

CALLBACK_ID_USER_UNSELECTED

This callback will be called on a currently selected character when a different character is selected in DI-Guy Scenario.

CALLBACK_ID_IGUY_INTERACT

This callback will be called when the character has been clicked on while DI-Guy Scenario is in I-Guy Input Mode. Note that this is called on the character clicked, not the I-Guy character.

CALLBACK_ID_IGUY_INTERACT_WITH_SUBJECT

This callback will be called when the character, as the I-Guy character, has clicked another character in I-Guy Input Mode. Note that this is called for the I-Guy character, not the clicked character.

CALLBACK_ID_GUIDE_POSITION_ACQUIRED

This callback will be called when the character has reached its desired position as set by `set_desired_position()`.

CALLBACK_ID_GUIDE_POSITION_UNACQUIRED

This callback will be called if the character moves too far away from its desired position after it has been previously reached.

CALLBACK_ID_GUIDE_ORIENTATION_ACQUIRED

This callback will be called when the character has reached its desired orientation as set by `set_desired_orientation()`.

CALLBACK_ID_GUIDE_ORIENTATION_UNACQUIRED

This callback will be called if the character turns too far away from its desired orientation after it has been previously reached.

CALLBACK_ID_GUIDE_ALTITUDE_ACQUIRED

This callback will be called when the character has reached its desired altitude as set by `set_desired_orientation()` or other functions.

CALLBACK_ID_GUIDE_ALTITUDE_UNACQUIRED

This callback will be called if the character turns too far away from its desired orientation after it has been previously reached.

CALLBACK_ID_CURRENT_TOUT_REACHED

This callback will be called when scenario time reaches this character's tout time as set by `set_current_tout()`.

CALLBACK_ID_POST_CREATE_GEOMETRY

This callback will be called just after the character's geometry has been created.

CALLBACK_ID_PRE_DESTROY_GEOMETRY

This callback will be called just before the character's geometry is broken down and destroyed.

CALLBACK_ID_END_OF_PATH_REACHED

This callback will be called when a character reaches the end of their current path. Note the path must finish, forcing to other paths will not trigger this callback.

DI-Guy Scenario UI Callback enums:*CALLBACK_ID_UI_PRE_TRANSLATION*

This callback will be called when the user interface starts moving a character.

CALLBACK_ID_UI_POST_TRANSLATION

This callback will be called when the user interface finishes moving a character.

CALLBACK_ID_UI_MOUSE_DOWN

This callback will be called when the user interface gets a mouse click in the `DIGUY_SCENARIO_INPUT_MODE_POINT_SELECT` input mode.

CALLBACK_ID_UI_MOUSE_UP

This callback will be called when the user interface gets a mouse release in the `DIGUY_SCENARIO_INPUT_MODE_POINT_SELECT` input mode.

DI-Guy AI Agent Callback enums:*CALLBACK_ID_AGENT_NEW_BEHAVIOR*

This callback will be called when an agent changes its current behavior.

CALLBACK_ID_AGENT_NEW_FOCUS_CHARACTER

This callback will be called when an agent changes its current focus character; this happens most frequently when it is focused on a group.

CALLBACK_ID_AGENT_ATTACK_OUT_OF_TARGETS

This callback will be called when an agent in attack behavior evaluates the group it's focused on and cannot find an appropriate target. This can happen if all group members are dead.

CALLBACK_ID_AGENT_ATTACK_NO_VISIBLE_TARGETS

This callback will be called when an agent in attack behavior can't see any potential members of the group it's focused on.

CALLBACK_ID_AGENT_PURSUE_TARGET_REACHED

This callback will be called when an agent in pursue behavior has come within the pursuit distance of its pursuit target.

CALLBACK_ID_AGENT_PURSUE_TARGET_LOST

This callback will be called when an agent in pursue behavior is no longer within the pursuit distance of its pursuit target.

CALLBACK_ID_AGENT_MINGLE_TARGET_REACHED

This callback will be called when an agent in mingle behavior has reached its mingle target.

CALLBACK_ID_AGENT_MINGLE_TARGET_LOST

This callback will be called when an agent in mingle behavior is no longer in range of its mingle target.

CALLBACK_ID_AGENT_WANDER_TARGET_REACHED

This callback will be called when an agent in wander behavior has reached its wander target.

CALLBACK_ID_AGENT_WANDER_TARGET_LOST

This callback will be called when an agent in wander behavior is no longer in range of its wander target.

CALLBACK_ID_AGENT_FLEE_AREA_LEFT

This callback will be called when an agent in flee behavior has successfully moved out range of all fled characters or locations. This will only be called when the agent is transitioning from inside the flee area to outside of it.

CALLBACK_ID_AGENT_FLEE_AREA_ENTERED

This callback will be called when an agent in flee behavior has moved inside of the range of any fled characters or locations. This will only be called when the agent is transitioning from outside the flee area to inside of it.

CALLBACK_ID_AGENT_TRAVEL_WAYPOINT_REACHED

This callback will be called when an agent in travel behavior has reached in intermediate waypoint and is about to move on to the next.

CALLBACK_ID_AGENT_TRAVEL_FORWARD_DEST_REACHED

This callback will be called when an agent in travel behavior has reached the end of its travel path when moving forward along the path.

CALLBACK_ID_AGENT_TRAVEL_BACKWARD_DEST_REACHED

This callback will be called when an agent in travel behavior has reached the beginning of its travel path when moving backward along the path.

CALLBACK_ID_AGENT_CROWD_MEMBER_KILLED

This callback will be called when fellow member of the crowd is killed.

- ◇ This is the character version of
diguyCrowd::CALLBACK_ID_CROWD_MEMBER_KILLED
- ◇ The crowd's callback character is the crowd member that was killed.
- ◇ The crowd's callback impact contains the impact information.

CALLBACK_ID_AGENT_CROWD_MEMBER_IMPACT

This callback will be called when a fellow member of the crowd is hit by a detonation. This callback does not replace diguyCharacter::CALLBACK_ID_IMPACT, which still needs to be handled to implement damage models.

- ◇ The crowd's callback character is the crowd member that was hit.
- ◇ The crowd's callback impact contains the impact information.

CALLBACK_ID_AGENT_NEARBY_SCENE_OBJECT_IMPACT

This callback will be called when a detonation occurs within the awareness radius (as set by set_awareness_radius()) of the crowd's current bounds.

- ◇ This is the character version of
diguyCrowd::CALLBACK_ID_NEARBY_SCENE_OBJECT_IMPACT
- ◇ The crowd's callback character is the character that caused the detonation.
- ◇ The crowd's callback impact contains the impact information.

CALLBACK_ID_AGENT_NEARBY_WEAPON_FIRED

This callback will be called when a weapon is fired within the awareness radius (as set by `set_awareness_radius( )`) of the crowd's current bounds.

◇ The crowd's callback character is the character that fired the weapon.

CALLBACK_ID_AGENT_VEHICLE_COLLISION

This callback will be called on a vehicle agent when it hits a member of a companion crowd.

CALLBACK_ID_AGENT_VEHICLE_NEAR_COLLISION

This callback will be called on a vehicle agent when it approaches another vehicle.

CALLBACK_ID_AGENT_NON_ZERO_REPULSION

This callback will be called when an agent character's repulsion forces become non-zero.

CALLBACK_ID_AGENT_ZERO_REPULSION

This callback will be called when an agent character's repulsion forces become zero.

CALLBACK_ID_AGENT_PATH_PLAN_COMPLETED

This callback will be called when a character who's requested a path plan via the background path planner (`agent_move_to_point_bg`) gets a path result. Call `diguyCharacter::get_path_planning_result( )` to get the status of the result.

CALLBACK_ID_AGENT_MIND_CURRENT_STATE_CHANGED

This callback will be called when the current state the agent's mind, if it has one, has changed. Call `diguyCharacter::get_mind_current_state_name( )` to get the new state name.

5.12.186 function diguyCharacter::add_callback**Prototype:**

```
int add_callback(int callback_id,
                diguyCharacterCallback* callback,
                void* callback_params = 0,
                void* callback_user_data = 0);
```

Description:

This function adds a user callback. Callbacks can be removed with [remove_callback\(\)](#) or [remove_callback_with_user_data\(\)](#).

Arguments:

callback	pointer to function with prototype diguyCharacterCallback (typedefed above)
callback_id	integer id of when this callback is to be called
callback_params	not currently used; pass NULL
callback_user_data	pointer for user's own use; DI-Guy will do nothing to the contents of this pointer beyond passing it back when the callback is invoked

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.12.187 function diguyCharacter::remove_callback**Prototype:**

```
int remove_callback(int callback_id,  
                    diguyCharacterCallback* callback);
```

Description:

This function removes a user callback. All callbacks matching the specified `callback_id` and callback function will be removed.

Arguments:

<code>callback_id</code>	integer id of when this callback is to be called
<code>callback</code>	pointer to function with prototype <code>diguyCharacterCallback</code> (typedefed above)

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.12.188 function**diguyCharacter::remove_callback_with_user_data****Prototype:**

```
int remove_callback_with_user_data(int callback_id,  
void* callback_user_data);
```

Description:

This function removes a user callback. All callbacks matching the specified callback_id and callback_user_data pointer will be removed.

Arguments:

callback_id	integer id of when this callback is to be called
callback_user_data	pointer for user's own use

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.12.189 function diguyCharacter::add_callback_script**Prototype:**

```
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function adds a user callback script. Callback scripts can be removed with [remove_callback_script\(\)](#).

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback to be added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

lua specific:

When the script is called, the object for which it is being called will be in the `callback_object` global.

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Lua Example:

```
-----
local pre_fire_weapon_callback = [[
local ch_name = callback_object:get_name();
bdi_log_print(BDI_LOG_WARN, "Calling pre_fire_weapon_callback for character ".. ch_name);
]]
-----

character:add_callback_script(diguyCharacter_CALLBACK_ID_PRE_FIRE_WEAPON,
    pre_fire_weapon_callback,
    "lua");
```

5.12.190 function diguyCharacter::remove_callback_script**Prototype:**

```
int remove_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function removes a user callback script previously added with [add_callback_script\(\)](#).

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback previously added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script`, all callback scripts whose ids match `callback_id` and whose types match `callback_script_type` will be removed.

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

lua specific:

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

lua Example:

```
#
# Remove all callback scripts with id
# diguyCharacter_CALLBACK_ID_PRE_FIRE_WEAPON
#
character:remove_callback_script(diguyCharacter_CALLBACK_ID_PRE_FIRE_WEAPON,
nil,
"lua");
```

Event Handler Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.191 function diguyCharacter::map_event_handler_to_callback_id

Prototype:

```
int map_event_handler_to_callback_id(int callback_id,
    const char* handler_name);
```

Description:

This function maps the event handler with the given name to a callback id. This mapping will be saved in the .dss file and restored when the .dss file is loaded.

Mappings can also be made via the DI-Guy Scenario UI.

The event handler is one of the following:

- ◇ a character callback function registered by
[diguyScenario::register_character_event_handler\(\)](#)
 or
[diguyScenario::register_character_event_handler_from_library\(\)](#)

- ◇ a script registered by
`diguyScenario::register_character_event_handler_script()`
- ◇ a Script, Decision, or Library Function in the scenario whose "Event Type" is "Character"

Arguments:

`callback_id` integer id of callback
`handler_name` name of the event handler to map

Returns:

0 on success, -1 on failure

5.12.192 function
`diguyCharacter::unmap_event_handler_from_callback_id`

Prototype:

```
int unmap_event_handler_from_callback_id(int callback_id,
    const char* handler_name,
    int unmap_all_matches = 0);
```

Description:

This function unmaps the event handler with the given name from a callback id.

Arguments:

`callback_id` integer id of callback
`handler_name` name of the event handler to unmap
`unmap_all_matches` pass 0 to unmap only the first match, pass 1 to unmap all matches

Returns:

0 on success, -1 on failure

5.12.193 function `diguyCharacter::manually_invoke_event_handler`

Prototype:

```
diguyCallbackReturn manually_invoke_event_handler(const char* handler_name);
```

Description:

This function manually invokes the named character event handler. The `callback_id` that will be passed to the event handler will be `CALLBACK_ID_MANUALLY_INVOKED`.

See

[`diguyCharacter::map\_event\_handler\_to\_callback\_id\(\)`](#)
for a description of what comprises a character event handler.

Note that care should be taken not to end up in an infinite loop of event calls. In general an event handler should not end up directly or indirectly invoking itself.

Arguments:

`handler_name` name of the event handler to invoke

Returns:

`DIGUY_CALLBACK_CONTINUE` or `DIGUY_CALLBACK_STOP`

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

Path Functions

5.12.194 function diguyCharacter::create_path**Prototype:**

```
diguyCharacterPath* create_path(const char* name);
```

Description:

This function creates a new path. The created path has no waypoints and no actions. Waypoints can be added using

`diguyCharacterPath::create_waypoint()` or `diguyCharacterPath::add_waypoint()`. Action beads should be added using `diguyCharacterPath::create_action_bead()`.

To create a simple path that has a few waypoints and actions, use `create_simple_path()`.

The waypoints for the path should be created using the function `diguyScenario::create_waypoint()`. They should be deleted using the function `diguyScenario::destroy_waypoint()`. Copies of the waypoints are made during the function call, so it is safe to destroy them immediately after this function call returns, or use the waypoints again for another path.

The returned `diguyCharacterPath` pointer should be destroyed using `destroy_path()`.

Arguments:

The path makes internal copies of all arguments.

name	name of the new path

Returns:

pointer to type `diguyCharacterPath`

Callable From:

◇ C++

5.12.195 function diguyCharacter::create_simple_path

Prototype:

```
diguyCharacterPath* create_simple_path(const char* name,
    diguyWaypoint* waypoint0,
    diguyWaypoint* waypoint1,
    const char* transition_from_action_name,
    const char* fill_action_name,
    const char* transition_to_action_name);
```

Description:

This function creates a new simple path. The path is simple because it has only two waypoints and three specified actions.

The waypoints for the path should be created using the function [diguyScenario::create_waypoint\(\)](#). They should be deleted using the function [diguyScenario::destroy_waypoint\(\)](#). Copies of the waypoints are made during the function call, so it is safe to destroy them immediately after this function call returns, or use the waypoints again for another path.

The returned [diguyCharacterPath](#) pointer should be destroyed using [destroy_path\(\)](#).

Arguments:

The path makes internal copies of all arguments.

name	name of the new path
waypoint0	beginning waypoint
waypoint1	ending waypoint
transition_from_action_name	name of the action the character should be transitioning from at the beginning of the path; can be NULL
fill_action_name	name of the action that should fill the majority of the path; can be NULL, but see below
transition_to_action_name	name of the action the character should transition into at the end of the path

If `transition_from_action_name` is NULL, an initial action bead that transitions to the fill action will not be created.

If `fill_action_name` is NULL, no action beads will be created for the path, even if `transition_from_action_name` and/or `transition_to_action_name` are non-NULL.

If `transition_to_action_name` is NULL, a final action bead that transitions from the fill action will not be created.

Returns:

pointer to type [diguyCharacterPath](#)

Callable From:

◇ C++

5.12.196 function `diguyCharacter::create_path_from_path_shape`

Prototype:

```
diguyCharacterPath* create_path_from_path_shape(const char* name,
        diguyPathShape* source_path_shape,
        const char* transition_from_action_name,
        const char* fill_action_name,
        const char* transition_to_action_name);
```

Description:

This function is similar to [create_simple_path\(\)](#), but creates its waypoints based on waypoints from the passed path shape.

Arguments:

The path makes internal copies of all arguments.

<code>name</code>	name of the new path
<code>source_path_shape</code>	path shape whose waypoints are used to define shape of new path

transition_from_action_name same as for create_simple_path()
 fill_action_name same as for create_simple_path()
 transition_to_action_name same as for create_simple_path()

Returns:

pointer to type [diguyCharacterPath](#)

Callable From:

◇ C++

5.12.197 function diguyCharacter::create_and_force_bridge_path

Prototype:

```
int create_and_force_bridge_path(const char* destination_path_name,
    const char* destination_path_action_bead_name,
    const char* via_action_name);
```

Description:

This function creates a temporary path and immediately forces it as the character's current path. Once it has been traversed it will be automatically deleted.

The beginning waypoint of this path is at the character's current position; the ending waypoint is at the specified action bead on the specified path (or the first action bead of the specified path if no action bead is specified). The character will use the specified action to move from the beginning waypoint to the ending waypoint.

This function is useful for getting a free position mode character back onto a path.

This function will have no effect on a dead character (see `die_now()` and `get_dead()`).

Mode Effects:

- ◇ This function will put the character into path position mode.
- ◇ This function will put the character into path action mode.

Arguments:

<code>destination_path_name</code>	the path the character is trying to reach
<code>via_action_name</code>	the action used over the majority of the bridge path
<code>destination_path_action_bead_name</code>	the target action bead on the destination path; pass NULL to start at the beginning of the path

Returns:

0 on success, -1 on failure

5.12.198 function `diguyCharacter::create_and_force_bridge_path_to_point`

Prototype:

```
int create_and_force_bridge_path_to_point(float x, float y, float z,
    float path_offset,
    const char* via_action_name,
    const char* next_local_path = NULL,
    float orientation = 0.0f);
```

Description:

This function creates a temporary path and immediately forces it as the character's current path. Once the path has been traversed it will be automatically deleted. The character then begins the local path specified by `path_to_push`.

The beginning waypoint of this path is at the character's current position; the ending waypoint is at the specified x, y, z. A `path_offset` is available to shorten the distance to the end point along the vector to the destination. The character will use the specified action to move from the beginning waypoint to the ending waypoint.

An optional follow-on path can be specified that the character will move onto once the end position is reached. This must be the name of an existing path for the character. A local path copy of the specified path will be created, and will be pushed to occur after this bridge path completes.

This function is useful for branching from one location to another in the world.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Mode Effects:

- ◇ This function will put the character into path position mode.
- ◇ This function will put the character into path action mode.

Arguments:

<code>x, y, z</code>	location in world space the bridge path should aim for
<code>path_offset</code>	the amount to shorten the bridge path
<code>via_action_name</code>	the action used over the majority of the bridge path
<code>next_local_path</code>	local path to push when the character arrives at it's destination; pass NULL to not push a local path
<code>orientation</code>	amount to rotate the local path

Returns:

0 on success, -1 on failure

5.12.199 function diguyCharacter::destroy_path**Prototype:**

```
int destroy_path(diguyCharacterPath* path);
```

Description:

This function destroys a path created by [create_path\(\)](#), [create_simple_path\(\)](#), or a path from a loaded scenario.

The path the character is currently traveling, as returned by [get_current_traveled_path\(\)](#), cannot be destroyed. Attempting to destroy it will result in an error.

Arguments:

path	path to be destroyed

Returns:

0 on success, -1 on failure

5.12.200 function diguyCharacter::get_num_paths**Prototype:**

```
int get_num_paths();
```

Returns:

the number of paths belonging to this character

5.12.201 function diguyCharacter::get_path_at_index**Prototype:**

```
diguyCharacterPath* get_path_at_index(int index);
```

Returns:

pointer of type [diguyCharacterPath](#); NULL if no path at the specified index

Arguments:

index	index of the path; indices start at 0

5.12.202 function diguyCharacter::get_path_name_at_index**Prototype:**

```
const char* get_path_name_at_index(int index);
```

Returns:

the name of the path at the given index; NULL if no path at the specified index

5.12.203 function diguyCharacter::get_current_path_index

Prototype:

```
int get_current_path_index();
```

Description:

This function returns the index of the path the character is currently following. If the character is not following a path the returned index will be -1. (i.e., the character must be in path action mode **and** path position mode.)

Mode Restrictions:

◇ This function will only return a 0 or greater index if the character is in path action mode **and** path position mode.

Returns:

the index of the current path; -1 if not on a path

5.12.204 function diguyCharacter::get_current_traveled_path

Prototype:

```
diguyCharacterPath* get_current_traveled_path();
```

Description:

This function returns the path the character is currently traveling.

Mode Restrictions:

◇ This function can only be called in path position mode.

Returns:

pointer of type [diguyCharacterPath](#); NULL if no path is currently being traveled

5.12.205 function diguyCharacter::get_current_traveled_path_is**Prototype:**

```
int get_current_traveled_path_is(const char* path_name);
```

Description:

This function can be used to see if the current traveled has the passed name. This is primarily useful for decisions and other limited logic applications.

Returns:

1 if currently traveled path has the passed name; 0 if the name doesn't match, or no path is currently being traveled

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.206 function diguyCharacter::push_path**Prototype:**

```
int push_path(const char* path_name);
```

Description:

This function pushes the path with the specified name onto the queue of paths this character will follow.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

name	name of the path to be pushed

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.207 function diguyCharacter::force_path

Prototype:

```
int force_path(const char* path_name,
               float max_rampdown_interval = 0.5f);
```

Description:

This function clears the queue of paths the character will follow and then pushes the path with the specified name.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

name	name of the path to be pushed
max_rampdown_interval	max amount of time spent trying to smooth over any motion seams; set to 0.5 for a smooth transition, set to 0 for a potentially rough transition

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.208 function diguyCharacter::force_partial_path

Prototype:

```
int force_partial_path(const char* path_name,
                      const char* beginning_action_bead_name,
                      float max_rampdown_interval = 0.5f);
```

Description:

Same as [force_path\(\)](#), but only motions at or after the specified action bead are added.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.209 function diguyCharacter::resume_interrupted_path**Prototype:**

```
int resume_interrupted_path(float max_rampdown_interval = 0.5f);
```

Description:

Resumes a path that was interrupted with a call such as `set_desired_action()` or `force_action()`. Note that the `retain_path_shape` argument to these functions must have been 1 for the path to be resumable.

This function will have no effect on a dead character (see `die_now()` and `get_dead()`).

Mode Restrictions:

- ◇ This function can only be called if the character is in free action mode **and** path position mode.

Mode Effects:

- ◇ This function will put the character into path position mode.
- ◇ This function will put the character into path action mode.

Arguments:

`max_rampdown_interval` max amount of time spent trying to smooth over any motion seams; set to 0.5 for a smooth transition, set to 0 for a potentially rough transition

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.210 function diguyCharacter::force_local_path**Prototype:**

```
diguyCharacterPath* force_local_path(const char* path_name,
    int use_character_orientation,
    float rotation = 0.0f,
    int clamp_path = 1);
```

Description:

This function creates a temporary path that is a copy of path_name but transformed so that the starting point is the same as where the character currently is. It then forces it as the character's current path.

Arguments:

path_name	path to copy
use_character_orientation	whether the character's orientation be incorporated into the transform of the path
rotation	additional amount to rotate the path, in degrees
clamp_path	whether the path be ground clamped

Returns:

pointer to type [diguyCharacterPath](#)

5.12.211 function diguyCharacter::set_initial_path**Prototype:**

```
int set_initial_path(const char* name);
```

Description:

This function sets which path the character will start on when the scenario is reset.

Arguments:

name	name of the path that will be the initial path

Returns:

0 on success, -1 on failure

5.12.212 function diguyCharacter::get_initial_path**Prototype:**

```
const char* get_initial_path();
```

Description:

This function returns the name of the path the character will start on when the scenario is reset.

See [set_initial_path\(\)](#).

Returns:

name of initial path; NULL if there is none

5.12.213 function diguyCharacter::translate_all_paths**Prototype:**

```
void translate_all_paths(float tx, float ty, float tz,
    int reclamp_waypoints = 1);
```

Description:

This function translates all of the paths of this character.

Arguments:

tx, ty, tz	distance in meters to move all waypoints
reclamp_waypoints	pass 1 to reclamp waypoints to ground

5.12.214 function diguyCharacter::rotate_all_paths_about_point**Prototype:**

```
int rotate_all_paths_about_point(float rz, float rx, float ry,
    float rotation_pt_x, float rotation_pt_y, float rotation_pt_z,
    int reclamp_waypoints = 1);
```

Description:

This function rotates all of the paths of this character. The rotation will occur about the specified point.

Note: It is currently recommended that only rotations about the Z axis be done. (i.e., only rz should be non-zero.)

Arguments:

rz, rx, ry	amount of rotation, in degrees
rotation_pt_x, rotation_pt_y, rotation_pt_z	point around which rotation should occur

`reclamp_waypoints` pass 1 to reclamp waypoints to ground

5.12.215 function

`diguyCharacter::rotate_all_paths_about_current_waypoint`

Prototype:

```
int rotate_all_paths_about_current_waypoint(float rz, float rx, float ry,
int reclamp_waypoints = 1);
```

Description:

This function is similar to `rotate_all_paths_about_point()`, but rotates around the character's current waypoint instead of an explicitly specified point.

Note: It is currently recommended that only rotations about the Z axis be done. (i.e., only `rz` should be non-zero.)

Arguments:

`rz, rx, ry` amount of rotation, in degrees
`reclamp_waypoints` pass 1 to reclamp waypoints to ground

5.12.216 function diguyCharacter::jump_to_action_bead**Prototype:**

```
int jump_to_action_bead(const char* bead_name,
    float max_rampdown_interval = 0.5f);
```

Description:

This function causes the character to jump forward along its current path to the specified action bead. Motions and beads that would have otherwise been played or processed will be skipped.

Arguments:

bead_name	name of the bead to jump to
max_rampdown_interval	max amount of time spent trying to smooth over any motion seams; set to 0.5 for a smooth transition, set to 0 for a potentially rough transition

Returns:

0 on success, -1 on failure

5.12.217 function diguyCharacter::find_path**Prototype:**

```
diguyCharacterPath* find_path(const char* path_name);
```

Description:

This function finds and returns a pointer to the specified path.

Arguments:

name	name of the path to be found

Returns:

Pointer to object of type [diguyCharacterPath](#); NULL if path not found.

5.12.218 function diguyCharacter::apply_aim_bead**Prototype:**

```
int apply_aim_bead(diguyCharacterPathAimBead* bead,
                  float how_far = 0.0f);
```

Description:

This function causes the effects of the given aim bead to be applied to this character, regardless of whether the bead belongs to the character or whether character is within the bead's time interval.

Arguments:

how_far	how far into the bead the call should be; should be between 0 (beginning of bead time interval) and 1 (end of bead time interval)

Returns:

0 on success, -1 on failure

5.12.219 function diguyCharacter::apply_decision_bead**Prototype:**

```
int apply_decision_bead(diguyCharacterPathDecisionBead* bead,
                       float how_far = 0.0f);
```

Description:

This function causes the effects of the given decision bead to be applied to this character, regardless of whether the bead belongs to the character or whether character is within the bead's time interval.

Arguments:

how_far	how far into the bead the call should be; should be between 0 (beginning of bead time interval) and 1 (end of bead time interval)

Returns:

0 on success, -1 on failure

5.12.220 function diguyCharacter::apply_gaze_bead**Prototype:**

```
int apply_gaze_bead(diguyCharacterPathGazeBead* bead,
    float how_far = 0.0f);
```

Description:

This function causes the effects of the given gaze bead to be applied to this character, regardless of whether the bead belongs to the character or whether character is within the bead's time interval.

Arguments:

how_far	how far into the bead the call should be; should be between 0 (beginning of bead time interval) and 1 (end of bead time interval)

Returns:

0 on success, -1 on failure

5.12.221 function diguyCharacter::apply_script_bead**Prototype:**

```
int apply_script_bead(diguyCharacterPathScriptBead* bead,
    float how_far = 0.0f);
```

Description:

This function causes the effects of the given script bead to be applied to this character, regardless of whether the bead belongs to the character or whether character is within the bead's time interval.

Arguments:

how_far	how far into the bead the call should be; should be between 0 (beginning of bead time interval) and 1 (end of bead time interval)

Returns:

0 on success, -1 on failure

5.12.222 function diguyCharacter::set_distance_along_path**Prototype:**

```
int set_distance_along_path(float distance);
```

Description:

This function sets how far along the current path shape the character will be.

Mode Restrictions:

◇ This function can only be called if the character is in free action mode **and** path position mode. i.e., the character is following a path shape, and is not on a [diguyCharacterPath](#).

Arguments:

`distance` distance in meters; 0.0 is the beginning of the path shape

Returns:

0 on success, -1 on failure

5.12.223 function diguyCharacter::get_distance_along_path**Prototype:**

```
float get_distance_along_path();
```

Returns:

the distance in meters along the path shape; see [set_distance_along_path\(\)](#)

Mode Restrictions:

◇ This function can only be called in path position mode.

5.12.224 function diguyCharacter::leave_path**Prototype:**

```
int leave_path();
```

Description:

This function takes the character off of its current path or path shape.

Mode Restrictions:

◇ This function can only be called in path position mode.

Mode Effects:

◇ This function will put the character into free position mode.

◇ This function will put the character into free action mode.

Returns:

0 on success, -1 on failure

5.12.225 function diguyCharacter::get_nearest_waypoint_index_in_path**Prototype:**

```
int get_nearest_waypoint_index_in_path(const char* path_name);
```

Description:

This function returns the index of the nearest waypoint in a path, the path argument is looked up in both the character's path list and the scenario's paths.

Returns:

waypoint index

5.12.226 function diguyCharacter::get_distance_to_path**Prototype:**

```
float get_distance_to_path(const char* path_name,  
    const char* action_bead_name = NULL);
```

Returns:

Distance to action bead or first waypoint, 100000000.0 on failure

Link and Shape Functions

The following functions provide access to the links (bones) and shapes (visible geometry) of the character.

5.12.227 function diguyCharacter::get_position_link**Prototype:**

```
diguyGraphicsLink* get_position_link();
```

Description:

This function gets the position link of the character skeleton. The rest of the skeleton can then be traversed by recursively calling the `diguyGraphicsLink::get_num_child_links()` and `diguyGraphicsLink::get_child_link_at_index()` functions.

Alternatively, the links of the character can be traversed by calling the `get_num_links()` and `get_link_at_index()` functions below.

Returns:

pointer of type `diguyGraphicsLink`; NULL if there is no position link

5.12.228 function diguyCharacter::get_num_links**Prototype:**

```
int get_num_links();
```

Returns:

number of links on this character

5.12.229 function diguyCharacter::get_link_at_index**Prototype:**

```
diguyGraphicsLink* get_link_at_index(int index);
```

Returns:

pointer of type [diguyGraphicsLink](#); NULL if no link at the specified index

Arguments:

index	index of the link; indices start at 0

5.12.230 function diguyCharacter::find_link**Prototype:**

```
diguyGraphicsLink* find_link(const char* name);
```

Description:

This function returns a pointer to the specified link.

Arguments:

name	name of link to be found

Returns:

pointer of type [diguyGraphicsLink](#); NULL if not found

5.12.231 function diguyCharacter::get_link_name_at_index**Prototype:**

```
const char* get_link_name_at_index(int link_index);
```

Returns:

the string identifying the link at the given index

Arguments:

`link_index` index of the link; indices start at 0

5.12.232 function diguyCharacter::get_num_link_shapes**Prototype:**

```
int get_num_link_shapes(const char* link_name);
```

Returns:

the number of shapes on the specified link

Arguments:

`link_name` name of the link

5.12.233 function**diguyCharacter::get_link_shape_name_at_index****Prototype:**

```
const char* get_link_shape_name_at_index(const char* link_name,
                                         int shape_index);
```

Returns:

the name of the shape at the specified index on the link

Arguments:

`link_name` name of the link

`shape_index` index of the shape on the link; indices start at 0

Altitude Functions

While a character is in path position mode its altitude (position in z direction) is most often determined by the path. Characters in free position mode often need their altitude to be determined by another method. This can be done by either setting the position manually (e.g., via the [set_position\(\)](#) call) or semi-automatically by registering an altitude function.

An altitude function is a callback function that gets called once per character update. Based on the character's updated x and y position, the callback can calculate and return an appropriate z position. This is commonly done via some type of graphics intersection testing.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.234 function `diguyCharacter::set_altitude_function`

Prototype:

```
int set_altitude_function(diguyAltitudeFunction* altitude_function);
```

Description:

The following prototype should be used for the altitude function:

```
float altitude_func(diguyCharacter* character,  
    float x,  
    float y,  
    float old_z);
```

When a character needs new altitude info, DI-Guy calls this function with a pointer to the `diguyCharacter` in question, the character's current (x, y) coordinates, and its previous altitude in `old_z`.

The altitude function should then compute and return the new altitude based on this information. This can be done in a number of ways; a typical approach is to do an intersection test with the visual scene.

A default altitude function that should be set for all characters can be set by calling `diguyScenario::set_default_character_altitude_function()`.

Callable From:

◇ C++

5.12.235 function diguyCharacter::get_altitude_function**Prototype:**

```
diguyAltitudeFunction* get_altitude_function();
```

Returns:

current altitude function; see [set_altitude_function\(\)](#)

Callable From:

◇ C++

5.12.236 function diguyCharacter::set_altitude_function_enabled_in_free_position_**Prototype:**

```
int set_altitude_function_enabled_in_free_position_mode(int enabled);
```

Description:

This function sets whether the character's altitude function is enabled in free position mode. Altitude functions are set by the functions [set_altitude_function\(\)](#) or [diguyScenario::set_default_character_altitude_function\(\)](#).

Arguments:

enabled pass 1 to enable, 0 to disable

Mode Effects:

◇ This function will not change the position mode.

Mode Restrictions:

◇ This function will only affect characters in free position mode.

Returns:

0 on success, -1 on failure

5.12.237 function `diguyCharacter::get_altitude_function_enabled_in_free_position_`

Prototype:

```
int get_altitude_function_enabled_in_free_position_mode();
```

Returns:

whether the character's altitude function is enabled in free position mode; see `set_altitude_function_enabled_in_free_position_mode()`

5.12.238 function**diguyCharacter::set_altitude_function_enabled_in_path_position_****Prototype:**

```
int set_altitude_function_enabled_in_path_position_mode(int value);
```

Description:

This function sets whether the character's altitude function is enabled in path position mode. Altitude functions are set by the functions [set_altitude_function\(\)](#) or [diguyScenario::set_default_character_altitude_function\(\)](#).

Arguments:

enabled pass 1 to enable, 0 to disable

Mode Effects:

◇ This function will not change the position mode.

Mode Restrictions:

◇ This function will only affect characters in path position mode.

Returns:

0 on success, -1 on failure

5.12.239 function**diguyCharacter::get_altitude_function_enabled_in_path_position_****Prototype:**

```
int get_altitude_function_enabled_in_path_position_mode();
```

Returns:

whether the character's altitude function is enabled in path position mode; see [set_altitude_function_enabled_in_path_position_mode\(\)](#)

5.12.240 function diguyCharacter::set_altitude_update_rate**Prototype:**

```
int set_altitude_update_rate(int update_rate);
```

Description

This function sets the altitude update rate of the character.

The default value for new characters is 100.

Note that characters that are load managed ([get_is_load_managed\(\)](#) returns 1) have this parameter managed automatically. The corresponding function for load managed characters is [diguyLoadManager::set_zone_altitude_update_rate\(\)](#).

Arguments:

update_rate percentage of scenario ticks for which altitude should update; can be between 1 and 100

Passing 100 means altitude will update every tick. Passing 50 means altitude will update every other tick.

Returns:

0 on success, -1 on failure

5.12.241 function diguyCharacter::get_altitude_update_rate

Prototype:

```
int get_altitude_update_rate();
```

Returns:

character's altitude update rate as set by
[set_altitude_update_rate\(\)](#)

5.12.242 function diguyCharacter::set_altitude_interpolation_rate**Prototype:**

```
void set_altitude_interpolation_rate(float interpolation_rate);
```

Description

This function sets the altitude interpolation rate of the character. This function only has an effect if the character's altitude function is enabled.

When the altitude function determines that the character's Z position should be updated to bring it closer to the ground, the altitude interpolation rate determines how quickly the Z height is adjusted.

The default value is 10.

Arguments:

`interpolation_rate` how much of the error will be made up each second

An `interpolation_rate` of 100 means that all error will be made up each frame.

5.12.243 function diguyCharacter::get_altitude_interpolation_rate**Prototype:**

```
float get_altitude_interpolation_rate();
```

Returns:

character's altitude update rate as set by
`set_altitude_interpolation_rate()`

Pose Functions

These functions are used to query a character's joint angles, and set them to the joint angles to arbitrary values.

Pose Override Stages

The DI-Guy motion engine applies pose overrides to the base motion of the character in stages. Overrides happening at later stages can overwrite those at earlier stages. The override stages are as follows:

1. reserved for user
2. lpoint base gesture (see [lpoint_at_angle\(\)](#))
3. gaze/lpoint: back, cervical, shoulder_l, elbow_l joints (see [gaze_at_angle\(\)](#))
4. default of user pose overrides (see [create_pose_override\(\)](#))
5. gestures (see [create_gesture\(\)](#))
6. nodding and shaking of head (see [nod_head\(\)](#))
7. motion texture (see [add_motion_texture\(\)](#))
8. gaze: eyes (see [gaze_at_angle\(\)](#))
9. (nothing)
10. reserved for user

Notice that eye gaze happens at a late stage, so that the eyes can still track a gaze target even when the character is nodding, has a motion texture, is gesturing, etc.

Stages 0 and 9 are reserved for the user. Overrides happening at stage 9 will have precedent over every other pose override function.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

Pose Override C++ Example

The following example shows how to use some of the pose functions.

```
diguyScenario* scenario;
diguyCharacter* soldier;
diguyCharacterPoseOverride* pose_override;

// ... (Create and initialize the scenario)

//
// Create a character.
//
soldier = scenario->create_character("joe", "soldier");
```

```

//
// Create a pose override for that character.
//
pose_override = soldier->create_pose_override();

//
// Allocate a list of joint angle names and fill it.
// Print the joint angle names, then destroy the array.
//
char** varnames = pose_override->allocate_var_names();
pose_override->get_var_names(varnames);
for (i = 0; i < soldier->get_pose_array_size(); i++)
    bdi_log_printf(BDI_LOG_INFO, "%s\n", varnames[i]);
pose_override->free_var_names(varnames);

//
// Allocate a new array of pose angles.
// Allocate a weights_array, and initialize it to all 0s.
//
pose_array = pose_override->allocate_pose_array();

weights_array = pose_override->allocate_pose_array();
for (i=0; i<soldier->get_pose_array_size(); i++)
    weights_array[i] = 0.0f;

//
// Find the indices of the character's back angles. Set the
// weights for these angles in weights_array to 1. This
// means that when we call set_pose_in_radians(), the values
// in pose_array will completely override the original
// data for the back.
//
back_rz_index = pose_override->get_var_index("q.back_rz");
back_rx_index = pose_override->get_var_index("q.back_rx");
back_ry_index = pose_override->get_var_index("q.back_ry");

pose_array[back_rz_index] = 0.57f; // angle measured in radians
pose_array[back_rx_index] = 0.0f; // angle measured in radians
pose_array[back_ry_index] = 0.0f; // angle measured in radians

weights_array[back_rz_index] = 1.0f;
weights_array[back_rx_index] = 1.0f;
weights_array[back_ry_index] = 1.0f;

pose_override->set_pose_in_radians(pose_array,
    weights_array,
    0.0f);

... // (Rest of program runs)

//
// Destroy the pose override object and the arrays
// it allocated.
//
pose_override->free_pose_array(pose_array);
pose_override->free_pose_array(weights_array);
soldier->destroy_pose_override(pose_override);

```

5.12.244 function diguyCharacter::get_pose_array_size**Prototype:**

```
int get_pose_array_size();
```

Description:

Returns the number of variables in this character's pose array. See [get_pose_in_radians\(\)](#) for an example of use.

Returns:

size of this character's pose array

5.12.245 function diguyCharacter::get_pose_in_radians**Prototype:**

```
int get_pose_in_radians(float* pose_array);
```

Description:

Fills an array of floats with the character's current joint angles. Angles are expressed in radians. The size of the float array must be at least as large as that returned by [get_pose_array_size\(\)](#).

Arguments:

`pose_array` an array of floats

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

C++ Example:

```

diguyCharacter* ch = scenario->get_character_at_index(0);
float* pose_array;

// allocate a float array of the proper size
int n = ch->get_pose_array_size();
pose_array = new float[n];

// get the pose array
ch->get_pose_in_radians(pose_array);

```

5.12.246 function diguyCharacter::create_pose_override

Prototype:

```
diguyCharacterPoseOverride* create_pose_override(int override_stage = 3);
```

Description:

Creates and returns a pose override object. The newly created pose override object is added to the character's previously created override objects, if any.

How the pose override interacts with other effects that affect the pose of the character depends on the override stage. See the section "Pose Override Stages" for a discussion of pose override stages.

Arguments:

override_stage the stage in the DI-Guy motion engine at which the override should be applied

Returns:

object of type [diguyCharacterPoseOverride](#)

5.12.247 function diguyCharacter::destroy_pose_override

Prototype:

```
int destroy_pose_override(diguyCharacterPoseOverride* po);
```

Description:

Destroys a pose override object, ending its influence on the character.

Arguments:

po	pose override object to be destroyed

Returns:

0 on success, -1 on failure

5.12.248 function diguyCharacter::get_num_pose_overrides**Prototype:**

```
int get_num_pose_overrides();
```

Returns:

the number of pose overrides belonging to this character

5.12.249 function diguyCharacter::get_pose_override_at_index**Prototype:**

```
diguyCharacterPoseOverride* get_pose_override_at_index(int index);
```

Returns:

pointer of type [diguyCharacterPoseOverride](#); NULL if no pose override at the specified index

Arguments:

index	index of the pose override; indices start at 0

5.12.250 function

`diguyCharacter::create_complete_pose_override`

Prototype:

```
diguyCharacterPoseOverride* create_complete_pose_override();
```

Description:

Creates and returns a pose override object that completely overrides all other character pose operations, including motions, aiming, gazing, gestures, other pose overrides, etc. The values set in the complete pose override object will be the values used to animate the character.

Having only one thing that affects a character's pose allows for much faster updating of the character.

There can only be one complete pose override object per character.

To remove the complete pose override, call `destroy_complete_pose_override()`.

Returns:

object of type `diguyCharacterPoseOverride`

5.12.251 function `diguyCharacter::destroy_complete_pose_override`

Prototype:

```
void destroy_complete_pose_override();
```

Description:

Removes and destroys the complete pose override created by `create_complete_pose_override()`.

Gaze and Point Functions

5.12.252 function diguyCharacter::gaze_at_angle

Prototype:

```
int gaze_at_angle(float azimuth,
                  float elevation,
                  float distance,
                  int is_new_gaze = 1);
```

Description:

This function sets the current azimuth, elevation, and distance of the character's gaze. A fixed point in 3D space is then found using the two angles and the given distance from the *Boston Dynamics* coordinate system origin. Angles are given with respect to the *Boston Dynamics* coordinate system.

See [set_position\(\)](#) for a description of the coordinate system.

Gazing will continue until [end_gaze\(\)](#) is called.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

<code>azimuth</code>	rotation in degrees about up (vertical) axis; positive rotates to character's left
<code>elevation</code>	rotation in degrees about forward (horizontal) axis; positive rotates down
<code>distance</code>	distance in meters from the position of the character to the gaze target; positive values only, typically this is a large value (e.g. 10,000 meters)
<code>is_new_gaze</code>	set to 1 if this is a new gaze; set to 0 if this is continuing an already started gaze

Returns:

0 on success, -1 on failure

5.12.253 function diguyCharacter::gaze_at_angle_local**Prototype:**

```
int gaze_at_angle_local(float azimuth,
                        float elevation,
                        float distance = 10000.0f,
                        int is_new_gaze = 1);
```

Description:

This function is identical to [gaze_at_angle\(\)](#) except that it uses body local coordinates for the character. The angles are given with respect to the idealized position of the character. Use [get_position\(\)](#) to learn the idealized position. The distance is between the idealized position of the character and the fixation point.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

5.12.254 function diguyCharacter::gaze_at_point**Prototype:**

```
int gaze_at_point(float tx,
                  float ty,
                  float tz,
                  int is_new_gaze = 1);
```

Description:

This function sets the current fixation point of the character's gaze. Point coordinates are given in meters in the *Boston Dynamics* global coordinate system. The function uses default gaze behavior settings. Gazing will continue until [end_gaze\(\)](#) is called.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

`tx, ty, tz` position in meters from the *Boston Dynamics* global origin

`is_new_gaze` set to 1 if this is a new gaze; set to 0 if this is continuing an already started gaze

Returns:

0 on success, -1 on failure

5.12.255 function `diguyCharacter::gaze_at_point_local`

Prototype:

```
int gaze_at_point_local(float tx,
    float ty,
    float tz,
    int is_new_gaze = 1);
```

Description:

This function is identical to `gaze_at_point()` except that it uses body local coordinates for the character. Use `get_position()` to learn the idealized position of the character if necessary.

This function will have no effect on a dead character (see `die_now()` and `get_dead()`).

5.12.256 function diguyCharacter::gaze_at_character**Prototype:**

```
int gaze_at_character(const char* target_character_name,
    const char* target_character_link_name = NULL,
    float offset_tx = 0.0f,
    float offset_ty = 0.0f,
    float offset_tz = 0.0f);
```

Description:

This function effectively does a [gaze_at_point\(\)](#) each tick. The point to be gazed at is updated each tick to be the position of the specified link on the specified character.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

<code>target_character_name</code>	name of the character to be gazed at
<code>target_character_link_name</code>	link on the character that should be gazed at; pass NULL to gaze at the position link of the character
<code>offset_tx, offset_ty, offset_tz</code>	offset onto the link, in meters

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

C++ Example:

```
// gaze at the head of character "soldier-1".
ch->gaze_at_character("soldier-1",
    "cervical",
    0.0,
    0.0,
    0.0);
```

5.12.257 function diguyCharacter::get_gaze_is_active**Prototype:**

```
int get_gaze_is_active();
```

Returns:

1 if the character is actively gazing, 0 if not

5.12.258 function diguyCharacter::get_gaze_point**Prototype:**

```
int get_gaze_point(float* x, float* y, float* z);
```

Returns:

the current gaze point relative to the *Boston Dynamics* global coordinate system

5.12.259 function diguyCharacter::get_gaze_is_acquired

Prototype:

```
int get_gaze_is_acquired();
```

Description:

A gaze is acquired if the final link in the gaze link chain has fully oriented toward the gaze target, even if other links in the gaze chain are still moving.

If the gaze is acquired, but any links in the gaze chain are still moving, the gaze is considered acquired but not steady (see [get_gaze_is_steady\(\)](#).) If all links have reached a relatively steady state, the gaze is considered acquired and steady.

If either the acquired or steady status of a gaze has changed (e.g., was acquired but now is not, or was steady but now a link is moving), the [diguyCharacter](#) callback with callback_id CALLBACK_ID_GAZE_STATUS will be called.

Returns:

- ◇ 1 if the final link in the gaze can and has become oriented toward the gaze target
- ◇ 0 if not
- ◇ -1 on error

5.12.260 function diguyCharacter::get_gaze_is_steady**Prototype:**

```
int get_gaze_is_steady();
```

Description:

A gaze is steady if it is acquired, and if all links involved in the gaze have reached a steady state.

For example, since the eyes move faster than the neck, which moves faster than the back, it's possible for the gaze to be acquired (see [get_gaze_is_acquired\(\)](#)) by the eyes as the neck and back are still moving. As the neck and back move into a more natural position, the eyes will not need to be turned as far to keep the gaze acquired.

If either the acquired or steady status of a gaze has changed (e.g., was acquired but now is not, or was steady but now a link is moving), the [diguyCharacter](#) callback with callback_id `CALLBACK_ID_GAZE_STATUS` will be called.

Returns:

- ◇ 1 if all links participating in the gaze have reached a relatively steady state
- ◇ 0 if not
- ◇ -1 on error

5.12.261 function diguyCharacter::end_gaze**Prototype:**

```
int end_gaze();
```

Description:

This function removes the fixation target of an ongoing gaze, allowing the character to return to a non-gaze posture and behavior. This effectively frees the character's posture to other influences.

5.12.262 function diguyCharacter::lpoint_at_angle**Prototype:**

```
int lpoint_at_angle(float azimuth,
    float elevation,
    float distance,
    int is_new_lpoint = 1,
    const char* base_gesture_name = 0);
```

Description:

This function is similar to [gaze_at_angle\(\)](#), but sets the target of left pointing rather than gaze.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

<code>azimuth</code>	rotation in degrees about up (vertical) axis; positive rotates to character's left
<code>elevation</code>	

	rotation in degrees about forward (horizontal) axis; positive rotates down
<code>distance</code>	distance in meters from the position of the character to the lpoint target; positive values only
<code>is_new_lpoint</code>	set to 1 if this is a new lpoint; set to 0 if this is continuing an already started lpoint
<code>base_gesture_name</code>	the underlying gesture that should be used for motion of the arm; this is a placeholder argument for future functionality and should be set to NULL

5.12.263 function `diguyCharacter::lpoint_at_angle_local`

Prototype:

```
int lpoint_at_angle_local(float azimuth,
    float elevation,
    float distance = 10000.0f,
    int is_new_lpoint = 1,
    const char* base_gesture_name = 0);
```

Description:

This function is identical to `gaze_at_local( )` except that it uses body local coordinates for the character. Use `get_position( )` to learn the idealized position of the character if necessary.

This function will have no effect on a dead character (see `die_now( )` and `get_dead( )`).

5.12.264 function diguyCharacter::lpoint_at_point**Prototype:**

```
int lpoint_at_point(float tx,
    float ty,
    float tz,
    int is_new_lpoint = 1,
    const char* base_gesture_name = 0);
```

Description:

This function sets the current fixation point for the character's left pointing. Point coordinates are given in meters in the *Boston Dynamics* global coordinate system.

See [set_position\(\)](#) for a description of the coordinate system.

Pointing will continue until [end_lpoint\(\)](#) is called.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

tx, ty, tz	position in meters from the <i>Boston Dynamics</i> global origin of the lpoint target
is_new_gaze	set to 1 if this is a new lpoint; set to 0 if this is continuing an already started lpoint
base_gesture_name	the underlying gesture that should be used for motion of the arm; this is a placeholder argument for future functionality and should be set to NULL

Returns:

0 on success, -1 on failure

5.12.265 function diguyCharacter::lpoint_at_point_local**Prototype:**

```
int lpoint_at_point_local(float tx,
    float ty,
    float tz,
    int is_new_lpoint = 1,
    const char* base_gesture_name = 0);
```

Description:

This function is identical to `lpoint_at_local()` except that it uses body local coordinates for the character. Use `get_position()` to learn the idealized position of the character if necessary.

This function will have no effect on a dead character (see `die_now()` and `get_dead()`).

5.12.266 function diguyCharacter::lpoint_at_character**Prototype:**

```
int lpoint_at_character(const char* target_character_name,
    const char* target_character_link_name = NULL,
    float offset_tx = 0.0f,
    float offset_ty = 0.0f,
    float offset_tz = 0.0f);
```

Description:

This function is identical to `gaze_at_character()`, but sets the target of left pointing rather than gaze.

This function will have no effect on a dead character (see `die_now()` and `get_dead()`).

5.12.267 function diguyCharacter::get_lpoint_point**Prototype:**

```
int get_lpoint_point(float* x, float* y, float* z);
```

Returns:

the current lpoint target point relative to the *Boston Dynamics* global coordinate system

5.12.268 function diguyCharacter::get_lpoint_base_gesture_name**Prototype:**

```
const char* get_lpoint_base_gesture_name();
```

Returns:

the current base gesture of the lpoint

5.12.269 function diguyCharacter::get_lpoint_is_active**Prototype:**

```
int get_lpoint_is_active();
```

Returns:

1 if the character is actively lpointing, 0 if not

5.12.270 function diguyCharacter::get_lpoint_is_acquired**Prototype:**

```
int get_lpoint_is_acquired();
```

Returns:

1 if the final link in the lpoint can and has become oriented toward the lpoint target; 0 if not; -1 on error

5.12.271 function diguyCharacter::get_lpoint_is_steady**Prototype:**

```
int get_lpoint_is_steady();
```

Returns:

1 if all links participating in the lpoint have reached a relatively steady state;
0 if not; -1 on error

5.12.272 function diguyCharacter::end_lpoint**Prototype:**

```
int end_lpoint();
```

Description:

This function removes the fixation target of an ongoing lpoint, allowing the character to return to a non-point posture and behavior. This effectively frees the character's posture to other influences.

5.12.273 function diguyCharacter::set_gaze_params_to_defaults**Prototype:**

```
int set_gaze_params_to_defaults();
```

Description:

This function sets all gaze control parameters back to their default settings. This undoes modifications to gaze control parameters made by the following calls:

```
◇ set_gaze_param_desired_locate_time()
◇ set_gaze_link_uses_desired_locate_time()
◇ set_gaze_link_azimuth_limits()
◇ set_gaze_link_elevation_limits()
◇ set_gaze_link_azimuth_tolerance()
◇ set_gaze_link_elevation_tolerance()
◇ set_gaze_link_max_angular_vel()
◇ set_gaze_link_scale_factor()
```

Returns:

0 on success, -1 on failure

5.12.274 function diguyCharacter::set_gaze_param_desired_locate_time**Prototype:**

```
int set_gaze_param_desired_locate_time(float t);
```

Description:

This function sets the desired amount of time in seconds it will take for the gaze or point target to be acquired.

Having a desired locate time effectively limits how fast links in the gaze chain can turn; they will turn as fast as it takes for them to arrive at their proper values, or as fast as their maximum angular velocity, whichever is longer.

The use of desired locate time can be turned on and off on a link by link bases using the function

```
set_gaze_link_uses_desired_locate_time().
```

This value is only a guideline and may be overridden on a link by link basis

during the actual gaze, as links try not to exceed their maximum angular velocities.

The default locate time is 1.0 seconds.

Arguments:

t lower value results in faster turning rate

Returns:

0 on success, -1 on failure

5.12.275 function diguyCharacter::get_gaze_param_desired_locate_time

Prototype:

```
float get_gaze_param_desired_locate_time();
```

Returns:

the most recent setting of
`set_gaze_param_desired_locate_time()`

5.12.276 function**diguyCharacter::set_gaze_link_uses_desired_locate_time****Prototype:**

```
int set_gaze_link_uses_desired_locate_time(const char* link_name,
int uses_desired_locate_time);
```

Description:

This function sets whether the specified link will pay attention to the desired locate time (as set by [set_gaze_param_desired_locate_time\(\)](#)), or whether the link turns as fast as its maximum angular velocity allows (as set by [set_gaze_link_max_angular_vel\(\)](#)).

Note that if this value is set to 1, the link still pays attention to its maximum angular velocity as an upper limit on how fast it can turn.

Valid link names and defaults:

```
orientation 1
back        1
cervical    1
eye_l       0
shoulder_l  1
elbow_l     1
```

Arguments:

link_name	name of the link
uses_desired_locate_time	1 to use desired locate time, 0 to disregard

Returns:

0 on success, -1 on failure

5.12.277 function diguyCharacter::get_gaze_link_uses_desired_locate_time

Prototype:

```
int get_gaze_link_uses_desired_locate_time(const char* link_name);
```

Returns:

the most recent setting of
[set_gaze_link_uses_desired_locate_time\(\)](#)

5.12.278 function diguyCharacter::set_gaze_link_azimuth_limits

Prototype:

```
int set_gaze_link_azimuth_limits(const char* link_name, float min, float max);
```

Description:

This function sets the maximum amount the character can turn the specified link horizontally in pursuit of a gaze or point target.

Valid link names and defaults:

back	-45 to 45
cervical	-80 to 80
eye_l	-50 to 50
shoulder_l	-30 to 135
elbow_l	-30 to 10

Arguments:

<code>link_name</code>	name of the link
<code>min</code>	value ≤ 0 ; in degrees
<code>max</code>	value ≥ 0 ; in degrees

Returns:

0 on success, -1 on failure

5.12.279 function `diguyCharacter::get_gaze_link_azimuth_limits`

Prototype:

```
int get_gaze_link_azimuth_limits(const char* link_name,
    float* min,
    float* max);
```

Returns:

the most recent setting of `set_gaze_link_azimuth_max( )`

5.12.280 function diguyCharacter::set_gaze_link_elevation_limits**Prototype:**

```
int set_gaze_link_elevation_limits(const char* link_name,
    float min,
    float max);
```

Description:

This function sets the maximum amount the character can turn the specified link vertically in pursuit of a gaze or point target.

Valid link names and defaults:

back	-15 to 45
cervical	-80 to 45
eye_l	-40 to 40
shoulder_l	-80 to 80
elbow_l	-170 to 0

Arguments:

link_name	name of the link
min	value ≤ 0 ; in degrees
max	value ≥ 0 ; in degrees

Returns:

0 on success, -1 on failure

5.12.281 function

diguyCharacter::get_gaze_link_elevation_limits

Prototype:

```
int get_gaze_link_elevation_limits(const char* link_name,
    float* min,
    float* max);
```

Returns:

the most recent setting of set_gaze_link_elevation_max()

5.12.282 function

diguyCharacter::set_gaze_link_azimuth_tolerance

Prototype:

```
int set_gaze_link_azimuth_tolerance(const char* link_name,
    float min,
    float max);
```

Description:

This function sets how much difference there can be between the link's current horizontal orientation and the gaze or point target before the link begins to turn to acquire the gaze or point target.

The combination of this function and [set_gaze_link_scale_factor\(\)](#) determine how close the link will turn toward the gaze or point target.

With a looser (higher) tolerance the link will not turn as much toward the target. With a tighter (lower) tolerance the link will turn more toward the target.

Valid link names and defaults:

orientation	-30 to 30
back	-45 to 45
cervical	-5 to 5
eye_l	-1 to 1
shoulder_l	-5 to 5
elbow_l	-5 to 5

Arguments:

link_name	name of the link
min	value ≤ 0 ; in degrees
max	value ≥ 0 ; in degrees

Returns:

0 on success, -1 on failure

5.12.283 function diguyCharacter::get_gaze_link_azimuth_tolerance

Prototype:

```
int get_gaze_link_azimuth_tolerance(const char* link_name,
    float* min,
    float* max);
```

Returns:

the most recent setting of set_gaze_link_azimuth_max()

5.12.284 function

diguyCharacter::set_gaze_link_elevation_tolerance

Prototype:

```
int set_gaze_link_elevation_tolerance(const char* link_name,
    float min,
    float max);
```

Description:

The elevation equivalent of
[set_gaze_link_azimuth_tolerance\(\)](#),

Valid link names and defaults:

orientation	-90 to 90
back	-45 to 45
cervical	-5 to 5
eye_l	-1 to 1
shoulder_l	-5 to 5
elbow_l	-5 to 5

Arguments:

link_name	name of the link
min	value <= 0; in degrees
max	value >= 0; in degrees

Returns:

0 on success, -1 on failure

5.12.285 function**diguyCharacter::get_gaze_link_elevation_tolerance****Prototype:**

```
int get_gaze_link_elevation_tolerance(const char* link_name,
    float* min,
    float* max);
```

Returns:

the most recent setting of set_gaze_link_elevation_max ()

5.12.286 function**diguyCharacter::set_gaze_link_max_angular_vel****Prototype:**

```
int set_gaze_link_max_angular_vel(const char* link_name, float angular_vel);
```

Description:

This function sets the maximum angular velocity that the back link angles may attain in pursuit of a gaze or point target.

Valid link names and defaults:

back	30 deg/s
cervical	60 deg/s
eye_l	180 deg/s
shoulder_l	180 deg/s
elbow_l	180 deg/s

Arguments:

`link_name` name of the link
`angular_vel` angular velocity in degrees per second

Returns:

0 on success, -1 on failure

5.12.287 function `diguyCharacter::get_gaze_link_max_angular_vel`

Prototype:

```
int get_gaze_link_max_angular_vel(const char* link_name, float* angular_vel);
```

Returns:

the most recent setting of `set_gaze_link_max_angular_vel()`

5.12.288 function diguyCharacter::set_gaze_link_scale_factor**Prototype:**

```
int set_gaze_link_scale_factor(const char* link_name, float scale_factor);
```

Description:

This function sets how much the link will make up the difference toward the gaze or point target.

The combination of this function and the tolerances determine how close the link will turn toward the gaze or point target.

With a smaller scale factor the link will not turn as much toward the target. With a higher scale factor the link will turn more toward the target.

For example, if the gaze or point target is at 60 degrees yaw in back coordinates and the scale factor is 0.5, the back will turn 30 degrees in pursuit of the target.

Valid link names and defaults:

back	0.75
cervical	0.5
eye_l	1.0
shoulder_l	0.9
elbow_l	1.0

Arguments:

link_name	name of the link
scale_factor	scale factor between 0 and 1

Returns:

0 on success, -1 on failure

5.12.289 function diguyCharacter::get_gaze_link_scale_factor**Prototype:**

```
int get_gaze_link_scale_factor(const char* link_name, float* scale_factor);
```

Returns:

the most recent setting of `set_gaze_link_scale_factor()`

5.12.290 function diguyCharacter::set_link_enabled_for_gaze**Prototype:**

```
int set_link_enabled_for_gaze(const char* link_name, int enabled);
```

Description:

This function sets whether the character can turn the specified link in pursuit of the gaze target.

Valid link names and defaults:

orientation	0
back	1
cervical	1
eye_l	1 if character has movable eyes, else 0
shoulder_l	0
elbow_l	0

Arguments:

<code>link_name</code>	name of the link
<code>enabled</code>	1 to enable, 0 to disable

Returns:

0 on success, -1 on failure

5.12.291 function `diguyCharacter::get_link_enabled_for_gaze`

Prototype:

```
int get_link_enabled_for_gaze(const char* link_name);
```

Returns:

the most recent setting of `set_link_enabled_for_gaze()`

5.12.292 function

`diguyCharacter::set_gaze_can_change_posture`

Prototype:

```
int set_gaze_can_change_posture(int enabled);
```

Description:

This function sets whether the character can change its current posture in pursuit of the gaze target.

Gaze can change current posture by default.

Arguments:

`enabled` 1 to enable, 0 to disable

Returns:

0 on success, -1 on failure

5.12.293 function

`diguyCharacter::get_gaze_can_change_posture`

Prototype:

```
int get_gaze_can_change_posture();
```

Returns:

the most recent setting of `set_gaze_can_change_posture()`

5.12.294 function diguyCharacter::set_gaze_can_change_variant**Prototype:**

```
int set_gaze_can_change_variant(int enabled);
```

Description:

This function sets whether the character can change its current variant in pursuit of the gaze target.

Gaze can change current variant by default.

Arguments:

`enabled` 1 to enable, 0 to disable

Returns:

0 on success, -1 on failure

5.12.295 function diguyCharacter::get_gaze_can_change_variant**Prototype:**

```
int get_gaze_can_change_variant();
```

Returns:

the most recent setting of `set_gaze_can_change_variant()`

5.12.296 function diguyCharacter::set_link_enabled_for_lpoint**Prototype:**

```
int set_link_enabled_for_lpoint(const char* link_name, int enabled);
```

Description:

This function sets whether the character can move the specified link in pursuit of the lpoint target.

Valid link names and defaults:

orientation	0
back	1
cervical	0
eye_l	0
shoulder_l	1
elbow_l	1

Arguments:

link_name	name of the link
enabled	1 to enable, 0 to disable

Returns:

0 on success, -1 on failure

5.12.297 function diguyCharacter::get_link_enabled_for_lpoint**Prototype:**

```
int get_link_enabled_for_lpoint(const char* link_name);
```

Returns:

the most recent setting of [set_link_enabled_for_lpoint\(\)](#)

5.12.298 function diguyCharacter::set_nod**Prototype:**

```
int set_nod(float azimuth,
            float elevation,
            float duration = 0.5f);
```

Description:

This function adjusts the current azimuth and elevation offsets of the character's head. If a gaze is ongoing, then the nod will be supplemented to the gazing, and the character will attempt to maintain its gaze.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

<code>azimuth</code>	rotation offset about z (vertical) axis, in degrees; positive rotates to character's left
<code>elevation</code>	rotation offset about y (horizontal) axis, in degrees; positive

	rotates down
duration	how long it should take to attain the new azimuth and elevation

Returns:

0 on success, -1 on failure

5.12.299 function diguyCharacter::nod_head

Prototype:

```
int nod_head(float duration,
             float nod_count,
             float nod_elevation0 = 15.0f,
             float nod_elevation1 = 0.0f,
             float nod_elevation_out = 0.0f);
```

Description:

This function causes the character to nod its head for the specified number of times over the specified duration.

If `nod_count` is 0.5, the elevation of the head will move from its current elevation straight to `nod_elevation_out`. The arguments `nod_elevation0` and `nod_elevation1` will be ignored.

If `nod_count` is 1.0, the elevation of the head will move from its current elevation to `nod_elevation0`, then to `nod_elevation_out`. The argument `nod_elevation1` will be ignored.

If `nod_count` is 1.5 or greater, the nod elevation will begin at its current elevation, move to `nod_elevation0`, then alternate between `nod_elevation0` and `nod_elevation1`, before finally ending at `nod_elevation_out`.

The head will maintain an elevation offset of `nod_elevation_out` until a new call to `nod_head()` or `set_nod()` is made.

This function overrides any nod settings made by the `set_nod()` function.

This function will have no effect on a dead character (see `die_now()` and `get_dead()`).

Arguments:

<code>duration</code>	for how long the nod should be executed
<code>nod_count</code>	how many nods the character should perform; this will be rounded to the nearest multiple of 0.5
<code>nod_elevation0</code>	elevation of initial nod in degrees
<code>nod_elevation1</code>	elevation of return nod in degrees
<code>nod_elevation_out</code>	ending elevation when nod is complete

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.300 function diguyCharacter::shake_head**Prototype:**

```
int shake_head(float duration,
float shake_count,
float shake_azimuth0 = -20.0f,
float shake_azimuth1 = 20.0f,
float shake_azimuth_out = 0.0f);
```

Description:

This function is similar to [nod_head\(\)](#), except the direction of head movement is side to side rather than up and down. See the [nod_head\(\)](#) function for a detailed description of usage.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

<code>duration</code>	for how long the shake should be executed
<code>shake_count</code>	how shakes the character should perform; this will be rounded to the nearest multiple of 0.5
<code>shake_azimuth0</code>	azimuth of initial shake in degrees
<code>shake_azimuth1</code>	azimuth of return shake in degrees
<code>shake_azimuth_out</code>	ending azimuth when shake is complete

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.301 function diguyCharacter::get_nod_azimuth**Prototype:**

```
float get_nod_azimuth();
```

Returns:

current nod azimuth setting; see [set_nod\(\)](#), [nod_head\(\)](#), and [shake_head\(\)](#)

5.12.302 function diguyCharacter::get_nod_elevation**Prototype:**

```
float get_nod_elevation();
```

Returns:

current nod elevation setting; see [set_nod\(\)](#), [nod_head\(\)](#), and [shake_head\(\)](#)

Aim Functions

The functions in this section generally only affect characters that have aim actions, such as soldiers or "suspects".

Aiming characters can use one of two aim algorithms. Each algorithm uses a different approach for moving the aim toward the desired aim azimuth and elevation.

Aim algorithm 8 was added in DI-Guy 8, it is the only option currently. It has a fair number of parameters that can be tweaked to achieve the desired balance of quick response and smooth movement.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.303 function diguyCharacter::set_aim

Prototype:

```
int set_aim(float azimuth, float elevation);
```

Description:

This function sets the current desired azimuth and elevation of the character's aim. The angles are in character-local coordinates. Zero values mean straight ahead and level. Aiming will continue until [end_aim\(\)](#) is called.

This function effectively calls [aim_at_angle_local\(\)](#), with a distance of 10000 meters, and `is_new_aim` set to 1.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

<code>azimuth</code>	rotation about z (vertical) axis, in degrees; positive rotates to character's left

elevation rotation about y (horizontal) axis, in degrees; positive rotates down

Returns:

0 on success, -1 on failure

5.12.304 function diguyCharacter::aim_at_angle

Prototype:

```
int aim_at_angle(float azimuth,
                 float elevation,
                 float distance,
                 int is_new_aim = 1);
```

Description:

This function sets the current azimuth, elevation, and distance of the character's aim. A fixed point in 3D space is then found using the two angles and the given distance from the character's idealized position. Angles are given with respect to the *Boston Dynamics* global coordinate system.

See [set_position\(\)](#) for a description of the coordinate system.

Aiming will continue until [end_aim\(\)](#) is called.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

<code>azimuth</code>	rotation in degrees about up (vertical) axis; positive rotates to the character's left; zero means in the direction of the positive X axis of the <i>Boston Dynamics</i> coordinate system
<code>elevation</code>	rotation in degrees about horizontal axis; positive rotates down
<code>distance</code>	distance in meters from the idealized position of the character to the aim target; positive values only, typically this is a large value (e.g. 10,000 meters)
<code>is_new_aim</code>	set to 1 if this is a new aim, for jumping to a new target; set to 0 if this is continuing an already started aim, for tracking an existing target

Returns:

0 on success, -1 on failure

5.12.305 function `diguyCharacter::aim_at_angle_local`

Prototype:

```
int aim_at_angle_local(float azimuth,
                      float elevation,
                      float distance,
                      int is_new_aim = 1);
```

Description:

This function is identical to `aim_at_angle( )` except that it uses body local coordinates for the character. The angles are given with respect to the idealized position of the character. Use `get_position( )` to learn the idealized position. The distance is between the idealized position of the character and the fixation point. Aiming will continue until `end_aim( )` is called.

This function will have no effect on a dead character (see `die_now( )` and `get_dead( )`).

5.12.306 function diguyCharacter::aim_at_point**Prototype:**

```
int aim_at_point(float x, float y, float z, int is_new_aim = 1);
```

Description:

This function sets the current fixation point of the character's aim. Point coordinates are given in meters in the *Boston Dynamics* global coordinate system.

The function uses default aim behavior settings. Aiming will continue until [end_aim\(\)](#) is called.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

tx, ty, tz	position in meters from the <i>Boston Dynamics</i> global origin
is_new_aim	set to 1 if this is a new aim; set to 0 if this is continuing an already started aim

Returns:

0 on success, -1 on failure

5.12.307 function diguyCharacter::aim_at_point_local**Prototype:**

```
int aim_at_point_local(float x, float y, float z, int is_new_aim = 1);
```

Description:

This function is identical to [aim_at_point\(\)](#) except that it uses body local coordinates for the character. Use [get_position\(\)](#) to learn the idealized position of the character if necessary.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

5.12.308 function diguyCharacter::aim_at_character**Prototype:**

```
int aim_at_character(const char* target_character_name,
    const char* target_character_link_name = NULL,
    float offset_tx = 0.0f,
    float offset_ty = 0.0f,
    float offset_tz = 0.0f);
```

Description:

This function effectively does an [aim_at_point\(\)](#) each tick. The point to be aimed at is updated each tick to be the position of the specified link on the specified character.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Callable From:

- ◇ C++
- ◇ Script

◇ Decision Bead

Arguments:

<code>target_character_name</code>	name of the character to be aimed at
<code>target_character_link_name</code>	link on the character that should be aimed at; pass NULL or "" to aim at the position link of the character
<code>offset_tx, offset_ty, offset_tz</code>	offset onto the link, in meters

Returns:

0 on success, -1 on failure

C++ Example:

```
// aim at the head of character "soldier-1".
ch->aim_at_character("soldier-1",
"cervical",
0.0,
0.0,
0.0);
```

5.12.309 function diguyCharacter::end_aim**Prototype:**

```
void end_aim();
```

Description:

This function terminates aiming started by the `set_aim()`, `aim_at_point()`, `aim_at_point_local()`, `aim_at_angle()`, `aim_at_angle_local()`, and `aim_at_character()` functions.

5.12.310 function diguyCharacter::aim_converge**Prototype:**

```
int aim_converge(int max_iterations = 30);
```

Description:

This function attempts to have the character reach its desired aim angles instantly. Because of the algorithm used, this can take a number of iterations to achieve.

The max_iterations argument sets a maximum limit this function will use to try to acquire the aim. Setting this value too high can reduce performance.

Arguments:

`max_iterations` maximum number of internal iterations to use to acquire aim

Returns:

1 if aim converged to within tolerances; 0 if not

5.12.311 function diguyCharacter::get_aim_azimuth**Prototype:**

```
float get_aim_azimuth();
```

Returns:

current aim azimuth; see [set_aim\(\)](#)

5.12.312 function diguyCharacter::get_aim_trajectory**Prototype:**

```
int get_aim_trajectory(float* muzzle_x, float* muzzle_y, float* muzzle_z,
    float* far_x, float* far_y, float* far_z);
```

Returns:

1 if the character is aiming 0 if not

Arguments:

muzzle_x,	position of muzzle in meters from the origin
muzzle_y,	
muzzle_z	
far_x,	position of end target in meters from the origin
far_y,	
far_z	

Pass NULL for any values that are not needed. Note: values will be zero if the character isn't currently aiming

5.12.313 function diguyCharacter::get_aim_elevation**Prototype:**

```
float get_aim_elevation();
```

Returns:

current aim elevation; see [set_aim\(\)](#)

5.12.314 function diguyCharacter::get_aim_is_possible**Prototype:**

```
int get_aim_is_possible();
```

Description:

Not all actions of a character will be affected by aim commands. This function returns 1 if this character is currently in an aim-able action.

Returns:

1 if current action is aim-able; 0 if not

5.12.315 function diguyCharacter::get_aim_is_acquired**Prototype:**

```
int get_aim_is_acquired();
```

Returns:

1 if the aim is acquired, 0 if not

5.12.316 function diguyCharacter::get_aim_is_steady**Prototype:**

```
int get_aim_is_steady();
```

Returns:

1 if all links participating in the aim have reached a relatively steady state, 0 if not, -1 on error

5.12.317 function diguyCharacter::set_aim_algorithm**Prototype:**

```
void set_aim_algorithm(int aim_algorithm);
```

Description:

Currently, 8 is the only valid input for this function, makes a DI-Guy 8 style aim.

5.12.318 function diguyCharacter::get_aim_algorithm**Prototype:**

```
int get_aim_algorithm();
```

Returns:

current aim algorithm defaults to 8

5.12.319 function diguyCharacter::get_is_aiming**Prototype:**

```
int get_is_aiming();
```

Returns:

1 if character is currently aiming, 0 if not

Aim Algorithm 8 Parameters

This aim algorithm works by adjusting the aiming character's pose such that the difference between the current aim angles and the desired aim angles is decreased each scenario tick.

How much of that difference that is made up each frame is determined by the factor parameters. The difference between the desired aim angles and the current aim angles, multiplied by a calculated factor, is added to the current aim angles each frame. This causes the aim to "home in" on the desired aim angles.

For example, if for each frame the calculated factor is 0.5, half of the difference between the

desired and current aim angles is made up.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.320 function diguyCharacter::set_aim_param_acquire_factor

Prototype:

```
void set_aim_param_acquire_factor(float factor);
```

Description:

This function sets the steady-state factor that is used once an aim is no longer considered "new".

See the function [set_aim_param_new_aim_duration\(\)](#) for information on when an aim is considered new, and how the effective acquire is calculated from this steady-state acquire factor and the new aim factor.

Setting the acquire factor to something very close to 1 (which would try to make up all of the difference in one frame) can result in an unstable, jittery convergence. The maximum recommended value is 0.95. If the aim is jittery, back off to 0.9 or lower.

To get "instant" convergence, call [aim_converge\(\)](#) or

`set_aim_converge_every_frame()`.

Arguments:

factor unitless value between 0 and 1; default is 0.6

5.12.321 function `diguyCharacter::get_aim_param_acquire_factor`

Prototype:

```
float get_aim_param_acquire_factor();
```

Returns:

the current aim acquire factor parameter

5.12.322 function diguyCharacter::set_aim_param_new_aim_factor

Prototype:

```
void set_aim_param_new_aim_factor(float factor);
```

Description:

This function sets the factor that is used when an aim is considered new.

See the function [set_aim_param_new_aim_duration\(\)](#) for information on when an aim is considered new, and how the effective acquire is calculated from this new aim factor and the steady-state acquire factor.

Arguments:

factor	unitless value between 0 and 1; default is 0.05

5.12.323 function diguyCharacter::get_aim_param_new_aim_factor

Prototype:

```
float get_aim_param_new_aim_factor();
```

Returns:

the current new aim factor parameter

5.12.324 function

diguyCharacter::set_aim_param_new_aim_duration

Prototype:

```
void set_aim_param_new_aim_duration(float duration);
```

Description:

This function sets how long an aim is considered "new".

When an aim is brand-new, the effective acquire factor is equal to the new aim factor as set by [set_aim_param_new_aim_factor\(\)](#). When the new aim duration has expired, the effective acquire factor is equal to the steady-state acquire factor as set by [set_aim_param_acquire_factor\(\)](#). In between, the effective acquire factor moves smoothly between the two.

For example, say that the new aim factor is 0.1, the acquire factor is 0.5, and the new aim duration is 0.2 seconds. If a new aim is started, say by a call to [aim_at_angle_local\(\)](#), the effective acquire factor of the first frame will be 0.1. Over the next 0.2 seconds, the effective acquire factor will smoothly ramp up to 0.5, where it will stay until the aim is acquired.

Arguments:

<code>duration</code>	how long, in seconds, a new aim should be considered new; default is 1 second

5.12.325 function**diguyCharacter::get_aim_param_new_aim_duration****Prototype:**

```
float get_aim_param_new_aim_duration();
```

Returns:

how long, in seconds, a new aim is considered new

5.12.326 function**diguyCharacter::set_aim_param_max_per_frame_adj_factor****Prototype:**

```
void set_aim_param_max_per_frame_adj_factor(float factor);
```

Description:

This function sets the maximum factor that can be used for any single frame for aim convergence attempts.

See [set_aim_param_acquire_factor\(\)](#) for a discussion of why setting this too close to 1 is not recommended.

Arguments:

<code>factor</code>	unitless value between 0 and 1; default is 0.95

5.12.327 function**diguyCharacter::get_aim_param_max_per_frame_adj_factor****Prototype:**

```
float get_aim_param_max_per_frame_adj_factor();
```

Returns:

the maximum per-frame adjustment factor parameter

5.12.328 function**diguyCharacter::set_aim_param_azimuth_acquired_range****Prototype:**

```
void set_aim_param_azimuth_acquired_range(float range);
```

Description:

This function sets how close the current aim azimuth must be to the desired aim azimuth in order for the aim to be considered acquired.

Note that the aim isn't fully acquired until both azimuth and elevation are acquired.

Setting this value too close to 0 will result in the aim never being acquired.

Arguments:

range	how close, in degrees, the azimuth must be for the aim to be acquired; default is 0.5 degrees

5.12.329 function**diguyCharacter::get_aim_param_azimuth_acquired_range****Prototype:**

```
float get_aim_param_azimuth_acquired_range();
```

Returns:

acquired

5.12.330 function**diguyCharacter::set_aim_param_elevation_acquired_range****Prototype:**

```
void set_aim_param_elevation_acquired_range(float range);
```

Description:

This function sets how close the current aim elevation must be to the desired aim elevation in order for the aim to be considered acquired.

Note that the aim isn't fully acquired until both azimuth and elevation are acquired.

Setting this value too close to 0 will result in the aim never being acquired.

Arguments:

range	how close, in degrees, the elevation must be for the aim to be acquired; default is 1.0 degrees

5.12.331 function**diguyCharacter::get_aim_param_elevation_acquired_range****Prototype:**

```
float get_aim_param_elevation_acquired_range();
```

Returns:

acquired

5.12.332 function**diguyCharacter::set_aim_converge_every_frame****Prototype:**

```
void set_aim_converge_every_frame(int max_iterations = 30);
```

Description:

This function attempts to have the character reach its desired aim angles instantly, every frame. Because of the algorithm used, this can take a number of iterations to achieve.

The `max_iterations` argument sets a maximum limit this function will use to try to acquire the aim. Setting this value too high can reduce performance.

Set `max_iterations` to 0 to disable instant aim convergence.

For a one-time instant convergence, call [aim_converge\(\)](#).

Arguments:

`max_iterations` maximum number of internal iterations to use to acquire aim

5.12.333 function `diguyCharacter::get_aim_converge_every_frame`

Prototype:

```
int get_aim_converge_every_frame();
```

Returns:

how many iterations will be used each frame for attempt at per-frame instant convergence; 0 if instant convergence is disabled

Weapon Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.334 function diguyCharacter::fire_weapon

Prototype:

```
int fire_weapon(diguyMuzzle muzzle = DIGUY_MUZZLE_DEFAULT);
```

Description:

This function causes a character to fire his weapon.

Different weapons have different "munition types", which specify, among other things, the sound to be played, the network info to be sent, and muzzle flash effect. See [get_weapon_muzzle_munition_type\(\)](#) for more information on munition types.

If the weapon fires live rounds (as set by [set_weapon_fires_live_rounds\(\)](#)) this function also computes a bullet trajectory, perhaps hitting and killing another character.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.335 function diguyCharacter::fire_weapon_n_times

Prototype:

```
int fire_weapon_n_times(int count,
    float dt = 0.2f,
    const char* override_sound_name = NULL,
    diguyMuzzle muzzle = DIGUY_MUZZLE_DEFAULT);
```

Description:

This function causes a character to fire his weapon the specified number of times. This causes a muzzle flash and the weapon's sound (if any) to be played.

The sound played will be one of, in order:

- ◇ the passed `override_sound_name`, or
- ◇ the sound set by `set_weapon_sound_override()`, or
- ◇ the default weapon sound based on the character's appearance

If the weapon fires live rounds (as set by `set_weapon_fires_live_rounds()`), this function also computes a

trajectory for each round, perhaps hitting and killing another character.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

count	number of times weapon should be fired
override_sound_name	override sound to be played
dt	time between firings if count > 1
muzzle	muzzle on which flash should appear

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.336 function diguyCharacter::fire_screen_space_bullet

Prototype:

```
int fire_screen_space_bullet(diguyView* view, float x, float y);
```

Description:

Fires a bullet at point (x, y) from the camera, using this character as the attacker.

Currently OpenGL only.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Returns:

number of hits

5.12.337 function diguyCharacter::get_weapon_fire_count**Prototype:**

```
int get_weapon_fire_count(diguyMuzzle muzzle = DIGUY_MUZZLE_DEFAULT);
```

Returns:

number of times weapon has been fired from specified muzzle

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.338 function diguyCharacter::set_default_weapon_muzzle**Prototype:**

```
int set_default_weapon_muzzle(diguyMuzzle muzzle);
```

Description:

This function sets the default muzzle that will flash when the character fires its weapon due to a call to [fire_weapon\(\)](#) or [fire_weapon_n_times\(\)](#).

The default value is [DIGUY_MUZZLE_DEFAULT](#).

Arguments:

`muzzle` the muzzle that will flash; muzzle 1 is common

Returns:

0 on success, -1 on failure

5.12.339 function diguyCharacter::get_default_weapon_muzzle**Prototype:**

```
diguyMuzzle get_default_weapon_muzzle();
```

Returns:

current default weapon muzzle that flashes when the character's weapon is fired; see [set_default_weapon_muzzle\(\)](#)

5.12.340 function diguyCharacter::get_weapon_muzzle_munition_type**Prototype:**

```
const char* get_weapon_muzzle_munition_type(diguyMuzzle muzzle);
```

Description:

When the character fires its weapon, the results depend on the munition type of the muzzle fired from. Munition types specify, among other things:

- ◇ sound played
- ◇ muzzle flash effect
- ◇ network data sent (DIS septet)
- ◇ detonation effect
- ◇ detonation radius
- ◇ etc.

Default munition types for the each muzzle are set automatically when the character's appearance is set. The munition type of the weapon can be overridden by calling [set_weapon_munition_type_override\(\)](#).

Current munition types include:

- ◇ m16
- ◇ m4
- ◇ m60
- ◇ m240
- ◇ m249
- ◇ m9
- ◇ hand_gun
- ◇ 50cal
- ◇ m256
- ◇ ak47
- ◇ pk74
- ◇ dragunov
- ◇ sa80
- ◇ rpg
- ◇ m203
- ◇ 40mm_hedp
- ◇ javelin
- ◇ stinger
- ◇ at4
- ◇ sa7
- ◇ molotov_cocktail
- ◇ 60mm_mortar
- ◇ 82mm_mortar
- ◇ 155mm_artillery
- ◇ IED_mortar
- ◇ VBIED
- ◇ m242
- ◇ 120mm
- ◇ 125mm
- ◇ ced
- ◇ pepper_spray

Returns:

munition type of specified muzzle

5.12.341 function

diguyCharacter::get_default_muzzle_munition_type

Prototype:

```
const char* get_default_muzzle_munition_type();
```

Returns:

munition type of default muzzle, as set by
[set_default_weapon_muzzle\(\)](#)

5.12.342 function

diguyCharacter::set_weapon_munition_type_override

Prototype:

```
int set_weapon_munition_type_override(const char* munition_type);
```

Description:

This function allows the default munition type to be overridden. The default value is NULL (don't override default munition type).

Arguments:

`munition_type` name of a munition type

Returns:

0 on success, -1 on failure

5.12.343 function diguyCharacter::get_weapon_munition_type_override

Prototype:

```
const char* get_weapon_munition_type_override();
```

Returns:

current munition type override in effect, or NULL if none;
[set_weapon_munition_type_override\(\)](#)

5.12.344 function diguyCharacter::set_weapon_fires_live_rounds

Prototype:

```
int set_weapon_fires_live_rounds(int weapon_fires_live_rounds);
```

Description:

This function sets whether the character's weapons fire live rounds. If so, when [fire_weapon\(\)](#) and [fire_weapon_n_times\(\)](#) are called the trajectory of the bullet will be calculated and checked against characters in the scenario.

The default value is 1.

Arguments:

`weapon_fires_live_rounds` set to 1 for live rounds; set to 0 for "blanks"

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.345 function `diguyCharacter::get_weapon_fires_live_rounds`

Prototype:

```
int get_weapon_fires_live_rounds();
```

Returns:

current setting of whether weapon fires live rounds or blanks; see
[`set\_weapon\_fires\_live\_rounds\(\)`](#)

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.346 function diguyCharacter::set_weapon_sound_override**Prototype:**

```
int set_weapon_sound_override(const char* sound_name);
```

Description:

A weapon sound is played when the character fires its weapon due to a call to `fire_weapon()` or `fire_weapon_n_times()`. The sound that is played is generally derived from the weapon's munition type, which can be different for each muzzle.

This function overrides the default sound that will be played. Note that this overrides the sound for all muzzles.

The default value is NULL (don't override default sound).

To **not** play a sound at all, set the weapon sound gain override to 0.

Arguments:

sound	name of a sound defined in the scenario; pass NULL to specify that default should be played

Returns:

0 on success, -1 on failure

5.12.347 function diguyCharacter::get_weapon_sound_override**Prototype:**

```
const char* get_weapon_sound_override();
```

Description:

The function returns the current weapon sound override, if one is set; see [set_weapon_sound_override\(\)](#).

This function will not return NULL. The empty string ("") is returned if a weapon sound override is not set.

Returns:

name of weapon sound override

5.12.348 function diguyCharacter::set_weapon_sound_gain_override**Prototype:**

```
int set_weapon_sound_gain_override(float gain);
```

Description:

This function overrides the default gain on the sound played due to a call to [fire_weapon\(\)](#) or [fire_weapon_n_times\(\)](#).

Pass 1.0 to use the sound's default amplitude. Lower numbers decrease volume, higher numbers increase volume.

Pass 0.0 to completely disable playing of weapon sound.

The default value is 3.0.

See [set_weapon_sound_override\(\)](#).

Arguments:

`sound_gain` amount by which to change sound volume

Returns:

0 on success, -1 on failure

5.12.349 function `diguyCharacter::get_weapon_sound_gain_override`

Prototype:

```
float get_weapon_sound_gain_override();
```

Returns:

current override gain for weapon sound; see
[`set\_weapon\_sound\_gain\_override\(\)`](#)

5.12.350 function diguyCharacter::set_weapon_spread**Prototype:**

```
void set_weapon_spread(float spread);
```

Description:

This function provides aiming assistance by making it possible for roughly-aimed weapons to hit targets. When spread is nonzero, fire_weapon searches for a target within a conical space centered on the muzzle. Within this cone, the live character within weapon range, and with the smallest angular deviation from the weapon, is chosen, by default, as the target. Other function calls may alter this aiming behavior, making it aim only at enemy targets, for example, or making it prefer close targets to far ones.

Arguments:

spread	angular deviation from weapon muzzle within which a target will be found

Spread values above 145 are suicidal (character can shoot himself). 90 is the highest sensible value, which covers everything ahead of the weapon muzzle. A value of 0 is the default, meaning that only targets in line with the muzzle are hit.

5.12.351 function diguyCharacter::set_weapon_near_range**Prototype:**

```
void set_weapon_near_range(float near_range);
```

Description:

This function affects how targets are chosen when the character's `weapon_spread` is nonzero. Only characters more than `near_range` meters away from the muzzle will be fired upon. Characters which are closer than `near_range` meters will not be fired upon.

This value does not affect weapon behavior when `weapon_spread` is zero.

If this function is never called for a character, it is as if the function had been called with a value of zero.

Arguments:

`near_range` distance closer than which no character will be targeted

5.12.352 function diguyCharacter::set_weapon_far_range**Prototype:**

```
void set_weapon_far_range(float far_range);
```

Description:

This function sets the range of the character's weapon.

If this function is never called for a character, it is as if the function had been called with a value of 500.0.

Arguments:

`far_range` range of the weapon in meters

5.12.353 function diguyCharacter::set_weapon_hit_ratio**Prototype:**

```
void set_weapon_hit_ratio(float hit_ratio);
```

Description:

This function sets a weapon's ratio of hits to fired rounds. A value of 1.0 (the default) means the weapon will hit everything it is aimed at. Values less than 1.0 cause the weapon to probabilistically miss targets, even when perfectly aimed at them. A value of zero means the weapon doesn't hit anything, even if it is perfectly aimed. A value of 0.5 will hit half of the time.

5.12.354 function diguyCharacter::set_weapon_max_impacts**Prototype:**

```
void set_weapon_max_impacts(int val);
```

Description:

This function sets the number of potential impacts a fired round can have. The default is 1. Specifying more than 1 means the round will completely go through objects, triggering a CALLBACK_ID_IMPACT callback each time, until the maximum number of impacts has been reached.

5.12.355 function diguyCharacter::get_accessory1_state**Prototype:**

```
int get_accessory1_state();
```

Returns:

state of accessory1

5.12.356 function diguyCharacter::get_rifle_state**Prototype:**

```
int get_rifle_state();
```

Returns:

state of the character's rifle

5.12.357 function

diguyCharacter::set_weapon_projectile_is_visible

Prototype:

```
int set_weapon_projectile_is_visible(diguyMuzzle muzzle, int is_visible);
```

Description

This function sets whether the projectiles of certain appearances' weapons (such as rocket launchers and RPGs) are visible. Note that most weapons do not have visible projectiles and this function will therefore have no effect.

Arguments:

`muzzle` which muzzle's projectile should be shown/hidden
`is_visible` pass 1 to show, pass 0 to hide

Returns:

0 on success, -1 on failure

5.12.358 function

diguyCharacter::get_weapon_projectile_is_visible

Prototype:

```
int get_weapon_projectile_is_visible(diguyMuzzle muzzle);
```

Description

This function returns the most recent setting of `set_weapon_projectile_is_visible()`.

Arguments:

`muzzle` which muzzle's projectile should be queried

Returns:

1 if visible, 0 if not

5.12.359 function diguyCharacter::get_weapon_aim_vector**Prototype:**

```
int get_weapon_aim_vector(float* muzzle_tx, float* muzzle_ty, float* muzzle_tz,
    float* end_pt_tx = NULL, float* end_pt_ty = NULL, float* end_pt_tz = NULL,
    float* muzzle_rz = NULL, float* muzzle_rx = NULL, float* muzzle_ry = NULL,
    int in_character_local_coordinates = 0,
    diguyMuzzle muzzle = DIGUY_MUZZLE_DEFAULT);
```

Description:

Retrieves the position and direction of the character's weapon.

Arguments:

muzzle_tx, muzzle_ty, muzzle_tz	muzzle position, and beginning of aim vector
end_pt_tx, end_pt_ty, end_pt_tz	end point of aim vector
muzzle_rz, muzzle_rx, muzzle_ry	orientation of muzzle in degrees
muzzle	the muzzle for which the aim vector is needed
in_character_local_coordinates	pass 0 for values in world coordinates (the default), 1 for values in character-local coordinates

The muzzle position and end point are in meters from the origin (if using world coordinates) or in meters from the character's position (if using character-local coordinates).

The distance of the end point from the muzzle position should be the weapon's far range, set by [set_weapon_far_range\(\)](#).

Pass NULL for any values that are not needed.

Note: The muzzle argument is currently not used, and is present for future implementation.

Returns:

0 on success, -1 on failure

5.12.360 function diguyCharacter::get_muzzle_link**Prototype:**

```
const char* get_muzzle_link(diguyMuzzle muzzle = DIGUY_MUZZLE_DEFAULT);
```

Description

This function returns the link the passed muzzle is associated with. For example, soldier rifle muzzles are attached to the "rifle_joint" link.

Call [get_muzzle_offset\(\)](#) to get the offset of the muzzle on this link.

Arguments:

muzzle	muzzle of interest

Returns:

name of link, or NULL if the character doesn't have the specified muzzle

5.12.361 function diguyCharacter::get_muzzle_offset**Prototype:**

```
int get_muzzle_offset(diguyMuzzle muzzle = DIGUY_MUZZLE_DEFAULT,
float* offset_x = NULL,
float* offset_y = NULL,
float* offset_z = NULL);
```

Description

This function returns the offset of the muzzle from the link it is attached to as returned by [get_muzzle_link\(\)](#).

Arguments:

muzzle	muzzle of interest
offset_x, offset_y, offset_z	variables in which to return results

Returns:

0 on success, -1 on failure

5.12.362 function `diguyCharacter::set_weapon_max_azimuth_error`

Prototype:

```
int set_weapon_max_azimuth_error(float h_error);
```

Description

This function sets the highest horizontal error possible when the weapon is fired. If non-0, a random horizontal error angle will be computed and added to the weapon's horizontal aim angle.

Arguments:

<code>h_error</code>	highest possible horizontal error

Returns:

0 on success, -1 on failure

5.12.363 function

diguyCharacter::get_weapon_max_azimuth_error

Prototype:

```
float get_weapon_max_azimuth_error();
```

Returns:

maximum horizontal aim error

5.12.364 function

diguyCharacter::set_weapon_max_elevation_error

Prototype:

```
int set_weapon_max_elevation_error(float v_error);
```

Description

Similar to [set_weapon_max_azimuth_error \(\)](#), but for vertical aim.

5.12.365 function diguyCharacter::get_weapon_max_elevation_error

Prototype:

```
float get_weapon_max_elevation_error();
```

Description

Similar to [get_weapon_max_azimuth_error\(\)](#), but for vertical aim.

5.12.366 function diguyCharacter::set_weapon_tracers_enabled

Prototype:

```
int set_weapon_tracers_enabled(int enabled);
```

Description

This function sets whether tracer effects will be shown when the weapon is fired. Not all weapons will support this effect.

Arguments:

`enabled` pass 1 to enable, 0 to disable

Returns:

0 on success, -1 on failure

5.12.367 function diguyCharacter::get_weapon_tracers_enabled**Prototype:**

```
int get_weapon_tracers_enabled();
```

Returns:

1 if tracer effect is enabled, 0 if not

**5.12.368 function
diguyCharacter::get_weapon_rounds_until_tracer****Prototype:**

```
int get_weapon_rounds_until_tracer(diguyMuzzle muzzle);
```

Returns:

number of rounds until next tracer

5.12.369 function `diguyCharacter::set_weapon_shell_eject_enabled`

Prototype:

```
int set_weapon_shell_eject_enabled(int enabled);
```

Description

This function sets whether a shell eject effect will be shown when the weapon is fired. Not all weapons will support this effect.

Arguments:

`enabled` pass 1 to enable, 0 to disable

Returns:

0 on success, -1 on failure

5.12.370 function `diguyCharacter::get_weapon_shell_eject_enabled`

Prototype:

```
int get_weapon_shell_eject_enabled();
```

Returns:

1 if shell eject effect is enabled, 0 if not

5.12.371 function diguyCharacter::set_weapon_smoke_enabled**Prototype:**

```
int set_weapon_smoke_enabled(int enabled);
```

Description

This function sets whether a smoke effect will be shown when the weapon is fired. Not all weapons will support this effect.

Arguments:

`enabled` pass 1 to enable, 0 to disable

Returns:

0 on success, -1 on failure

5.12.372 function diguyCharacter::get_weapon_smoke_enabled**Prototype:**

```
int get_weapon_smoke_enabled();
```

Returns:

1 if muzzle smoke effect is enabled, 0 if not

5.12.373 function**diguyCharacter::trigger_detonation_on_character****Prototype:**

```
int trigger_detonation_on_character(const char* munition_name,
    const char* attacker_name = NULL,
    float radius_override = -1.0f,
    int broadcast_on_network = 1);
```

Description:

Similar to [diguyScenario::trigger_detonation\(\)](#), but detonation occurs at character's position.

Arguments:

<code>munition_name</code>	name of the munition to detonate
<code>attacker_name</code>	this MUST be specified if the detonation is being broadcast over the network
<code>radius_override</code>	defaults to the data in the munition config file
<code>broadcast_on_network</code>	pass 0 to not broadcast detonation on DIS network

Note: Overriding the detonation radius will NOT work on broadcast detonations since the override value is not transmitted.

Face Functions

The functions in this section are available only if `diguy_facefx_initialize()` has been called, and if the character's appearance or head appearance is a "facefx" appearance.

All functions in this section are callable from both C++ and Script.

5.12.374 function `diguyCharacter::set_face_expression`

Prototype:

```
int set_face_expression(const char* facex_name, float blend_duration, float strength)
```

Description:

This function changes the face of the character to reflect the specified user-defined named facial expression.

Currently affects only "facefx" appearances.

Arguments:

<code>facex_name</code>	name of face expression to be applied
<code>blend_duration</code>	how long (in seconds) to take to change the face
<code>strength</code>	a multiplier between 0 and 1 on the expression

Face expressions available by default in new scenarios are:

- ◇ Neutral
- ◇ Smile_cl
- ◇ Trust
- ◇ Sad
- ◇ Mad
- ◇ Distrust
- ◇ Conniving
- ◇ Smile
- ◇ BrowsDown
- ◇ BrowsUp

New face expressions can be added using the `diguyScenario::create_face_expression()` function.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

C++ Example:

```
diguyCharacter* ch = scenario->get_character_at_index(0);  
  
// take one second to get mad  
ch->set_face_expression("Mad", 1.0);
```

5.12.375 function diguyCharacter::set_orientation_eyes**Prototype:**

```
int set_orientation_eyes(float azimuth,
                        float elevation,
                        float blend_time = 0.0f);
```

Description:

This function sets the current azimuth and elevation of the character's eyes.

Currently affects only "facefx" appearances.

Arguments:

<code>azimuth</code>	rotation about up (vertical) axis, in degrees; positive rotates to character's left
<code>elevation</code>	rotation about forward (horizontal) axis, in degrees; positive rotates down
<code>blend_time</code>	how long to take to reach new orientation

Returns:

0 on success, -1 on failure

5.12.376 function diguyCharacter::get_orientation_eyes**Prototype:**

```
int get_orientation_eyes(float* azimuth, float* elevation);
```

Description:

This function gets the current azimuth and elevation of the character's eyes. It currently doesn't take blending into account.

Currently affects only "facefx" appearances.

5.12.377 function diguyCharacter::set_eye_droop**Prototype:**

```
int set_eye_droop(float droop_amount, float blend_time = 0.0f);
```

Description:

Sets how open the eyes are. The eyes will stay at this level indefinitely. To get automatic blinking behavior, call [blink\(\)](#).

Arguments:

droop_amount 1 = eyes closed, 0 = eyes open, 0.5 = half open

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

- ◇ Script
- ◇ Decision Bead

5.12.378 function diguyCharacter::get_eye_droop

Prototype:

```
float get_eye_droop();
```

Returns:

how open the eyes are as set by [set_eye_droop\(\)](#).

5.12.379 function diguyCharacter::blink**Prototype:**

```
int blink(float duration_in = 0.1f,
          float duration_out = 0.2f,
          float target_in = 1.0f,
          float target_out = 0.0f,
          float tin = DIGUY_DEFAULT_FLOAT);
```

Description:

This function blinks the eyes. To have the eyes close to a specific amount and stay there, call [set_eye_droop\(\)](#).

Only characters with expressive face appearances can blink.

Arguments:

<code>duration_in</code>	how long in seconds the first part of the blink (typically eyes closing) should take
<code>duration_out</code>	how long in seconds the second part of the blink (typically eyes opening) should take
<code>target_in</code>	how closed the eyes should be at <code>duration_in</code>
<code>target_out</code>	how open the eyes should be at <code>duration_out</code> and thereafter
<code>tin</code>	time at which blink should occur; default is now

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.380 function diguyCharacter::set_automatic_blinks_enabled**Prototype:**

```
int set_automatic_blinks_enabled(int enabled,
    float period = 6.0f,
    float period_variation = 3.0f);
```

Description:

This function enables automatic eye blinking, so that blinks will happen without further function calls. The various controlling factors of blinks are set by the following functions:

- ◇ [set_automatic_blink_duration_in\(\)](#)
- ◇ [set_automatic_blink_duration_out\(\)](#)
- ◇ [set_automatic_blink_target_in\(\)](#)
- ◇ [set_automatic_blink_target_out\(\)](#)

See [blink\(\)](#) for information on what these factors mean.

Only characters with expressive face appearances can blink.

Arguments:

<code>enabled</code>	pass 1 to enable automatic blinks; 0 to disable
<code>period</code>	how many seconds between each blink
<code>period_variation</code>	variation on how many seconds between each blink

If blinks are enabled, the first blink will happen immediately. Successive blinks will happen between times $(\text{period} - \text{period\_variation})$ and $(\text{period} + \text{period\_variation})$ after the previous blink ends.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.381 function diguyCharacter::get_automatic_blinks_enabled**Prototype:**

```
int get_automatic_blinks_enabled();
```

Returns:

1 if automatic eye blinks are enabled, 0 if not; see [set_automatic_blinks_enabled\(\)](#)

5.12.382 function diguyCharacter::set_automatic_blink_duration_in**Prototype:**

```
void set_automatic_blink_duration_in(float duration_in);
```

Description:

This function sets how long it will take the eye to close during automatic eye blinks. See [set_automatic_blinks_enabled\(\)](#) and [blink\(\)](#).

Arguments:

duration_in time in seconds for eye droop to go from droop target in to target out

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.383 function diguyCharacter::get_automatic_blink_duration_in

Prototype:

```
float get_automatic_blink_duration_in();
```

Returns:

most recent setting of [set_automatic_blink_duration_in\(\)](#)

5.12.384 function diguyCharacter::set_automatic_blink_duration_out

Prototype:

```
void set_automatic_blink_duration_out(float duration_out);
```

Description:

This function sets how long it will take the eye to open during automatic eye blinks. See [set_automatic_blinks_enabled\(\)](#) and [blink\(\)](#).

Arguments:

duration_out time in seconds for eye droop to go from droop target out to target in

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.385 function diguyCharacter::get_automatic_blink_duration_out

Prototype:

```
float get_automatic_blink_duration_out();
```

Returns:

most recent setting of [set_automatic_blink_duration_out\(\)](#)

5.12.386 function diguyCharacter::set_automatic_blink_target_in

Prototype:

```
void set_automatic_blink_target_in(float target_in);
```

Description:

This function sets how much the eye will close during automatic eye blinks. See [set_automatic_blinks_enabled\(\)](#) and [blink\(\)](#).

Arguments:

`target_in` how far to close eye; 1.0 is all closed, 0.0 is all open

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.387 function diguyCharacter::get_automatic_blink_target_in**Prototype:**

```
float get_automatic_blink_target_in();
```

Returns:

most recent setting of `set_automatic_blink_target_in()`

5.12.388 function diguyCharacter::set_automatic_blink_target_out**Prototype:**

```
void set_automatic_blink_target_out(float target_out);
```

Description:

This function sets how much the eye will open during automatic eye blinks. See `set_automatic_blinks_enabled()` and `blink()`.

Arguments:

`target_out` how far to open eye; 1.0 is all closed, 0.0 is all open

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.389 function diguyCharacter::get_automatic_blink_target_out

Prototype:

```
float get_automatic_blink_target_out();
```

Returns:

most recent setting of `set_automatic_blink_target_out()`

5.12.390 function diguyCharacter::is_speaking

Prototype:

```
int is_speaking();
```

Returns:

1 if the character is speaking, 0 if not. With a FaceFX character this is true if a character is actively playing a FaceFX animation

5.12.391 function

`diguyCharacter::set_current_facefx_animation_group`

Prototype:

```
int set_current_facefx_animation_group(const char* anim_group);
```

Description:

FaceFX animations are referred to as a group/animation name. This function allows you to set the current group once. See [diguyScenario::facefx_mount_animset\(\)](#) for how to load a new animset on an actor.

5.12.392 function

`diguyCharacter::get_current_facefx_animation_group`

Prototype:

```
const char* get_current_facefx_animation_group();
```

Returns:

the current FaceFX animation group set for this character

5.12.393 function diguyCharacter::get_facefx_actor**Prototype:**

```
const char* get_facefx_actor();
```

Returns:

the name of the FaceFX actor that goes with the current head appearance;
returns NULL if there's no FaceFX actor

5.12.394 function diguyCharacter::facefx_mount_animset**Prototype:**

```
int facefx_mount_animset(const char* file_name);
```

Description:

Mounts an animset on the current FaceFX actor. See also
[diguyScenario::facefx_mount_animset\(\)](#)

5.12.395 function diguyCharacter::play_facefx_animation**Prototype:**

```
int play_facefx_animation(const char* animation,
    float magnitude_scale = 1.0f,
    float duration_scale = 1.0f,
    float blendin_time = 0.1f,
    float blendout_time = 0.1f,
    int loop = 0);
```

Description:

Plays a FaceFX animation. This is the main way a facial animation sequence is triggered. The animation can be named "group_name/animation" or just "animation" if the proper group has been set with [set_current_facefx_animation_group\(\)](#).

5.12.396 function diguyCharacter::stop_facefx_animations**Prototype:**

```
int stop_facefx_animations(float blend_time = 0.25f, int stop_sounds = 1);
```

Description:

Stops all active FaceFX animation.

5.12.397 function diguyCharacter::get_num_facefx_nodes**Prototype:**

```
int get_num_facefx_nodes();
```

Returns:

the number of nodes in the FaceFX actor's face graph

5.12.398 function diguyCharacter::get_facefx_node_name**Prototype:**

```
const char* get_facefx_node_name(int index);
```

Returns:

the name of the node with index 'index' in the FaceFX actor's face graph

5.12.399 function diguyCharacter::find_facefx_node_index**Prototype:**

```
int find_facefx_node_index(const char* node_name);
```

Returns:

the index of the node with name 'node_name'

5.12.400 function diguyCharacter::set_facefx_node_value**Prototype:**

```
int set_facefx_node_value(int node_index,  
    float value,  
    float blend_time = 0.25f);
```

Description:

Allows low level control of individual nodes of the characters face graph. End users can use this function to get finer grained control over eye direction and emotional state.

5.12.401 function diguyCharacter::get_facefx_node_value**Prototype:**

```
float get_facefx_node_value(int node_index);
```

Description:

Retreives the current final value of a node

Gesture Functions

A gesture is a motion a character can do that temporarily overrides a subset of the character's joints. Gestures are designed to supplement the stock set of character motions. Most gestures are available to all characters.

Gesture Stages

Many gestures proceed in stages. Before stage 2 of the gesture is played, stage 1 must be played. Sometimes a stage is repeated multiple times before the gesture moves on to the next stage.

An example of a three stage gesture could be a wave. In stage 1, the gesture raises the character's arm. In stage 2, the arm waves back and forth once. Stage 2 may be repeated multiple times to get the character to wave more than once. Stage 3 lowers the arm back to the character's side.

Note that stages are numbered 1 to 3, not 0 to 2.

Gesture Channels

Some gestures have more than one channel. Each channel in a multi-channel gesture contains a variation of the gesture. As the gesture progresses, the character may move from one channel to another.

An example of a two channel gesture could be a chop beat gesture for emphasizing speech. Channel A is a small, tentative chop. Channel B is a strong, emphatic chop. If the weights of channels A and B are 1 and 0, respectively, the gesture will be the small chop. If the weights are 0 and 1, respectively, the gesture will be the emphatic chop. If the weights are 0.5 and 0.5, respectively, the gesture will be a combination -- partly small, partly emphatic.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.402 function diguyCharacter::create_gesture

Prototype:

```
diguyCharacterGesture* create_gesture(const char* gesture_name);
```

Description:

This function creates (but does not begin) the specified gesture. Exact control over the gesture, such as number of repetitions, when the gesture should begin, when it should end, when it should reach specific points within the gestures, etc., is then possible by making calls through the returned [diguyCharacterGesture](#) object.

To begin execution of the gesture, call the `begin_now( )` function of the [diguyCharacterGesture](#) object.

Do not call `delete` on the returned object. It should be destroyed either by calling [destroy_gesture\(\)](#), or by calling `set_automatic_destroy_flag(1)` on the [diguyCharacterGesture](#) object.

Arguments:

`gesture_name` name of the gesture of create

Returns:

pointer to object of type `diguyCharacterGesture`

5.12.403 function `diguyCharacter::destroy_gesture`

Prototype:

```
int destroy_gesture(diguyCharacterGesture* gesture);
```

Description:

This function destroys a gesture object created by `create_gesture()`.

Arguments:

`gesture` gesture to destroy

Returns:

0 on success, -1 on failure

5.12.404 function diguyCharacter::execute_gesture**Prototype:**

```
int execute_gesture(const char* gesture_name,
    int reps = DIGUY_DEFAULT_INT,
    float overall_duration = DIGUY_DEFAULT_FLOAT,
    float channel_A_weight = DIGUY_DEFAULT_FLOAT);
```

Description:

This function executes the specified gesture.

The gesture will automatically begin and end. The only control over the executed gesture is whether it should be ended or aborted early. See [end_executing_gesture\(\)](#) and [abort_executing_gesture\(\)](#). For full control of gesture attributes, use [create_gesture\(\)](#) instead.

A gesture takes over control of a subset of a character's pose variables and can be overlaid on top of most base actions of the character.

Note that gestures should be preloaded at the beginning of a scenario to avoid a hitch in frame rate. See the function [diguyScenario::preload_gesture\(\)](#).

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

<code>gesture_name</code>	name of the gesture to execute
<code>reps</code>	how many times the gesture should be repeated; default is 1 if no duration is specified, or best fit if there is a duration specified
<code>overall_duration</code>	for how long the gesture should be executed; default is such that no time scaling occurs
<code>channel_A_weight</code>	the weight given to channel A of the gesture; ignored if the gesture does not have multiple channels

If the gesture is a multi-stage gesture, the overall duration will be spread proportionately over each stage.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.405 function diguyCharacter::execute_1stage_gesture

Prototype:

```
int execute_1stage_gesture(const char* gesture_name,
    int reps = DIGUY_DEFAULT_INT,
    float stage1_duration = DIGUY_DEFAULT_FLOAT,
    float channel_A_weight = DIGUY_DEFAULT_FLOAT);
```

Description:

The gesture will automatically begin and end. The only control over the executed gesture is whether it should be ended or aborted early. See [end_executing_gesture\(\)](#) and [abort_executing_gesture\(\)](#). For full control of gesture attributes, use [create_gesture\(\)](#) instead.

If the specified gesture is not a 1 stage gesture, no gesture is executed and an error value is returned.

The specified gesture can be a one or two channel 1 stage gesture, in which case the specified channel weight will be used over the course of the entire gesture.

A 1 stage gesture is a comprised of a single motion that begins and ends the gesture.

n repetitions of a 1 stage gesture will repeat the stage 1 motion n times.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

<code>gesture_name</code>	name of the gesture of execute
<code>reps</code>	how many times the gesture should be repeated; default is 1 if no duration is specified, or best fit if there is a duration specified
<code>stage1_duration</code>	how long stage 1 of the gesture should take; default is such that no time scaling occurs
<code>channel_A_weight</code>	the weight given to channel A of the gesture; ignored if the gesture does not have multiple channels

Returns:

0 on success, -1 on failure

5.12.406 function `diguyCharacter::execute_2stage_gesture`

Prototype:

```
int execute_2stage_gesture(const char* gesture_name,
    int reps = DIGUY_DEFAULT_INT,
    float stage1_duration = DIGUY_DEFAULT_FLOAT,
    float stage2_duration = DIGUY_DEFAULT_FLOAT,
    float channel_A_weight = DIGUY_DEFAULT_FLOAT);
```

Description:

This function is similar to [execute_1stage_gesture\(\)](#).

Stage 1 of a 2 stage gesture is a motion that begins the gesture and leads to the "peak" of the gesture.

Stage 2 of a 2 stage gesture is the motion that returns the gesture from the "peak" to the starting position.

n repetitions of a 2 stage gesture will repeat the stage 1 motion followed by the stage 2 motion n times.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

<code>gesture_name</code>	name of the gesture of execute
<code>reps</code>	how many times the gesture should be repeated; default is 1 if no duration is specified, or best fit if there is a duration specified
<code>stage1_duration</code>	how long stage 1 of the gesture should take, i.e., how far into the gesture the "peak" will occur; default is such that no time scaling occurs
<code>stage2_duration</code>	how long stage 2 of the gesture should take; default is such that no time scaling occurs
<code>channel_A_weight</code>	the weight given to channel A of the gesture; ignored if the gesture does not have multiple channels

Returns:

0 on success, -1 on failure

5.12.407 function `diguyCharacter::execute_3stage_gesture`

Prototype:

```
int execute_3stage_gesture(const char* gesture_name,
    int reps = DIGUY_DEFAULT_INT,
    float stage1_duration = DIGUY_DEFAULT_FLOAT,
    float stage2_duration = DIGUY_DEFAULT_FLOAT,
    float stage3_duration = DIGUY_DEFAULT_FLOAT,
    float channel_A_weight = DIGUY_DEFAULT_FLOAT);
```

Description:

This function is similar to [execute_1stage_gesture\(\)](#).

Stage 1 of a 3 stage gesture is a motion that begins the gesture and leads to stage 2; e.g., raising the right arm in preparation for waving.

Stage 2 of a 3 stage gesture is a motion that can loop indefinitely; e.g., waving the right arm back and forth.

Stage 3 of a 3 stage gesture is the motion that ends the gesture; e.g., returning the right arm to the side of the body to complete the wave.

n repetitions of a 3 stage gesture will play the stage 1 motion once, followed by n reps of the stage 2 motion, followed by 1 stage 3 motion.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

<code>gesture_name</code>	name of the gesture of execute
<code>reps</code>	how many times the gesture should be repeated; default is 1 if no duration is specified, or best fit if there is a duration specified
<code>stage1_duration</code>	how long stage 1 of the gesture should take; default is such that no time scaling occurs
<code>stage2_duration</code>	how long each repetition of stage 2 of the gesture should take; default is such that no time scaling occurs
<code>stage3_duration</code>	how long stage 3 of the gesture should take; default is such that no time scaling occurs
<code>channel_A_weight</code>	the weight given to channel A of the gesture; ignored if the gesture does not have multiple channels

Returns:

0 on success, -1 on failure

5.12.408 function diguyCharacter::execute_table_gesture**Prototype:**

```
int execute_table_gesture(const char* gesture_name,
    int reps = DIGUY_DEFAULT_INT,
    const char* loop_action = NULL,
    const char* begin_action = NULL,
    const char* end_action = NULL);
```

Description:

The gesture will automatically begin and end. The only control over the executed gesture is whether it should be ended or aborted early. See [end_executing_gesture\(\)](#) and [abort_executing_gesture\(\)](#). For full control of gesture attributes, use [create_gesture\(\)](#) instead.

This function serves as a quick way of executing a table gesture. There are many capabilities of table gestures that are accessible only through the full [diguyCharacterGesture](#) object.

If the specified gesture is not a table gesture, no gesture is executed and an error value is returned.

A table stage gesture is comprised of a table of gesture actions that can be transitioned between without going through a neutral position.

n repetitions of a table gesture will repeat the loop action n times.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

<code>gesture_name</code>	name of the gesture of execute
<code>reps</code>	how many times the gesture should be repeated; default is 1
<code>loop_action</code>	which action in the gesture should be looped; default is action at index 1 of table
<code>begin_action</code>	which action the gesture should begin from; default is action at index 0
<code>end_action</code>	which action the gesture should end at; default is action at index 0

Returns:

0 on success, -1 on failure

5.12.409 function**diguyCharacter::set_executing_table_gesture_desired_action****Prototype:**

```
int set_executing_table_gesture_desired_action(const char* gesture_name,
const char* desired_action);
```

Description:

Documentation Pending

5.12.410 function diguyCharacter::end_executing_gesture**Prototype:**

```
int end_executing_gesture(const char* gesture_name);
```

Description:

This function alerts all the specified gesture that it should end. The gesture will stop repeating and play its "out" motions as soon as possible.

In order to immediately stop the gesture, use the [abort_executing_gesture\(\)](#) function.

Arguments:

`gesture_name` name of executing gesture to end

Returns:

0 on success, -1 on failure

See Also:[abort_executing_gesture\(\)](#)**5.12.411 function diguyCharacter::abort_executing_gesture****Prototype:**

```
int abort_executing_gesture(const char* gesture_name,
                           float max_rampdown_interval);
```

Description:

This function aborts a gesture previously executed by one of the `execute_*_gesture()` calls, if it is currently executing. If a gesture with the given name is not executing, this function will have no effect.

This call won't cause the gesture to switch to its "out" motion; it is a straight-forward blend out of whatever the gesture is currently doing. Use the [end_executing_gesture\(\)](#) call to make the gesture play its "out" motion as soon as possible, therefore ending the gesture in a smoother way.

Arguments:

<code>gesture_name</code>	name of executing gesture to abort
<code>max_rampdown_interval</code>	

max amount of time spent trying to smooth over any motion seams; set to 0.5 for a smooth transition, set to 0 for an immediate abort

Returns:

0 on success, -1 on failure

See Also:

[end_executing_gesture\(\)](#), [abort_all_gestures\(\)](#)

5.12.412 function diguyCharacter::end_all_gestures

Prototype:

```
int end_all_gestures();
```

Description:

This function alerts all active gestures that they should end. The gestures will stop repeating and play their "out" motions as soon as possible.

In order to immediately stop all gestures, use the [abort_all_gestures\(\)](#) function.

Returns:

0 on success, -1 on failure

See Also:

[abort_all_gestures\(\)](#),
[diguyCharacterGesture::end_gesture\(\)](#)

5.12.413 function diguyCharacter::abort_all_gestures**Prototype:**

```
int abort_all_gestures(float max_rampdown_interval);
```

Description:

This function aborts all active gestures. All gestures will immediately begin to lose their effect on the character's pose.

This won't cause the gestures to switch to their "out" motions; it is a straight-forward blend out of whatever the gesture is currently doing. Use the [end_all_gestures\(\)](#) call to make all active gestures play their "out" motions as soon as possible, therefore ending the gestures in a smoother way.

Arguments:

<code>max_rampdown_interval</code>	max amount of time spent trying to smooth over any motion seams; set to 0.5 for a smooth transition, set to 0 for an immediate abort

Returns:

0 on success, -1 on failure

See Also:

[end_all_gestures\(\)](#),
[diguyCharacterGesture::abort_now\(\)](#)

Sound Functions

These functions control the playback of [diguySound](#) objects owned by the scenario.

A [diguySoundInstance](#) is a playing copy of a [diguySound](#). Any given sound file (as encapsulated by [diguySound](#)) may be playing multiple times. For example, many soldiers firing their weapons may cause multiple instances the same gunshot sounds to be played simultaneously. It is possible for a characters to play a looping sound, but they are restricted to playing one instance of a looping sound at a time.

Any time a sound is played a sound instance is being used, either implicitly or explicitly. If the function calls [diguyCharacter::play_sound\(\)](#) or [diguyCharacter::play_3d_sound\(\)](#) are used, the sound instance is managed internally by DI-Guy; the sound instance will play to completion and then be destroyed.

If a sound instance is created by calling [diguyCharacter::create_sound_instance\(\)](#), more control is provided for sound playback. The sound can be started multiple times, stopped before it has completed, etc. Refer to the [diguySoundInstance](#) class for details.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.414 function diguyCharacter::play_sound**Prototype:**

```
int play_sound(const char* sound_name,
               float sound_gain = 1.0f,
               int sound_loops = 0);
```

Description:

This function will play the specified sound. The sound will be "ambient"; i.e., it will be equally audible from everywhere within the world. Use the function [play_3d_sound\(\)](#) to play a sound that moves with the character.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Unless the sound loops, the sound will play to completion and then stop. To stop the sound early or stop a looping sound call [diguyCharacter::stop_all_sounds\(\)](#) or [diguyCharacter::stop_sound\(\)](#).

Arguments

`sound_name` name of sound to play

`sound_gain` by how much sound's amplitude should be multiplied

`sound_loop` indicates that sound should loop indefinitely

There must be a [diguySound](#) in the scenario with the passed `sound_name`. See [diguyScenario::create_sound\(\)](#).

See [set_current_voice_actor\(\)](#) for information on how a voice actor can change which sound in the scenario is selected.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.415 function diguyCharacter::play_3d_sound**Prototype:**

```
int play_3d_sound(const char* sound_name,
                 float sound_gain = 1.0f,
                 int sound_loops = 0);
```

Description:

This function will play the specified sound at the character's location. As the character moves, the sound will move with it. Use the function [play_sound\(\)](#) to play a sound that is ambient and does not move with the character.

Note that not all sound modules support 3D sounds.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Unless the sound loops, the sound will play to completion and then stop. To stop the sound early or stop a looping sound call [diguyCharacter::stop_all_sounds\(\)](#) or [diguyCharacter::stop_sound\(\)](#).

Arguments

`sound_name` the name of the sound to play

`sound_gain` by how much the sound's amplitude should be multiplied

`sound_loop` indicates that the sound should loop indefinitely

There must be a [diguySound](#) in the scenario with the passed `sound_name`. See [diguyScenario::create_sound\(\)](#).

See [set_current_voice_actor\(\)](#) for information on how a voice actor can change which sound in the scenario is selected.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.416 function diguyCharacter::create_sound_instance

Prototype:

```
diguySoundInstance* create_sound_instance(const char* sound_name);
```

Description:

This function creates (but does not start playing) an instance of the specified sound.

Exact control over the sound instance is possible by making calls through the returned [diguySoundInstance](#) object.

To begin playing the sound, call [diguySoundInstance::begin_now\(\)](#).

Do not call delete on the returned object. It should be destroyed either by calling [diguyCharacter::destroy_sound_instance\(\)](#) or by calling [diguySoundInstance::set_automatic_destroy_flag\(1\)](#) on the returned object.

Arguments:

sound_name name of the sound for which to create an instance

There must be a [diguySound](#) in the scenario with the passed **sound_name**. See [diguyScenario::create_sound\(\)](#).

Returns:

pointer to object of type [diguySoundInstance](#)

5.12.417 function diguyCharacter::destroy_sound_instance**Prototype:**

```
int destroy_sound_instance(diguySoundInstance* sound_instance);
```

Description:

This function destroys a sound instance object created by [create_sound_instance\(\)](#).

Do not destroy a sound for which the [diguySoundInstance::set_automatic_destroy_flag\(1\)](#) has been called.

Arguments:

`sound_instance` sound instance to destroy

Returns:

0 on success, -1 on failure

5.12.418 function diguyCharacter::stop_all_sounds**Prototype:**

```
int stop_all_sounds(float rampdown_time = 0.0f);
```

Description:

This function stops all active sound instances from this character. All sounds being played by this character will fade out and stop after `rampdown_time`, by default fade out is instant.

Sound instances that are automatically managed will be destroyed. This includes sound started by a call to [diguyCharacter::play_sound\(\)](#) or a sound instance whose `set_automatic_destroy_flag()` function has been called.

[diguySoundInstance](#) pointers returned by a call to [diguyCharacter::create_sound_instance\(\)](#), which have not been set to be automatically destroyed, will still be valid.

Arguments:

`rampdown_time` how long to fade out the sound

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.419 function `diguyCharacter::stop_sound`

Prototype:

```
int stop_sound(const char* sound_name, float rampdown_time = 0.0f);
```

Description:

This function stops a named sound instance from this character.

Sound instances that are automatically managed will be destroyed. This includes sound started by a call to `diguyCharacter::play_sound()` or a sound instance whose `set_automatic_destroy_flag()` function has been called.

`diguySoundInstance` pointers returned by a call to `diguyCharacter::create_sound_instance()`, which have not been set to be automatically destroyed, will still be valid.

Arguments:

`sound_name` the name of the sound to stop

`rampdown_time` how long to fade out the sound

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.420 function `diguyCharacter::is_playing_sound`

Prototype:

```
int is_playing_sound(const char* sound_name);
```

Returns:

1 if the character is playing a the sound specified, 0 if not

5.12.421 function diguyCharacter::set_current_voice_actor**Prototype:**

```
int set_current_voice_actor(const char* voice_actor);
```

Description:

Sets the "voice actor" for the character.

Having a voice actor can change which sounds from the scenario a character selects when the `diguyCharacter::play_sound()` or `diguyCharacter::play_3d_sound()`.

When `play_sound()` is called the character asks the scenario for the sound with the specified name. If a voice actor has been set the character will first ask the scenario for the sound name prepended with the voice actor and a slash ("/"). If there isn't a sound with that modified name, it will ask the scenario for a sound with the exact specified name.

For example, if the character has the voice actor "carl" and is told to play the sound "run_away", the character will first look for a sound named "carl/run_away". If that isn't found, it will look for a sound named "run_away". (If that isn't found no sound is played.)

Lua Example:

A more specific example: say that various characters in the scenario need to say "hello". Some of the characters are male, some are female. There are two ways this can be set up.

The first way, not using a voice actor, is to have two sounds specified in the scenario: "male_hello" and "female_hello". These sounds specify the sound files "male_hello.wav" and "female_hello.wav", respectively. A male character would then do the following to say hello:

```
male_character:play_sound("male_hello")
```

A female character would do:

```
female_character:play_sound("female_hello")
```

This is fine, except when one script or character mind needs to be shared by a range of characters, some male and some female.

In this case it's better to use a voice actor. Say that the sounds specified above have the names "male/hello" and "female/hello"; they still have the separate sound files specified. If the character has its voice actor specified as "male" and is told to play the sound "hello", it will select the sound "male/hello" as

the sound to play. If its voice actor is "female" and is told to play the sound "hello", it will select the sound "female/hello".

```
-- called earlier: character:set_voice_actor("male")
character:play_sound("hello") -- plays sound "male/hello"
```

```
-- called earlier: character:set_voice_actor("female")
character:play_sound("hello") -- plays sound "female/hello"
```

Arguments:

`voice_actor` name of voice actor

`voice_actor` can be an arbitrary string. It doesn't need to match any character name, but should have matching sounds specified in the scenario.

Returns:

0 on success, -1 on failure

5.12.422 function `diguyCharacter::get_current_voice_actor`

Prototype:

```
const char* get_current_voice_actor();
```

Returns:

name of current voice actor, as set by `set_current_voice_actor()`

Network Functions

The following functions get and set attributes that are used by the DI-Guy Networking module.

In DI-Guy Networking, characters in a scenario can be divided into two groups:

- ◆ Characters that are "owned" by the scenario and are published to the network for other networking applications. These characters are usually called "local" or "published" characters.
- ◆ Characters that are created to represent network entities published by other networking applications. These characters are usually called "remote" or "reflected" characters.

A character cannot be both published and reflected.

Some functions are valid for published characters, some are valid for reflected characters, some are valid for both.

Some functions are valid with the DIS networking protocol, some are valid with the HLA protocol.

The "other networking applications" can be DI-Guy Scenario running on another computer, an application built against the DI-Guy SDK and DI-Guy Networking, or some other DIS or HLA compatible application such as various forms of SAF (semi-automated forces) programs.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.423 function diguyCharacter::get_DIS_lifeform_state**Prototype:**

```
int get_DIS_lifeform_state();
```

Description:

This function returns the state of the character in the form required by DIS.

Note that there is no corresponding `set_DIS_lifeform_state( )` function. The lifeform state is derived from the character's action, posture, and variant information.

Returns:

one of the following values defined in `diguy_dis_constants.h`:

Callable For Networked Entities That Are:

◇ DIS published and reflected

5.12.424 function diguyCharacter::get_DIS_primary_weapon_position**Prototype:**

```
int get_DIS_primary_weapon_position();
```

Description:

This function returns the position of the character's primary weapon, in the form required by DIS.

Returns:

one of the following values defined in `diguy_dis_constants.h`:

Callable For Networked Entities That Are:

◇ DIS published

5.12.425 function diguyCharacter::get_DIS_secondary_weapon_position

Prototype:

```
int get_DIS_secondary_weapon_position();
```

Description:

This function returns the position of the character's secondary weapon, in the form required by DIS.

Returns:

one of the following values defined in diguy_dis_constants.h:

Callable For Networked Entities That Are:

◊ DIS published

5.12.426 function diguyCharacter::set_network_marking

Prototype:

```
int set_network_marking(const char* network_marking);
```

Description:

This function sets the Marking Text for characters that are published to the network by DI-Guy Networking. The default network marking is the empty string "".

This function should only be called for published characters. The Marking Text is set internally for reflected characters.

Returns:

0 on success, -1 on failure

Callable For Networked Entities That Are:

◊ DIS published
◊ HLA published

5.12.427 function diguyCharacter::get_network_marking**Prototype:**

```
const char* get_network_marking();
```

Description:

This function gets the Marking Text for characters.

For published characters this value will be what was most recently set by [set_network_marking\(\)](#).

For reflected characters this value will be read from the reflected characters' entity state information.

Returns:

character's Marking Text; will never be NULL

Callable For Networked Entities That Are:

- ◇ DIS published and reflected
- ◇ HLA published and reflected

5.12.428 function diguyCharacter::set_is_network_published**Prototype:**

```
int set_is_network_published(int is_network_published);
```

Description:

This function sets whether this character will be published by DI-Guy Networking. This value is persistent, and does not depend on whether there actually **is** currently a network connection.

The default is 1, the character is published.

This function should **not** be called for reflected characters. It affects to both DIS and HLA networking.

Returns:

0 on success, -1 on failure

Callable For Networked Entities That Are:

- ◇ DIS published
- ◇ HLA published

5.12.429 function diguyCharacter::get_is_network_published

Prototype:

```
int get_is_network_published();
```

Description:

This function returns whether the character is published by DI-Guy Networking. It will always return 0 for reflected characters.

This function can be called for any character.

Returns:

1 if published, 0 if not

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

Callable For Networked Entities That Are:

- ◇ DIS published and reflected
- ◇ HLA published and reflected

5.12.430 function diguyCharacter::get_is_network_reflected

Prototype:

```
int get_is_network_reflected();
```

Description:

This function returns whether the character is a reflected character created by DI-Guy Networking.

This function can be called for any character.

Returns:

1 if reflected, 0 if not

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

Callable For Networked Entities That Are:

- ◇ DIS published and reflected
- ◇ HLA published and reflected

5.12.431 function diguyCharacter::get_is_network_paused

Prototype:

```
int get_is_network_paused();
```

Description:

This function returns whether this character is in a "paused" state. This will only be true for reflected characters.

A reflected character will be paused if the network entity it is representing is paused; i.e., its personal timeline is not advancing. The most common case of this is when this character is reflecting a published character of a DI-Guy Scenario application that is in a paused or stopped state.

If DI-Guy Scenario, paused characters will remain frozen in their current postures.

Returns:

1 if paused, 0 if not

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.432 function diguyCharacter::get_network_entity_number**Prototype:**

```
int get_network_entity_number();
```

Description:

This function returns the character's "network entity number". This number is valid and persistent regardless of whether the character is currently published.

This number is primarily used internally by DI-Guy Networking for creating entity IDs.

Returns:

character's network entity number

Callable For Networked Entities That Are:

- ◇ DIS published and reflected
- ◇ HLA published and reflected

5.12.433 function diguyCharacter::get_action_from_entity_state**Prototype:**

```
const char* get_action_from_entity_state(int damage_state,
    int life_form_state,
    int primary_weapon_state,
    float speed,
    int forward);
```

Description:

This function returns the name of an action for a character, given the DIS damage state, lifeform state, primary weapon state, speed of the character, and direction of travel.

Arguments:

damage_state	one of the values listed below
life_form_state	one of the values listed below
primary_weapon_state	one of the values listed below
speed	in meters/second
forward	whether the action should move forward (pass 1), or backward (pass 0)

Valid values for damage_state (defined in diguy_dis_constants.h):

- ◇ DIGUY_DIS_DAMAGE_NONE (0)
- ◇ DIGUY_DIS_DAMAGE_SLIGHT (1)
- ◇ DIGUY_DIS_DAMAGE_MODERATE (2)
- ◇ DIGUY_DIS_DAMAGE_DESTROYED (3)

Valid values for life_form_state (defined in diguy_dis_constants.h):

- ◇ DIGUY_DIS_LIFEFORM_UNUSED (0)
- ◇ DIGUY_DIS_LIFEFORM_UPRIGHT_STANDING_STILL (1)
- ◇ DIGUY_DIS_LIFEFORM_UPRIGHT_WALKING (2)
- ◇ DIGUY_DIS_LIFEFORM_UPRIGHT_RUNNING (3)
- ◇ DIGUY_DIS_LIFEFORM_KNEELING (4)
- ◇ DIGUY_DIS_LIFEFORM_PRONE (5)
- ◇ DIGUY_DIS_LIFEFORM_CRAWLING (6)
- ◇ DIGUY_DIS_LIFEFORM_SWIMMING (7)
- ◇ DIGUY_DIS_LIFEFORM_PARACHUTING (8)
- ◇ DIGUY_DIS_LIFEFORM_JUMPING (9)
- ◇ DIGUY_DIS_LIFEFORM_SITTING (10)
- ◇ DIGUY_DIS_LIFEFORM_SQUATTING (11)
- ◇ DIGUY_DIS_LIFEFORM_CROUCHING (12)
- ◇ DIGUY_DIS_LIFEFORM_WADING (13)

Valid values for primary_weapon_state (defined in diguy_dis_constants.h):

- ◇ DIGUY_DIS_PRIMARY_WEAPON_NOT_PRESENT (0)
- ◇ DIGUY_DIS_PRIMARY_WEAPON_STOWED (1)
- ◇ DIGUY_DIS_PRIMARY_WEAPON_DEPLOYED (2)
- ◇ DIGUY_DIS_PRIMARY_WEAPON_IN_FIRING_POSITION (3)

Note that a backward motion can be requested in a couple of ways: either by passing a negative speed, or by passing 0 for the forward argument, or both.

Returns:

name of an action if a match is found, NULL if not

Callable For Networked Entities That Are:

- ◇ DIS published

5.12.434 function diguyCharacter::set_DIS_damage_state

Prototype:

```
void set_DIS_damage_state(int state);
```

Description:

Sets the damage state for the network published character. It's still up to the end user to change the appearance based on this information.

Arguments:

state one of the values listed below

Valid values for state (defined in diguy_dis_constants.h):

- ◇ DIGUY_DIS_DAMAGE_NONE (0)
- ◇ DIGUY_DIS_DAMAGE_SLIGHT (1)
- ◇ DIGUY_DIS_DAMAGE_MODERATE (2)
- ◇ DIGUY_DIS_DAMAGE_DESTROYED (3)

Callable For Networked Entities That Are:

◇ DIS published

5.12.435 function diguyCharacter::get_DIS_damage_state

Prototype:

```
int get_DIS_damage_state();
```

Description:

Returns the damage state of a network published character. Note that it's still up to the end user to change the character appearance based on this information.

one of the values shown in set_DIS_damage_state()

5.12.436 function diguyCharacter::set_DIS_smoke_state**Prototype:**

```
void set_DIS_smoke_state(int state);
```

Description:

Sets the smoke state for a network published character. Starts or stops the "smoke" and "engine_smoke" appearance effects as appropriate.

Arguments:

state one of the values listed below

Valid values for state (defined in diguy_dis_constants.h):

- ◇ DIGUY_DIS_SMOKE_NOT_SMOKING (0)
- ◇ DIGUY_DIS_SMOKE_SMOKE_PLUME (1)
- ◇ DIGUY_DIS_SMOKE_ENGINE_SMOKE (2)
- ◇ DIGUY_DIS_SMOKE_ENGINE_SMOKE_AND_SMOKE_PLUME (3)

Callable For Networked Entities That Are:

- ◇ DIS published

5.12.437 function diguyCharacter::get_DIS_smoke_state**Prototype:**

```
int get_DIS_smoke_state();
```

Description:

Returns the smoke state of a network published character.

one of the values shown in set_DIS_smoke_state()

5.12.438 function diguyCharacter::set_DIS_flames_state**Prototype:**

```
void set_DIS_flames_state(int state);
```

Description:

Sets the flames state for a network published character. Starts or stops the "flames" appearance effect as appropriate.

Arguments:

state – one of the values listed below

Valid values for state (defined in diguy_dis_constants.h):

- ◇ DIGUY_DIS_FLAMES_NONE (0)
- ◇ DIGUY_DIS_FLAMES_PRESENT (1)

Callable For Networked Entities That Are:

- ◇ DIS published

5.12.439 function diguyCharacter::get_DIS_flames_state**Prototype:**

```
int get_DIS_flames_state();
```

Description:

Returns the flames state of a network published character.

one of the values shown in set_DIS_flames_state()

5.12.440 function diguyCharacter::set_DIS_power_plant_state**Prototype:**

```
void set_DIS_power_plant_state(int state);
```

Description:

Sets the power plant state for a network published character. Vehicle power plants default to being on.

Arguments:

state – one of the values listed below

Valid values for state (defined in diguy_dis_constants.h):

- ◇ DIGUY_DIS_POWER_PLANT_OFF (0)
- ◇ DIGUY_DIS_POWER_PLANT_ON (1)

Callable For Networked Entities That Are:

- ◇ DIS published

5.12.441 function diguyCharacter::get_DIS_power_plant_state**Prototype:**

```
int get_DIS_power_plant_state();
```

Description:

Returns the power plant state of a network published character.

one of the values shown in set_DIS_power_plant_state()

5.12.442 function diguyCharacter::set_DIS_trailing_effect_state**Prototype:**

```
void set_DIS_trailing_effect_state(int state);
```

Description:

Sets the trailing effect state for a network published character.

Arguments:

state – one of the values listed below

Valid values for state (defined in diguy_dis_constants.h):

- ◇ DIGUY_DIS_TRAILING_EFFECT_NONE (0)
- ◇ DIGUY_DIS_TRAILING_EFFECT_SMALL (1)
- ◇ DIGUY_DIS_TRAILING_EFFECT_MEDIUM (2)
- ◇ DIGUY_DIS_TRAILING_EFFECT_LARGE (3)

Callable For Networked Entities That Are:

◇ DIS published

5.12.443 function diguyCharacter::get_DIS_trailing_effect_state

Prototype:

```
int get_DIS_trailing_effect_state();
```

Description:

Returns the trailing effect state of a network published character.

one of the values shown in set_DIS_smoke_trail_state()

Formation Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.444 function diguyCharacter::call_formation

Prototype:

```
int call_formation(const char* formation_name,
                  int snap_to_position_flag = 0,
                  int break_existing_formation_flag = 1);
```

Description:

This function will create a new active formation with this character as its leader. The formation followers will fall into the formation given by the formation name.

The formation is broken up, or disbanded, by calling `break_formation()`.

Arguments:

<code>formation_name</code>	the name of the formation
<code>snap_to_position_flag</code>	if 1, following characters will immediately snap to their

assigned positions; otherwise following characters will move toward their assigned positions from their current positions

`break_existing_formation_flag` if 1, any formations this character is leading with be broken as if the `break_formation()` call were made

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.12.445 function `diguyCharacter::break_formation`

Prototype:

```
int break_formation(const char* formation_name = NULL);
```

Description:

This function breaks up a formation initiated by `call_formation()`.

Arguments:

`formation_name` the name of the formation to break; passing NULL or "(all)" will break all formations this character is leading

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

Guide Functions

A guide is something that can influence the position, orientation, or action of a character that is in free position mode and free action mode.

Most guides will look at the desired position and orientation of the character and do what is necessary to get the character there (or at least closer). How a guide accomplishes this depends on:

- ◆ the guide algorithm being used
- ◆ the parameters set for that algorithm
- ◆ the difference between current and desired settings

The desired position and orientation for a character can be explicitly set by the calls `set_desired_position()` and `set_desired_orientation()`. They may also be implicitly set if that character is in a formation.

Guides can be explicitly added to a character by calling `add_guide()` or `create_guide()`. They can also be implicitly added to a character if that character is called into a formation.

Guide parameters are set by calls to `diguyCharacterGuide::set_guide_algorithm_float_parameter()`.

Once a guide has been added to a character it can potentially affect the position, orientation, and/or action of a character until the guide has been removed. Guides can be explicitly removed from a character by calling `remove_guide()` or `remove_all_guides()`. They can be implicitly removed by the break-up of a formation, or by an automatic removal guide acquiring the target destination.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

Guide Algorithms

Below are descriptions of each guide algorithm, including an overview, its effects (if any) on action, position, and orientation, and the parameters that can be set for the algorithm.

Exact

Overview:

This is a trivial guide algorithm that simply sets the current position and orientation of the character to be equal to the desired position and orientation.

It can be used with or without a formation.

Effects on Action:

If the character is following a formation leader, and the leader has the same character type as the character, the character will match its action to that of the leader.

If the character types do not match, the character will try to use a suitable alternative action.

If the character is not in a formation, the action of the character is not affected.

Effects on Position and Orientation:

The character's position and orientation will be set to exactly equal the desired values without delay. If there is a substantial difference between current and desired values, a large jump in will be evident.

Parameters:

(none)

Follow1**Overview:**

The Follow1 guide algorithm attempts to reach the desired position by changing the character's action and orientation.

The Follow1 algorithm is commonly used by followers in formations. It can also be used by characters not in a formation.

Effects on Action:

If the character is in distance_zone_0 and is following a formation leader that is stopped, the character will match its action to that of the leader. Otherwise characters in distance_zone_0 will switch to their default still action (usually stand).

If the character is outside distance_zone_0 and is following a formation leader that is moving, the character will match its action to that of the leader. Otherwise characters outside distance_zone_0 will switch to their default moving action (usually walk).

If the character is outside distance_zone_1 the character will scale its speed by up to max_speed_scale.

Effects on Position:

There is no direct effect on position. The character attempts to reach its desired position by changing its current action and orientation.

Effects on Orientation:

If the character is in distance_zone_0 there is no effect on orientation.

If the character is beyond distance_zone_0 the character's orientation will be set to turn the character directly toward the desired position.

Note that currently the desired orientation is not used; instead the character orients toward the desired position.

Parameters:

max_speed_scale (description pending)

distance_zone_0 (description pending)

distance_zone_1 (description pending)

Follow2

Overview:

The Follow2 guide algorithm attempts to reach the desired position by changing the character's action and orientation.

The Follow2 algorithm is commonly used by followers in formations. It can also be used by characters not in a formation.

Effects on Action:

If following a formation leader:

- In distance_zone_0, leader stopped: Character will stop and match its still action to that of the leader if they are the same character type, otherwise the character will pick a different still action.
- In distance_zone_0, leader moving: Character will stop and use its default still action.
- In distance_zone_1, leader stopped: Character will use an appropriate moving action.
- In distance_zone_1, leader moving: Character will match its action to that of the leader.
- Beyond distance_zone_1, leader stopped or moving: Character will select a fast moving action, or speed up a slower action.

If not in a formation:

- In distance_zone_0: Character will stop using its default still action.
- In distance_zone_1: Character will use an appropriate moving action.
- Beyond distance_zone_1: Character will select a fast moving action, or speed up a slower action.

Effects on Position:

There is no direct effect on position. The character attempts to reach its desired position by changing its current action and orientation.

Effects on Orientation:

In general, if the character is moving it will turn toward its desired position. If stopped, the character will not turn.

Note that currently the desired orientation is not used; instead the character orients toward the desired position.

Parameters:

`distance_zone_0` distance zone 0 radius; default 0.5 meters
`distance_zone_1` distance zone 1 radius; default 2.0 meters
`max_speed_scale` determines how much character can speed up outside `distance_zone_1`

Drift1

Overview:

The Drift1 guide algorithm servos the character toward its desired position and orientation, effectively "scooting" the character toward where it should be.

The Drift1 algorithm can be used by formation followers, and can also be used for characters that are trying to match desired positions and orientations from outside sources.

Effects on Action:

If the character is following a formation leader, and the leader has the same character type as the character, the character will match its action to that of the leader.

If the character is not in a formation, the action of the character is not affected.

Effects on Position:

Drift 1 distance zones are spherical, centered on the character's current position.

If the character is in `distance_zone_0`, the character is considered "close enough", and no drifting occurs.

If the character is in `distance_zone_1`, the character will drift toward the desired position. The magnitude of the drift is proportional to the parameter `position_time_constant`.

If the character is in `distance_zone_2` or higher, the character is considered to be too far away and will be teleported directly to the desired position and orientation.

Effects on Orientation:

DI-Guy SDK Version 12.5.2 Index

Drift1 azimuth zones are symmetrical, extending from the character's forward direction symmetrically around to its left and right.

If the desired position is in azimuth_zone_0, the character is considered "close enough", and no drifting occurs.

If the desired position is in azimuth_zone_1, the character will drift toward the desired orientation. The magnitude of the drift is proportional to the parameter orientation_time_constant.

If the desired position is in azimuth_zone_2 or higher, the character is considered to be too far away and will be immediately reoriented directly toward the desired orientation.

Parameters:

distance_zone_0	distance zone 0 radius; default 0.0 meters
distance_zone_1	distance zone 1 radius; default 4.0 meters
azimuth_zone_0	azimuth zone 0 extent; default 0.0 degrees
azimuth_zone_1	azimuth zone 1 extent; default 180.0 degrees
position_time_constant	higher value results in slower drift; default is 1.0
orientation_time_constant	higher value results in slower drift; default is 1.0

5.12.446 function diguyCharacter::add_guide**Prototype:**

```
diguyCharacterGuide* add_guide(const char* guide_name,
    int automatically_disable_when_acquired = 0);
```

Description:

This function adds a copy of the specified scenario guide to the character. Any changes that are subsequently made to the added guide -- for example by a call to `diguyCharacterGuide::set_algorithm_float_parameter( )` -- are private to this copy of the guide and will not affect others.

The guide will be enabled by default. Guides can be disabled by calling `diguyCharacterGuide::set_guide_enabled(0)`, or by passing 1 for the argument `automatically_disable_when_acquired`.

Note that disabling a guide is not the same as removing it. Disabled guides are still a part of the character and can later be re-enabled by calling `diguyCharacterGuide::set_enabled( )`.

Arguments:

<code>guide_name</code>	name of guide to find and copy
<code>automatically_disable_when_acquired</code>	set to 1 to have the guide be automatically disabled when it has acquired its target

Returns:

object of type `diguyCharacterGuide`; if a guide with `guide_name` is not found, returns NULL

5.12.447 function diguyCharacter::create_guide**Prototype:**

```
diguyCharacterGuide* create_guide(const char* guide_name,
    const char* guide_algorithm,
    int automatically_disable_when_acquired = 0);
```

Description:

This function creates a new guide and adds it to the character.

The guide will be enabled by default. Guides can be disabled by calling `diguyCharacterGuide::set_guide_enabled( )`, or by passing 1 for the argument `automatically_disable_when_acquired`.

Important note: a new guide will always be created, even if one with the same name already exists

Arguments:

<code>guide_name</code>	name of guide to find and copy
<code>guide_algorithm</code>	name of guide algorithm to be used
<code>automatically_disable_when_acquired</code>	set to 1 to have the guide be automatically disabled when it has acquired its target

Returns:

object of type `diguyCharacterGuide`; if a guide with `guide_name` is not found, returns NULL

5.12.448 function diguyCharacter::find_or_create_guide

Prototype:

```
diguyCharacterGuide* find_or_create_guide(const char* guide_name,
    const char* guide_algorithm,
    int automatically_disable_when_acquired = 0);
```

Description:

Like [create_guide\(\)](#), but returns guide if it already exists

Arguments:

<code>guide_name</code>	name of guide to find and copy
<code>guide_algorithm</code>	name of guide algorithm to be used
<code>automatically_disable_when_acquired</code>	set to 1 to have the guide be automatically disabled when it has acquired its target

Returns:

object of type [diguyCharacterGuide](#); if a guide with `guide_name` is not found, returns NULL

5.12.449 function diguyCharacter::find_guide**Prototype:**

```
diguyCharacterGuide* find_guide(const char* guide_name);
```

Description:

This function returns a pointer to the specified guide of the character, if it exists.

Arguments:

`guide_name` name of guide to find

Returns:

object of type `diguyCharacterGuide`; if a guide with `guide_name` is not found, returns NULL

5.12.450 function diguyCharacter::remove_guide_with_name**Prototype:**

```
int remove_guide_with_name(const char* guide_name);
```

Description:

This function removes the specified guide from the character.

Arguments:

`guide_name` name of the guide to remove; must match name used by `add_guide()` or `create_guide()` call

Returns:

0 on success, -1 on failure

5.12.451 function diguyCharacter::remove_guide**Prototype:**

```
int remove_guide(diguyCharacterGuide* guide);
```

Description:

This function removes the specified guide from the character.

Arguments:

guide	object returned by add_guide() call

Returns:

0 on success, -1 on failure

5.12.452 function diguyCharacter::remove_all_guides**Prototype:**

```
int remove_all_guides();
```

Description:

This function removes all guides from the character.

Returns:

0 on success, -1 on failure

5.12.453 function diguyCharacter::remove_all_disabled_guides**Prototype:**

```
int remove_all_disabled_guides();
```

Description:

This function removes all guides that are disabled.

Returns:

0 on success, -1 on failure

Motion Texture Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.454 function diguyCharacter::add_motion_texture**Prototype:**

```
int add_motion_texture(const char* motex_name = "default",
    float texture_gain = 1.0f,
    int end_previous_motex = 1,
    float rampup_interval = 0.2f);
```

Description:

This function adds a "motion texture" to the character. A motion texture is a relatively long (usually 10 seconds or more) loop that will add natural-looking, low amplitude joint angle adjustments.

This function will have no effect on a dead character (see [die_now\(\)](#) and [get_dead\(\)](#)).

Arguments:

<code>motex_name</code>	name of the motion texture to add; see below
<code>texture_gain</code>	the relative amplitude of the motion texture; higher numbers result in more pronounced movements
<code>end_previous_motex</code>	flag denoting whether this motion texture replaces all previously added motion textures; pass 1 to end existing textures
<code>rampup_interval</code>	how long, in seconds, to take ramping up into the motion texture

List of currently available motion textures:

- ◇ "default"
- ◇ "motex_stand_casual"

If 1 is passed for the argument `end_previous_motex`, the `rampup_interval` argument will be used as the interval for ending existing motion textures.

Returns:

0 on success, -1 on failure

5.12.455 function diguyCharacter::end_motion_texture**Prototype:**

```
int end_motion_texture(float rampdown_interval = 0.2f);
```

Description:

This function gradually ends any active motion textures added by [add_motion_texture\(\)](#).

Arguments:

`rampdown_interval` how long, in seconds, to take ending the motion texture

Returns:

0 on success, -1 on failure

5.12.456 function diguyCharacter::set_initial_motion_texture**Prototype:**

```
int set_initial_motion_texture(const char* motex_name);
```

Description:

This function sets an initial motion texture that should be used by the character whenever the character is loaded from a scenario or the character's scenario is reset.

The gain of the motion texture can be set by calling [set_initial_motion_texture_gain\(\)](#). The initial motion texture gain is 1.0 by default.

See Also:

[add_motion_texture\(\)](#),
[set_initial_motion_texture_gain\(\)](#)

Arguments:

`motex_name` name of the motion texture

Returns:

0 on success, -1 on failure

5.12.457 function `diguyCharacter::get_initial_motion_texture`

Prototype:

```
const char* get_initial_motion_texture();
```

Description:

This function returns the initial motion texture as set by a call to `set_initial_motion_texture()`.

Returns:

name of initial motion texture

5.12.458 function**diguyCharacter::set_initial_motion_texture_gain****Prototype:**

```
int set_initial_motion_texture_gain(float motex_gain);
```

Description:

This function sets the gain of the initial motion texture of the character. The initial motion texture can be set by calling [set_initial_motion_texture\(\)](#).

Arguments:

`motex_gain` gain of the motion texture; higher gains result in more pronounced motion

Returns:

0 on success, -1 on failure

5.12.459 function**diguyCharacter::get_initial_motion_texture_gain****Prototype:**

```
float get_initial_motion_texture_gain();
```

Description:

This function returns the initial motion texture gain as set by a call to [set_initial_motion_texture_gain\(\)](#).

Returns:

gain of initial motion texture

History Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.460 function diguyCharacter::set_history_type

Prototype:

```
int set_history_type(diguyHistoryType history_type);
```

Description:

This function sets what type of "history" will be kept for the character.

If time is run backwards in a scenario, characters with history will replay the motions they performed at that time. Characters without history will stay frozen in place until time advances past the point from which it was run backwards.

The default is `DIGUY_HISTORY_TYPE_NONE` for characters created using the `diguyScenario::create_character()` function. Characters loaded from a .dss file will retain their history settings from the DI-Guy Scenario editing session.

Arguments:

`history_type` history type to be used for the character

`history_type` should be one of the following values:

`DIGUY_HISTORY_TYPE_NONE`

This history type saves no history.

`DIGUY_HISTORY_TYPE_COMPLETE`

This history type saves all of a character's history.

The initial amount of history to be saved, before memory buffers need to be resized, can be set by the function `set_history_size_t( )`. The size by which memory buffers grow each time they do need to get resized by be set by `set_history_size_increment_t( )`.

`DIGUY_HISTORY_TYPE_FIRST`

This history type is no longer supported. Characters from a .dss that have this history type will have their history type set to `DIGUY_HISTORY_TYPE_COMPLETE`.

`DIGUY_HISTORY_TYPE_LAST`

This history type is no longer supported. Characters from a .dss that have this history type will have their history type set to `DIGUY_HISTORY_TYPE_COMPLETE`.

Returns:

0 on success, -1 on failure

5.12.461 function diguyCharacter::get_history_type**Prototype:**

```
diguyHistoryType get_history_type();
```

Returns:

the current history type of the character; see [set_history_type\(\)](#)

5.12.462 function diguyCharacter::set_history_size_t**Prototype:**

```
int set_history_size_t(float t);
```

Description:

This function sets the size of the character's history buffer, in seconds.

Arguments:

t	length of history buffer, in seconds

Returns:

0 on success, -1 on failure

5.12.463 function diguyCharacter::get_history_size_t**Prototype:**

```
float get_history_size_t();
```

Description:

This function returns the current history buffer size. See [set_history_size_t\(\)](#).

Returns:

current history size

5.12.464 function diguyCharacter::set_history_size_increment_t**Prototype:**

```
int set_history_size_increment_t(float t);
```

Description:

This function sets the size of the character's history buffer increment, in seconds.

Arguments:

t	length of history buffer increment, in seconds

Returns:

0 on success, -1 on failure

5.12.465 function diguyCharacter::get_history_size_increment_t**Prototype:**

```
float get_history_size_increment_t();
```

Description:

This function returns the current history buffer size increment. See [set_history_size_increment_t\(\)](#).

Returns:

current history size increment

Meta-Action Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.466 function diguyCharacter::set_desired_posture**Prototype:**

```
void set_desired_posture(diguyMotionPosture desired_posture);
```

Description:

This function sets the desired posture of the character. This, along with the desired variant, is used primarily by some guides. Specifically, to help choose which actions are used to reach the desired position and orientation of the character.

Pass **DIGUY_MOTION_POSTURE_ANY** for there to be no desired posture.

Arguments:

posture	new desired posture

5.12.467 function diguyCharacter::get_desired_posture**Prototype:**

```
diguyMotionPosture get_desired_posture();
```

Returns:

current desired posture of the character

5.12.468 function diguyCharacter::set_desired_variant**Prototype:**

```
void set_desired_variant(diguyMotionVariant desired_variant);
```

Description:

This function sets the desired variant of the character. This, along with the desired posture, is used primarily by some guides. Specifically, to help choose which actions are used to reach the desired position and orientation of the character.

Pass [DIGUY_MOTION_VARIANT_ANY](#) for there to be no desired variant.

Arguments:

<code>variant</code>	new desired variant

5.12.469 function diguyCharacter::get_desired_variant**Prototype:**

```
diguyMotionVariant get_desired_variant();
```

Returns:

current desired variant of the character

5.12.470 function diguyCharacter::get_action_from_description**Prototype:**

```
const char* get_action_from_description(diguyMotionDirection direction,
    diguyMotionPosture posture,
    diguyMotionVariant variant,
    float speed,
    diguyParameterComparison comparison = DIGUY_PARAMETER_COMPARISON_NEAREST,
    float comparison_tolerance = DIGUY_DEFAULT_FLOAT,
    int match_posture_before_variant = 1,
    diguyMotionPosture posture_alternate1 = DIGUY_MOTION_POSTURE_UNKNOWN,
    diguyMotionPosture posture_alternate2 = DIGUY_MOTION_POSTURE_UNKNOWN,
    diguyMotionVariant variant_alternate1 = DIGUY_MOTION_VARIANT_UNKNOWN,
    diguyMotionVariant variant_alternate2 = DIGUY_MOTION_VARIANT_UNKNOWN,
    int return_random_match = 0);
```

Description:

This function looks up the character action that most closely matches the given criteria.

Arguments:

direction	the direction of travel relative to the character's facing direction
posture	overall desired posture; pass DIGUY_MOTION_POSTURE_ANY to specify that any posture will do
variant	overall desired variant; pass DIGUY_MOTION_VARIANT_ANY to specify that any variant will do
speed	speed in meters per second
comparison	how the passed speed should compare to the action's speed
comparison_tolerance	tolerance of comparison
match_posture_before_variant	pass 1 if posture should be matched before variant if an exact match is not found; pass 0 if variant should be matched before posture
posture_alternate1	if an exact match is not found, use this as the first alternate posture; pass DIGUY_MOTION_POSTURE_UNKNOWN to specify that no alternate should be used
posture_alternate2	if an exact match is not found, use this as the second alternate posture; pass DIGUY_MOTION_POSTURE_UNKNOWN to specify that no alternate should be used
variant_alternate1	

	if an exact match is not found, use this as the first alternate variant; pass DIGUY_MOTION_VARIANT_UNKNOWN to specify that no alternate should be used
<code>variant_alternate2</code>	if an exact match is not found, use this as the second alternate variant; pass DIGUY_MOTION_VARIANT_UNKNOWN to specify that no alternate should be used
<code>return_random_match</code>	(Experimental) if multiple actions match the specified criteria randomly return one of them. Typically only used for still motions and restricted to EQUAL and NEAREST comparisons

The direction argument should be one of the following:

- ◇ `DIGUY_MOTION_DIRECTION_NONE` – does not move character
- ◇ `DIGUY_MOTION_DIRECTION_FORWARD` – moves character forward
- ◇ `DIGUY_MOTION_DIRECTION_BACKWARD` – moves character backward
- ◇ `DIGUY_MOTION_DIRECTION_LEFT` – moves character left
- ◇ `DIGUY_MOTION_DIRECTION_RIGHT` – moves character right
- ◇ `DIGUY_MOTION_DIRECTION_UP` – moves character up
- ◇ `DIGUY_MOTION_DIRECTION_DOWN` – moves character down
- ◇ `DIGUY_MOTION_DIRECTION_TURN_LEFT` – turns character left
- ◇ `DIGUY_MOTION_DIRECTION_TURN_RIGHT` – turns character right
- ◇ `DIGUY_MOTION_DIRECTION_NONLINEAR` – moves character in multiple directions
- ◇ `DIGUY_MOTION_DIRECTION_LINEAR_OFFAXIS` – moves character in a line, but not on any of the principle axes

Not every direction is supported by every character. If the character does not have any actions that move in the specified direction, the returned action will be NULL.

Use the calls

`diguyScenario::get_character_type_num_directions()`

and

`diguyScenario::get_character_type_direction_at_index()`

to enumerate which directions a character can travel.

The posture arguments should be one of the following:

- ◇ `DIGUY_MOTION_POSTURE_UNKNOWN` – unknown posture
- ◇ `DIGUY_MOTION_POSTURE_ANY` – any posture; a wildcard
- ◇ `DIGUY_MOTION_POSTURE_UPRIGHT` – upright posture; e.g., standing or walking

- ◇ [DIGUY_MOTION_POSTURE_CROUCHED](#) – crouched posture; e.g., kneeling or walking low
- ◇ [DIGUY_MOTION_POSTURE_PRONE](#) – prone posture; e.g., lying prone or crawling
- ◇ [DIGUY_MOTION_POSTURE_SITTING](#) – sitting posture

The posture arguments specify the overall posture the action should have. Most characters have an upright posture. The other postures a character has is dependent on character type. Passing [DIGUY_MOTION_POSTURE_ANY](#) for the arguments posture, posture_alterate1, or posture_alterate2 means that the posture of the action should not be used to determine a match.

Use the calls

[diguyScenario::get_character_type_num_postures\(\)](#) and [diguyScenario::get_character_type_posture_at_index\(\)](#) to enumerate which postures a specific character type has.

The variant arguments specify the overall mood the action should convey. Most characters have a "normal" variant. The other variants a character has is dependent on character type. Passing [DIGUY_MOTION_VARIANT_ANY](#) for the arguments variant, variant_alterate1, or variant_alterate2 means that the variant of the action should not be used to determine a match.

Use the calls

[diguyScenario::get_character_type_num_variants\(\)](#) and [diguyScenario::get_character_type_variant_at_index\(\)](#) to enumerate which variants a specific character type has.

The comparison argument should be one of the following:

- ◇ [DIGUY_PARAMETER_COMPARISON_EQUAL](#) – matched action's speed must be within comparison_tolerance of passed speed
- ◇ [DIGUY_PARAMETER_COMPARISON_NEAREST](#) – matched action's speed is closer than any other potential match's, but still within comparison_tolerance
- ◇ [DIGUY_PARAMETER_COMPARISON_GREATER](#) – matched action's speed faster than passed speed, but still within comparison_tolerance
- ◇ [DIGUY_PARAMETER_COMPARISON_LESS](#) – matched action's speed slower than passed speed, but still within comparison_tolerance

The difference between [DIGUY_PARAMETER_COMPARISON_EQUAL](#) and [DIGUY_PARAMETER_COMPARISON_NEAREST](#) is largely in the value used for comparison_tolerance if [DIGUY_DEFAULT_FLOAT](#) is passed; the comparison_tolerance argument defaults to 0.001 for [DIGUY_PARAMETER_COMPARISON_EQUAL](#), and 16384.0 for all other comparison types. Otherwise these two comparison types operate the same.

Returns:

name of an action if a match is found, NULL if not

C++ Example:

```
diguyCharacter* ch = scenario->get_character_at_index(0);
const char* new_action;

//
//  Get an action that moves forward in an upright posture,
//  whose speed move at least 2.0 meters per second:
//
new_action = ch->get_action_from_description(
DIGUY_MOTION_DIRECTION_FORWARD,
DIGUY_MOTION_POSTURE_UPRIGHT,
DIGUY_MOTION_VARIANT_ANY,
2.0,
DIGUY_PARAMETER_COMPARISON_GREATER, DIGUY_DEFAULT_FLOAT);

//
//  Get an action that is standing still, with the variant
//  DIGUY_MOTION_VARIANT_READY.  If no action has that variant,
//  try DIGUY_MOTION_VARIANT_AIM as variant_alterate1,
//  and DIGUY_MOTION_VARIANT_ANY as variant_alterate2:
//
new_action = ch->get_action_from_description(
DIGUY_MOTION_DIRECTION_NONE,
DIGUY_MOTION_POSTURE_ANY,
DIGUY_MOTION_VARIANT_READY,
0.0,
DIGUY_PARAMETER_COMPARISON_NEAREST, DIGUY_DEFAULT_FLOAT,
0,
DIGUY_MOTION_POSTURE_UNKNOWN,
DIGUY_MOTION_POSTURE_UNKNOWN,
DIGUY_MOTION_VARIANT_AIM,
DIGUY_MOTION_VARIANT_ANY);
```

5.12.471 function

diguyCharacter::get_turning_action_from_description

Prototype:

```
const char* get_turning_action_from_description(diguyMotionDirection direction,
    diguyMotionPosture posture,
    diguyMotionVariant variant,
    float turn_amount,
    diguyParameterComparison comparison = DIGUY_PARAMETER_COMPARISON_NEAREST,
    float comparison_tolerance = DIGUY_DEFAULT_FLOAT,
    int match_posture_before_variant = 1,
    diguyMotionPosture posture_alternate1 = DIGUY_MOTION_POSTURE_UNKNOWN,
    diguyMotionPosture posture_alternate2 = DIGUY_MOTION_POSTURE_UNKNOWN,
    diguyMotionVariant variant_alternate1 = DIGUY_MOTION_VARIANT_UNKNOWN,
    diguyMotionVariant variant_alternate2 = DIGUY_MOTION_VARIANT_UNKNOWN);
```

Description:

This function is very similar to [get_action_from_description\(\)](#), but substitutes turn amount per action iteration for speed.

Arguments:

direction	the direction of travel relative to the character's facing direction
posture	overall desired posture; pass DIGUY_MOTION_POSTURE_ANY to specify that any posture will do
variant	overall desired variant; pass DIGUY_MOTION_VARIANT_ANY to specify that any variant will do
turn_amount	how many degrees each iteration of the action should turn the character
comparison	how the passed turn amount should compare to the action's turn amount
comparison_tolerance	tolerance of comparison
match_posture_before_variant	pass 1 if posture should be matched before variant if an exact match is not found; pass 0 if variant should be matched before posture
posture_alternate1	if an exact match is not found, use this as the first alternate posture; pass DIGUY_MOTION_POSTURE_UNKNOWN to specify that no alternate should be used
posture_alternate2	if an exact match is not found, use this as the second alternate posture; pass DIGUY_MOTION_POSTURE_UNKNOWN to specify that no alternate should be used

<code>variant_alternate1</code>	if an exact match is not found, use this as the first alternate variant; pass <code>DIGUY_MOTION_VARIANT_UNKNOWN</code> to specify that no alternate should be used
<code>variant_alternate2</code>	if an exact match is not found, use this as the second alternate variant; pass <code>DIGUY_MOTION_VARIANT_UNKNOWN</code> to specify that no alternate should be used

The direction argument should be one of the following:

- ◇ `DIGUY_MOTION_DIRECTION_TURN_LEFT` – turns character left
- ◇ `DIGUY_MOTION_DIRECTION_TURN_RIGHT` – turns character right

Not every direction is supported by every character. If the character does not have any actions that move in the specified direction, the returned action will be `NULL`.

The turn amount argument should be a positive number.

Refer to `get_action_from_description()` for more information on other arguments.

Returns:

name of an action if a match is found, `NULL` if not

Variable Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.472 function diguyCharacter::get_num_variables

Prototype:

```
int    get_num_variables();
```

Returns:

number of variables the character has

5.12.473 function diguyCharacter::get_variable_at_index**Prototype:**

```
diguyVariable* get_variable_at_index(int index);
```

Returns:

pointer of type `diguyVariable`; NULL if no variable at the specified index

Arguments:

index	index of the variable; indices start at 0

5.12.474 function diguyCharacter::find_variable**Prototype:**

```
diguyVariable* find_variable(const char* name);
```

Description:

This function returns a pointer to the specified variable.

Arguments:

name	name of variable to be found

Returns:

pointer of type `diguyVariable`; NULL if not found

5.12.475 function diguyCharacter::find_or_create_variable**Prototype:**

```
diguyVariable* find_or_create_variable(const char* name);
```

Description:

This function finds the variable with the given name or creates it if it doesn't exist.

Arguments:

name	name of the variable to find or create

Returns:

pointer of type [diguyVariable](#); should never be NULL

5.12.476 function diguyCharacter::destroy_variable**Prototype:**

```
int destroy_variable(diguyVariable* variable);
```

Description:

This function destroys the passed variable.

Arguments:

variable	pointer to a diguyVariable

Returns:

0 on success, -1 on failure

5.12.477 function diguyCharacter::variable_equal_to_int**Prototype:**

```
int variable_equal_to_int(const char* varname, int value);
```

Returns:

1 if variable is equal to passed value, 0 if not

Arguments:

varname	name of variable to compare
value	value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.478 function diguyCharacter::variable_greater_than_int**Prototype:**

```
int variable_greater_than_int(const char* varname, int value);
```

Returns:

1 if variable is greater than passed value, 0 if not

Arguments:

varname	name of variable to compare
value	value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.479 function diguyCharacter::variable_less_than_int**Prototype:**

```
int variable_less_than_int(const char* varname, int value);
```

Returns:

1 if variable is less than passed value, 0 if not

Arguments:

varname	name of variable to compare
value	value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.480 function diguyCharacter::variable_equal_to_float**Prototype:**

```
int variable_equal_to_float(const char* varname, float value);
```

Returns:

1 if variable is equal to passed value, 0 if not

Arguments:

varname	name of variable to compare
value	value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.481 function diguyCharacter::variable_greater_than_float**Prototype:**

```
int variable_greater_than_float(const char* varname, float value);
```

Returns:

1 if variable is greater than passed value, 0 if not

Arguments:

varname	name of variable to compare
value	value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.482 function diguyCharacter::variable_less_than_float**Prototype:**

```
int variable_less_than_float(const char* varname, float value);
```

Returns:

1 if variable is less than passed value, 0 if not

Arguments:

varname	name of variable to compare
value	value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.483 function diguyCharacter::variable_equal_to_string**Prototype:**

```
int variable_equal_to_string(const char* varname, const char* value);
```

Returns:

1 if variable is equal than passed value, 0 if not

Arguments:

varname	name of variable to compare
value	value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.484 function diguyCharacter::variable_greater_than_string**Prototype:**

```
int variable_greater_than_string(const char* varname, const char* value);
```

Returns:

1 if variable is greater than passed value (comes later alphabetically), 0 if not

Arguments:

varname	name of variable to compare
value	value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.485 function diguyCharacter::variable_less_than_string**Prototype:**

```
int variable_less_than_string(const char* varname, const char* value);
```

Returns:

1 if variable is less than passed value (comes sooner alphabetically), 0 if not

Arguments:

varname	name of variable to compare
value	value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.486 function diguyCharacter::variable_decrement_value_as_int**Prototype:**

```
int variable_decrement_value_as_int(const char* varname, int value = 1);
```

Description:

This function decrements the variable by the passed amount.

Arguments:

varname	name of variable to modify
value	amount to decrement

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.487 function diguyCharacter::variable_increment_value_as_int

Prototype:

```
int variable_increment_value_as_int(const char* varname, int value = 1);
```

Description:

This function increments the variable by the passed amount.

Arguments:

varname	name of variable to modify
value	amount to increment

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.488 function diguyCharacter::variable_set_value_as_int

Prototype:

```
int variable_set_value_as_int(const char* varname,
                             int value,
                             int create_if_doesnt_exist = 1);
```

Description:

This function is a convenience function for setting the variable to a new integer value without retrieving the [diguyVariable](#) object.

Arguments:

<code>varname</code>	name of variable to modify
<code>value</code>	new value
<code>create_if_doesnt_exist</code>	pass 1 to create the variable if it doesn't exist

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.489 function diguyCharacter::set_ivar**Prototype:**

```
int set_ivar(const char* varname,  
            int value,  
            int create_if_doesnt_exist = 1);
```

Description:

Short form of `variable_set_value_as_int()`.

5.12.490 function diguyCharacter::get_ivar**Prototype:**

```
int get_ivar(const char* varname,
             int create_if_doesnt_exist = 1);
```

Description:

This function is a convenience function for retrieving the value of a variable without retrieving the [diguyVariable](#) object.

If the variable doesn't exist and create_if_doesnt_exist is 0, the returned value will be 0.

Arguments:

varname	name of variable to query
create_if_doesnt_exist	pass 1 to create the variable if it doesn't exist

Returns:

value of variable as int

5.12.491 function diguyCharacter::variable_set_value_as_float**Prototype:**

```
int variable_set_value_as_float(const char* varname,
                                float value,
                                int create_if_doesnt_exist = 1);
```

Description:

Similar to [variable_set_value_as_int\(\)](#), for sets a floating point value instead.

5.12.492 function diguyCharacter::set_fvar**Prototype:**

```
int set_fvar(const char* varname,  
            float value,  
            int create_if_doesnt_exist = 1);
```

Description:

Short form of `variable_set_value_as_float()`.

5.12.493 function diguyCharacter::get_fvar**Prototype:**

```
float get_fvar(const char* varname,  
              int create_if_doesnt_exist = 1);
```

Description:

Similar to `variable_get_value_as_int()`, for returns a floating point value instead.

If the variable doesn't exist and `create_if_doesnt_exist` is 0, the returned value will be 0.0.

5.12.494 function diguyCharacter::variable_set_value_as_string**Prototype:**

```
int variable_set_value_as_string(const char* varname,  
    const char* value,  
    int create_if_doesnt_exist = 1);
```

Description:

Similar to [variable_set_value_as_int\(\)](#), for sets a string value instead.

5.12.495 function diguyCharacter::set_svar**Prototype:**

```
int set_svar(const char* varname,  
    const char* value,  
    int create_if_doesnt_exist = 1);
```

Description:

Short form of [variable_set_value_as_string\(\)](#).

5.12.496 function diguyCharacter::get_svar

Prototype:

```
const char* get_svar(const char* varname,  
                    int create_if_doesnt_exist = 1);
```

Description:

Similar to `variable_get_value_as_int()`, for returns a string value instead.

If the variable doesn't exist and `create_if_doesnt_exist` is 0, the returned value will be "".

Label Functions

The functions in this section control the text, visibility, and colors of character labels.

Currently character labels are only enabled in DI-Guy Scenario.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.497 function diguyCharacter::get_name_label**Prototype:**

```
diguyViewLabel* get_name_label();
```

Description:

This function returns a pointer to the label object of the character.

Currently character labels are only enabled in DI-Guy Scenario.

Returns:

pointer to the character's label

5.12.498 function diguyCharacter::set_label_text**Prototype:**

```
int set_label_text(const char* label_text);
```

Description:

This function sets the text of the character's label. By default the label text matches the character's name.

Currently character labels are only enabled in DI-Guy Scenario.

Arguments:

`label_text` new text for the label

Returns:

0 on success, -1 on failure

5.12.499 function diguyCharacter::get_label_text**Prototype:**

```
const char* get_label_text();
```

Description:

This function returns the character's label text. See [set_label_text\(\)](#).

Returns:

current label text; the returned pointer will never be NULL

5.12.500 function diguyCharacter::set_label_is_visible**Prototype:**

```
int set_label_is_visible(int is_visible);
```

Description:

This function sets whether the label of this character is visible. This setting will only be taken into account if the visibility flag for Character Labels is set to "Per Object".

Currently character labels are only enabled in DI-Guy Scenario.

Arguments:

`is_visible` pass 1 to show, pass 0 to hide

Returns:

0 on success, -1 on failure

5.12.501 function diguyCharacter::get_label_is_visible**Prototype:**

```
int get_label_is_visible();
```

Description:

This function returns the most recent setting of `set_label_is_visible()`.

Returns:

1 if visible, 0 if not

5.12.502 function diguyCharacter::set_label_text_color**Prototype:**

```
int set_label_text_color(float r, float g, float b, float a);
```

Description:

This function sets the text color of the character's label. By default the label text color is opaque white.

Currently character labels are only enabled in DI-Guy Scenario.

Arguments:

`r, g, b,` the red, green, blue, and alpha components of the text color
`a`

An alpha component of 0 makes the text completely transparent. An alpha component of 1 makes the text completely opaque. Values in between result in varying levels of transparency.

Returns:

0 on success, -1 on failure

5.12.503 function diguyCharacter::set_label_text_color_vec4

Prototype:

```
int set_label_text_color_vec4(diguyVec4f color);
```

Description:

Alternate version of [set_label_text_color\(\)](#).

5.12.504 function diguyCharacter::get_label_text_color**Prototype:**

```
int get_label_text_color(float* r, float* g, float* b, float* a);
```

Description:

This function returns, in the passed variable pointers, the most recent setting of `set_label_text_color()`.

Returns:

0 on success, -1 on failure

5.12.505 function diguyCharacter::get_label_text_color_vec4**Prototype:**

```
diguyVec4f get_label_text_color_vec4();
```

Description:

Alternate version of `get_label_text_color()`.

5.12.506 function diguyCharacter::set_label_text_shadow_color**Prototype:**

```
int set_label_text_shadow_color(float r, float g, float b, float a);
```

Description:

This function sets the text shadow color of the character's label. By default the label text shadow color is gray.

Currently character labels are only enabled in DI-Guy Scenario.

Arguments:

r, g, b,	the red, green, blue, and alpha components of the text
a	shadow color

An alpha component of 0 makes the text completely transparent. An alpha component of 1 makes the text completely opaque. Values in between result in varying levels of transparency.

Returns:

0 on success, -1 on failure

5.12.507 function diguyCharacter::set_label_text_shadow_color_vec4

Prototype:

```
int set_label_text_shadow_color_vec4(diguyVec4f color);
```

Description:

Alternate version of [set_label_text_shadow_color\(\)](#).

5.12.508 function diguyCharacter::get_label_text_shadow_color

Prototype:

```
int get_label_text_shadow_color(float* r, float* g, float* b, float* a);
```

Description:

This function returns, in the passed variable pointers, the most recent setting of [set_label_text_shadow_color\(\)](#).

Returns:

0 on success, -1 on failure

5.12.509 function**diguyCharacter::get_label_text_shadow_color_vec4****Prototype:**

```
diguyVec4f get_label_text_shadow_color_vec4();
```

Description:

Alternate version of [get_label_text_shadow_color\(\)](#).

5.12.510 function**diguyCharacter::set_label_text_shadow_is_visible****Prototype:**

```
int set_label_text_shadow_is_visible(int is_visible);
```

Description:

This function sets the whether the text shadow is visible.

Arguments:

visible pass 1 to make shadow visible, 0 to make it invisible

Returns:

0 on success, -1 on failure

5.12.511 function diguyCharacter::get_label_text_shadow_is_visible

Prototype:

```
int get_label_text_shadow_is_visible();
```

Returns:

1 if text shadow is visible, 0 if not

5.12.512 function diguyCharacter::set_label_background_color

Prototype:

```
int set_label_background_color(float r, float g, float b, float a);
```

Description:

This function sets the background color of the character's label. By default the label background color is transparent blue.

Currently character labels are only enabled in DI-Guy Scenario.

Arguments:

r, g, b, the red, green, blue, and alpha components of the text color
a

An alpha component of 0 makes the text completely transparent. An alpha component of 1 makes the text completely opaque. Values in between result in varying levels of transparency.

Returns:

0 on success, -1 on failure

5.12.513 function

diguyCharacter::set_label_background_color_vec4

Prototype:

```
int set_label_background_color_vec4(diguyVec4f color);
```

Description:

Alternate version of [set_label_background_color\(\)](#).

5.12.514 function diguyCharacter::get_label_background_color**Prototype:**

```
int get_label_background_color(float* r, float* g, float* b, float* a);
```

Description:

This function returns, in the passed variable pointers, the most recent setting of `set_label_background_color()`.

Returns:

0 on success, -1 on failure

5.12.515 function diguyCharacter::get_label_background_color_vec4**Prototype:**

```
diguyVec4f get_label_background_color_vec4();
```

Description:

Alternate version of `get_label_background_color()`.

5.12.516 function diguyCharacter::set_label_link_offset_override**Prototype:**

```
void set_label_link_offset_override(float offset);
```

Description:

This function sets how high the label of the character will float above the character.

Arguments:

offset	offset, in meters, of label above character

5.12.517 function diguyCharacter::get_label_link_offset_override**Prototype:**

```
float get_label_link_offset_override();
```

Returns:

how high the label of the character will float above the character

5.12.518 function

diguyCharacter::set_label_shows_character_state

Prototype:

```
int set_label_shows_character_state(int show_state);
```

Description:

This function sets whether the label text should show some basic state information about the character instead of its name or the text set by [set_label_text\(\)](#).

The state shown is: Character name, current path, current action. The current path and current action will change as the character moves through paths and actions.

Arguments:

`show_state` pass 1 to show state, 0 to show original text

5.12.519 function

diguyCharacter::get_label_shows_character_state

Prototype:

```
int get_label_shows_character_state();
```

Returns:

whether character state information is shown instead of the character's name or other label text

Impact Functions

Impact records are how DI-Guy records impact events. These events are typically the result of a character firing its weapon.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.520 function diguyCharacter::get_last_impact_record

Prototype:

```
diguyImpact* get_last_impact_record();
```

Returns:

pointer of type [diguyImpact](#); NULL if there has been no recent impact recorded; useful for finding out who or what was hit

5.12.521 function diguyCharacter::get_last_impact_number**Prototype:**

```
int get_last_impact_number();
```

Returns:

the last impact number

5.12.522 function diguyCharacter::get_last_impact_been_processed**Prototype:**

```
int get_last_impact_been_processed();
```

Returns:

the processed flag on the [diguyImpact](#) class

5.12.523 function

`diguyCharacter::set_last_impact_been_processed`

Prototype:

```
void set_last_impact_been_processed();
```

Description:

Sets the processed flag on the `diguyImpact` object.

5.12.524 function

`diguyCharacter::set_max_unprocessed_impacts`

Prototype:

```
void set_max_unprocessed_impacts(int max_unprocessed_impacts);
```

Description:

Sets the maximum number of unprocessed impacts there can be.

The unprocessed impact array is a first-in-first-out queue. If an impact occurs that pushes the number of unprocessed impacts past this maximum number, earlier impacts will be dropped from the array.

The initial value is 10.

5.12.525 function diguyCharacter::get_max_unprocessed_impacts

Prototype:

```
int get_max_unprocessed_impacts();
```

Returns:

maximum number of unprocessed impacts

5.12.526 function diguyCharacter::get_last_impact_link

Prototype:

```
const char* get_last_impact_link();
```

Returns:

the last impact link

5.12.527 function diguyCharacter::get_last_impact_link_is**Prototype:**

```
int get_last_impact_link_is(const char* link_name);
```

Returns:

1 if the last impact link is the link named

**5.12.528 function
diguyCharacter::get_last_impact_is_from_vehicle****Prototype:**

```
int get_last_impact_is_from_vehicle();
```

Returns:

1 if the last impact resulted from an impact with a vehicle, 0 if not

5.12.529 function diguyCharacter::get_last_impact_speed_is_greater_than

Prototype:

```
int get_last_impact_speed_is_greater_than(float impact_speed);
```

Returns:

1 if the last impact speed is greater than the passed speed, 0 if not

5.12.530 function diguyCharacter::set_is_immune_to_impacts

Prototype:

```
void set_is_immune_to_impacts(int immune);
```

Description:

Sets if impacts are ever processed on this character. Useful for temporary effects and the like.

5.12.531 function diguyCharacter::get_is_immune_to_impacts**Prototype:**

```
int get_is_immune_to_impacts();
```

Returns:

1 if the character is immune to impacts 0 if not

Appearance Effect Functions

Appearance effects are additions to a character's appearance that can be turned on and off. These are commonly effect-type additions, such as flashing lights, smoke, etc.

Most appearance effects are implemented in some way for every character; for example, every character will respond to showing the "smoke" effect. Some appearance effects, however, don't make sense for certain characters; for example, emergency vehicles should usually be able to show "flashing_lights", but most humans cannot. Humans that are asked to show the "flashing_lights" effect will not do anything.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.532 function diguyCharacter::begin_appearance_effect**Prototype:**

```
int begin_appearance_effect(const char* effect,
    const char* override_link = NULL,
    float override_scale = DIGUY_DEFAULT_FLOAT,
    float override_offset_x = DIGUY_DEFAULT_FLOAT,
    float override_offset_y = DIGUY_DEFAULT_FLOAT,
    float override_offset_z = DIGUY_DEFAULT_FLOAT,
    float override_duration = DIGUY_DEFAULT_FLOAT);
```

Description:

This function starts the specified appearance effect on the character.

The following appearance effects are available:

- ◇ "brake_lights"
- ◇ "debris"
- ◇ "debris_dark"
- ◇ "dust"
- ◇ "engine_smoke"
- ◇ "explosion"
- ◇ "explosion_dark"
- ◇ "flames"
- ◇ "flashing_lights"
- ◇ "ground_dust_burst"
- ◇ "headlights"
- ◇ "rotor_wash"
- ◇ "smoke"
- ◇ "smoke_dark"
- ◇ "smoke_trail"

Arguments:

<code>effect</code>	name of appearance effect to be started
<code>override_link</code>	link to attach to; overrides default
<code>override_scale</code>	scale to use for effect; overrides default
<code>override_offset_x</code>	x offset to use for effect; overrides default
<code>override_offset_y</code>	y offset to use for effect; overrides default
<code>override_offset_z</code>	z offset to use for effect; overrides default
<code>override_duration</code>	overrides how long the effect should show before automatically ending

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.533 function diguyCharacter::end_appearance_effect

Prototype:

```
int end_appearance_effect(const char* effect,
    const char* override_link = NULL);
```

Description:

This function ends the specified appearance effect on the character.

The active particles of the effect will go to completion. Use [stop_all_appearance_effects\(\)](#) to immediately stop effects, including their active particles.

If the effect hasn't been started, nothing happens.

Arguments:

<code>effect</code>	name of appearance effect to be ended
<code>override_link</code>	optional override link effect is attached to

Pass "all" for the effect argument to end all of this character's appearance effects.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.534 function diguyCharacter::stop_appearance_effect

Prototype:

```
int stop_appearance_effect(const char* effect,
                           const char* override_link = NULL);
```

Description:

This function stops the specified appearance effect on the character.

The active particle systems of the effect will immediately be removed. Use [end_appearance_effect\(\)](#) to gradually stop effects; that function lets active particles move to completion.

If the effect hasn't been started, nothing happens.

Arguments:

<code>effect</code>	name of appearance effect to be ended
<code>override_link</code>	optional override link effect is attached to

Pass "all" for the effect argument to stop all of this character's appearance effects.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.535 function diguyCharacter::stop_all_appearance_effects

Prototype:

```
int stop_all_appearance_effects();
```

Description:

Equivalent to [stop_appearance_effect\("all"\)](#);

5.12.536 function diguyCharacter::appearance_effect_is_active**Prototype:**

```
int appearance_effect_is_active(const char* effect,
    const char* override_link = NULL);
```

Description:

This function checks to see if the specified appearance effect has been started with [begin_appearance_effect\(\)](#).

Arguments:

`effect` name of appearance effect to check
`override_link` optional override link effect is attached to

Returns:

1 if appearance is active, 0 if not

Chain Functions

These functions are for characters with character type "chain_simulation".

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.537 function diguyCharacter::get_chain_simulation

Prototype:

```
diguyChainSimulation* get_chain_simulation();
```

Description:

This function returns a pointer to a [diguyChainSimulation](#) if the character is a chain_simulation type of character.

Returns:

pointer to a [diguyChainSimulation](#)

Vehicle Functions

These functions are for some vehicle characters whose speeds and accelerations can be controlled interactively.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.538 function diguyCharacter::get_vehicle_controller**Prototype:**

```
diguyVehicleController* get_vehicle_controller();
```

Description:

This function returns a pointer to a [diguyVehicleController](#) if the character has one.

Returns:

pointer to a [diguyVehicleController](#); may return NULL if character does not have a vehicle controller object

5.12.539 function diguyCharacter::get_vehicle_body_orientation**Prototype:**

```
int get_vehicle_body_orientation(float* rz, float* rx, float* ry);
```

Description:

For most characters playing a travel forward action, the character's facing direction matches the direction of travel. For example, when humans walk forward their pelvis is generally aligned in the direction of travel.

Some vehicle characters behave differently. For vehicles whose front wheels steer, when turning the vehicle body will pivot around the rear axle. Only gradually will the rear of the vehicle swing in behind the direction of travel. Think of a car making a very sharp turn, and how the rear of the car swings in behind the front as the turn is completed.

This function will return the orientation of the body of the vehicle; i.e., the orientation of the forward vector along the vehicle's long axis. As mentioned above this may be different than the orientation of the direction of travel.

This function will return valid results for all characters, though the returned values from this function and that of `get_orientation()` will be the same for non-vehicle characters.

Returns:

0 on success, -1 on failure

Shader Functions

These functions set and return the shaders that are in use by characters.

Note that care should be taken when changing the shader of characters with skinned appearances. A shader capable of properly transforming vertices must be used for such characters.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.540 function diguyCharacter::set_render_mode_shader**Prototype:**

```
int set_render_mode_shader(const char* shader_name,
                           const char* render_mode = NULL);
```

Description:

This function sets the name of the shader program that will be used by the character when the scenario is using the specified render mode. See [diguyScenario::set_current_render_mode\(\)](#).

Note that care should be taken when changing the shader of characters with skinned appearances. A shader capable of properly transforming vertices must be used for such characters.

The shader with the given name should be loaded first. If the shader program is not one of DI-Guy's stock shaders, a custom shader program can be loaded explicitly by calling the [diguyApp::load_shader_program\(\)](#) function.

NOTE:

The specified shader program may not take effect immediately, if the scenario's current render mode is different than the passed render mode. See [diguyScenario::set_current_render_mode\(\)](#).

Arguments:

`shader_name` shader program name; can be NULL
`render_mode` render mode to modify; passing NULL will modify shader "normal" render mode

Returns:

0 on success, -1 on failure

5.12.541 function diguyCharacter::reset_render_mode_shader_to_default

Prototype:

```
void reset_render_mode_shader_to_default(const char* render_mode = NULL);
```

Description:

This function resets the shader program for the specified render mode to its default. This will be the shader program name returned by `diguyScenario::get_default_character_shader_program_name()`.

If the specified render mode is the current scenario render mode, the current shader will change immediately. If not, the change will not take effect until the specified render mode becomes the current scenario render mode. See [diguyScenario::set_current_render_mode\(\)](#).

Arguments:

`render_mode` render mode to modify; passing NULL will modify shader of "normal" render mode

5.12.542 function diguyCharacter::get_render_mode_shader

Prototype:

```
const char* get_render_mode_shader(const char* render_mode = NULL);
```

Description:

This function returns the name of the shader that will be used by the character when the scenario is using the specified render mode.

Arguments:

`render_mode` render mode for which to get program name; pass NULL for "normal" render mode

Returns:

name of shader program; NULL if none or error

5.12.543 function diguyCharacter::get_shader_program**Prototype:**

```
diguyGraphicsShaderProgram* get_shader_program();
```

Description:

This function returns the shader program that is in use by the character. Shader programs are shared objects that can be used by multiple characters.

Returns:

object of type [diguyGraphicsShaderProgram](#); NULL if none

5.12.544 function diguyCharacter::set_temporary_current_shader**Prototype:**

```
int set_temporary_current_shader(const char* shader_name);
```

Description:

This function overrides the current shader with the specified temporary shader. This temporary shader will remain in effect until:

- ◇ the scenario is reset
- ◇ [reset_current_shader_to_default\(\)](#) is called
- ◇ [set_temporary_current_shader\(\)](#) is called with NULL

Arguments:

`shader_name` shader name; can be NULL

Passing NULL for the shader name will un-set the temporary shader, resetting the render mode shader to its default.

Returns:

0 on success, -1 on failure

5.12.545 function `diguyCharacter::reset_current_shader_to_default`

Prototype:

```
void reset_current_shader_to_default();
```

Description:

This is a convenience function for resetting the current render mode shader to its default.

It is equivalent to retrieving the current render mode using `diguyScenario::get_current_render_mode()`, and calling `reset_render_mode_shader_to_default()` with that render mode.

5.12.546 function diguyCharacter::trigger_shader_refresh**Prototype:**

```
void trigger_shader_refresh();
```

Description:

This is a convenience function for refreshing the shader of a character. Usually used in conjunction with logic provided by the user with the graphics API.

5.12.547 function diguyCharacter::get_shader_instance**Prototype:**

```
diguyGraphicsShaderInstance* get_shader_instance();
```

Description:

This function returns the shader instance that is in use by the character. Each character has its own shader instance.

Note that the shader instance of a character can change over the lifetime of the character. For example, it will change if the appearance of the character changes. Therefore the returned pointer should not be stored, but should be re-read each time it is needed.

Returns:

object of type [diguyGraphicsShaderInstance](#); NULL if none

5.12.548 function**diguyCharacter::reset_shader_instance_variables_to_initial_values****Prototype:**

```
void reset_shader_instance_variables_to_initial_values();
```

Description:

This function resets current values of the variables of the shader back to their initial reset values.

5.12.549 function**diguyCharacter::reset_shader_instance_variables_to_program_defaults****Prototype:**

```
void reset_shader_instance_variables_to_program_defaults();
```

Description:

This function resets the current and initial values of variables of the shader to the built-in DI-Guy configuration defaults.

DI-Guy AI and Agent Functions

The following functions are for characters that are controlled by DI-Guy AI. AI-controlled characters are referred to as agents.

Almost all functions in this section require that the agent character be part of a crowd.

A DI-Guy AI license must be present for these functions to work.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

Attack Functions

5.12.550 function diguyCharacter::agent_attack_character

Prototype:

```
int agent_attack_character(const char* character_name);
```

Description:

This function sets the named character as the attack target of this agent. This agent will pursue the target until it is within range, and then will fire its weapon in an attempt to kill the target.

This function will implicitly put this agent into the attack behavior.

The current focus character will be set the specified character. Pass NULL or the empty string ("") to not change the focus character, but still change to the pursue behavior.

The current focus group will be un-set.

Arguments:

`character_name` character to attack

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.551 function `diguyCharacter::agent_attack_group`

Prototype:

```
int agent_attack_group(const char* group_name);
```

Description:

This function is similar to [agent_attack_character \(\)](#), but instead will choose attack targets from the characters in the specified group.

This function will implicitly put this agent into the attack behavior.

The current focus group will be set to the specified group. Pass NULL or the empty string ("") to not change the focus group, but still change to the attack behavior.

The current focus character will be un-set.

Arguments:

`group_name` name of group from which attack targets should be chosen

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.552 function `diguyCharacter::agent_set_attack_zone_enabled`

Prototype:

```
int agent_set_attack_zone_enabled(int use_attack_zone);
```

Description:

See [diguyAgentParams::set_attack_zone_enabled\(\)](#).

5.12.553 function `diguyCharacter::agent_get_attack_zone_enabled`

Prototype:

```
int agent_get_attack_zone_enabled();
```

Description:

See `diguyAgentParams::get_attack_zone_enabled()`.

5.12.554 function `diguyCharacter::agent_set_attack_zone`

Prototype:

```
int agent_set_attack_zone(float x, float y, float z, float radius);
```

Description:

See `diguyAgentParams::set_attack_zone()`.

5.12.555 function diguyCharacter::agent_get_attack_zone**Prototype:**

```
int agent_get_attack_zone(float* x, float* y, float* z, float* radius);
```

Description:

See `diguyAgentParams::get_attack_zone()`.

Flee Functions**5.12.556 function diguyCharacter::agent_flee_character****Prototype:**

```
int agent_flee_character(const char* character_name,
    float distance,
    int danger_level = 50);
```

Description:

This function adds a flee object to this agent. This agent will attempt to move away from the specified character until it is beyond the specified flee distance.

Note: Currently there can only be one flee object. This restriction will be lifted in a future version.

This function will implicitly put this agent into the flee behavior.

Neither the current focus character nor group will be changed by this function.

Arguments:

`character_name` character from which to flee
`distance` minimum distance to move away from character
`danger_level` how "strong" the flee should be; set from 0 to 100

The `danger_level` argument is not currently used, but will be in a future version.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.557 function `diguyCharacter::agent_flee_group`

Prototype:

```
int agent_flee_group(const char* group_name,
                    float distance,
                    int danger_level = 50);
```

Description:

This function adds a flee object to this agent. This agent will attempt to move away from the specified group until it is beyond the specified flee distance.

This function will implicitly put this agent into the flee behavior.

Neither the current focus character nor group will be changed by this function.

Arguments:

`group_name` group from which to flee
`distance` minimum distance to move away from character

`danger_level` how "strong" the flee should be; set from 0 to 100

The `danger_level` argument is not currently used, but will be in a future version.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.558 function `diguyCharacter::agent_flee_crowd`

Prototype:

```
int agent_flee_crowd(const char* crowd_name,
                    float distance,
                    int danger_level = 50);
```

Description:

This function is similar to [agent_flee_group\(\)](#), but the agent will instead flee the members of the specified crowd.

5.12.559 function diguyCharacter::agent_flee_location**Prototype:**

```
int agent_flee_location(float x, float y, float z,
    float distance,
    int danger_level = 50);
```

Description:

This function adds a flee object to this agent. This agent will attempt to move away from the specified location until it is beyond the specified flee distance.

Note: Currently there can only be one flee object. This restriction will be lifted in a future version.

This function will implicitly put this agent into the flee behavior.

Neither the current focus character nor group will be changed by this function.

Arguments:

<code>x, y, z</code>	location from which to flee
<code>distance</code>	minimum distance to move away from character
<code>danger_level</code>	how "strong" the flee should be; set from 0 to 100

The `danger_level` argument is not currently used, but will be in a future version.

Returns:

0 on success, -1 on failure

5.12.560 function `diguyCharacter::agent_remove_all_flee_objects`

Prototype:

```
void agent_remove_all_flee_objects();
```

Description:

This function removes all flee objects.

This function has no effect on current behavior.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

Mingle Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.561 function diguyCharacter::agent_mingle_in_region

Prototype:

```
int agent_mingle_in_region(const char* region_name = "(current)",
    const char* subregion_name = "base");
```

Description:

This function puts this agent into the mingle behavior, in which it will choose semi-random points in the specified region and subregion to move to. The agent will remain at the selected points for some amount of time, and then select a new point.

If all members of a crowd have this behavior they will tend to get together in small groups.

Use `diguyAgentParams::set_move_on_t_min()` and `diguyAgentParams::set_move_on_t_max()` for setting how long the agent will remain at the selected mingle points.

Arguments:

`region_name` region in which to mingle
`subregion_name` subregion in which to mingle

The behavior region will be set the specified region. Pass NULL or "(current)" to not change the current behavior region.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

Wander Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.562 function diguyCharacter::agent_wander_region

Prototype:

```
int agent_wander_region(const char* region_name = "(current)",
    const char* subregion_name = "base");
```

Description:

This function is similar to [agent_mingle_in_region\(\)](#), but agents will pick completely random points and not try to clump together in small groups.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

Pursue Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.563 function diguyCharacter::agent_pursue_character**Prototype:**

```
int agent_pursue_character(const char* character_name);
```

Description:

This function will cause this agent to pursue the specified character. This agent will attempt to move toward the specified character until it is within its still distance as specified in its agent params.

This function will implicitly put this agent into the pursue behavior.

The current focus group will be set to the specified group. Pass NULL or the empty string ("") to not change the focus group, but still change to the attack behavior.

The current focus character will be un-set.

Arguments:

`character_name` character to pursue

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.564 function diguyCharacter::agent_pursue_character_with_offset

Prototype:

```
int agent_pursue_character_with_offset(const char* character_name,
    float offset_x,
    float offset_y);
```

Description:

This function is similar to [agent_pursue_character \(\)](#), but the point toward the agent will move will be offset from the pursued character by the passed offset.

5.12.565 function diguyCharacter::agent_pursue_group

Prototype:

```
int agent_pursue_group(const char* group_name);
```

Description:

This function is similar to [agent_pursue_character \(\)](#), but instead will choose pursue characters from the characters in the specified group.

This function will implicitly put this agent into the pursue behavior.

The current focus group will be set to the specified group. Pass NULL or the empty string ("") to not change the focus group, but still change to the pursue behavior.

The current focus character will be un-set.

Arguments:

group_name name of group from which pursue characters should be

chosen

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.566 function diguyCharacter::agent_call_formation

Prototype:

```
int agent_call_formation(const char* formation_name,
    float target_x,
    float target_y);
```

Description:

Calls a formation with this character as a leader, oriented towards (target_x, target_y).

Returns:

0 on success, -1 on failure

Travel Functions

Unless otherwise specified, all functions callable from:

- ◆ C++

◆ Script

5.12.567 function diguyCharacter::agent_travel_behavior_path**Prototype:**

```
int agent_travel_behavior_path(const char* direction = NULL,
    const char* at_end = NULL,
    int waypoint_index = -1,
    float offset_y = 0.5f);
```

Description:

This function has the character travel along its behavior path shape, typically to the end and then back to the beginning. It will put the agent into the travel behavior.

Arguments:

<code>direction</code>	should be either "forward" or "backward"; pass NULL for default forward
<code>at_end</code>	what to do when end of path reached
<code>waypoint_index</code>	index of waypoint to start at; -1 for nearest, -2 for random

`offset_y` sideways offset from behavior path

Valid values for `at_end` are:

- ◇ "reverse"
- ◇ "loop"
- ◇ "stop"
- ◇ "teleport"

See the documentation for [diguyNavPathAtEnd](#) for information on what each value means.

5.12.568 function `diguyCharacter::agent_travel_path_shape`

Prototype:

```
int agent_travel_path_shape(const char* path_shape_name,
    const char* direction = NULL,
    const char* at_end = NULL,
    int waypoint_index = -1,
    float offset_y = 0.5f);
```

Description:

Similar to [agent_travel_behavior_path\(\)](#), but specific path can be set rather than using behavior path.

5.12.569 function diguyCharacter::agent_travel_region_border**Prototype:**

```
int agent_travel_region_border(const char* region_name = NULL,
                             const char* subregion_name = "base");
```

Description:

The agent will create a travel path around the border of the region specified. A `CALLBACK_ID_AGENT_TRAVEL_FORWARD_DEST_REACHED` callback will be triggered every time a character completes a loop around the border.

Arguments:

<code>region_name</code>	the name of the region to patrol; leaving blank or passing NULL will use the agent's current region
<code>subregion</code>	name of the subregion to use; defaults to the base subregion

Returns:

0 on success, -1 on failure

5.12.570 function diguyCharacter::agent_move_to_point**Prototype:**

```
int agent_move_to_point(float x, float y, float z,
                        const char* via_region = NULL,
                        const char* subregion_name = "base");
```

Description:

Agent will attempt to create a navigation path on the `via_region` and then travel it. An A* path planning algorithm is used to find the path, cost preferences from the character's current [diguyAgentParams](#) are used in the search.

Note: as of DI-Guy 12 there is a version of this function

[agent_move_to_point_bg\(\)](#) that uses a multi-threaded infrastructure. We strongly recommend using it for future developments.

Arguments:

<code>x, y, z</code>	the target location

via_region the name of the region to run A* on (pass in "" to have the code attempt to find the nearest region automatically.)

subregion_name the name of the subregion to restrict search to

Returns:

0 on success, -1 on failure

5.12.571 function diguyCharacter::agent_move_to_point_bg

Prototype:

```
int agent_move_to_point_bg(float x, float y, float z,
    const char* via_region = NULL,
    const char* subregion_name = "base");
```

Description:

A multi-threaded version of [agent_move_to_point\(\)](#). See that function for more information.

We recommend switching to using this over the single threaded version. When the path result has been computed a callback with id CALLBACK_ID_AGENT_PATH_PLAN_COMPLETED is triggered and the result is stored in [get_path_planning_result\(\)](#). You can also poll for completion via [get_is_path_planning\(\)](#).

Returns:

[diguyNavPathSearchResults](#) enum value

The return value is typically `DIGUY_NAV_PATH_ERROR_NONE` (0) on success, `DIGUY_NAV_PATH_ERROR_SEARCH_QUEUED` if this has been queued for multi-threading, or a negative value on failure.

5.12.572 function

`diguyCharacter::agent_move_to_point_via_subregions`

Prototype:

```
int agent_move_to_point_via_subregions(float x, float y, float z,
    const char* via_region = NULL,
    int preferred_subregions_mask = DIGUY_SUBREGION_MASK_NONE,
    float cost_bias_for_preferred_regions = 0.25f,
    float cost_bias_for_neutral_regions = 1.1f,
    int repulsed_regions_mask = DIGUY_SUBREGION_MASK_NONE,
    float cost_bias_for_repulsed_regions = 10.0f,
    int cost_rule = 0);
```

Description:

Agent will attempt to create a navigation path on the region `via_region` and then travel it. An A* path planning algorithm is used to find the path.

The subregion masks are used to specify which subregions to use or avoid. Cost bias values can be used to make tweak how strong the repulsion values are.

Note:

DI-Guy SDK Version 12.5.2 Index

As of DI-Guy 12 there is a version of this function `agent_move_to_point_via_subregions_bg` that uses a multi-threaded infrastructure. We strongly recommend using it for future developments.

This function will fail and return -1 if a path cannot be found.

Arguments:

<code>x, y, z</code>	the target location
<code>via_region</code>	the name of the region to run A* on (pass in "*" to have the code attempt to find the nearest region automatically)
<code>preferred_subregions_mask</code>	diguySubregionMask value of regions preferred for travel
<code>cost_bias_for_preferred_regions</code>	how much cheaper it will be to cross spaces that are part of desired subregion; should be < 1.0
<code>cost_bias_for_neutral_regions</code>	how much more expensive it will be to cross spaces that are not part of desired subregion; see below for more information
<code>repulsed_regions_mask</code>	diguySubregionMask value of regions not preferred for travel
<code>cost_bias_for_repulsed_regions</code>	how much more expensive it will be to cross regions that are marked as repulsive; pass <code>DIGUY_DEFAULT_FLOAT</code> to avoid completely

Pass [DIGUY_DEFAULT_FLOAT](#) for `cost_bias_for_neutral_regions` and `cost_bias_for_repulsed_regions` to specify that they should be avoided completely.

The `preferred_subregions_mask` and `repulsed_regions_mask` use DI-Guy subregion mask values combined together. For example, [DIGUY_SUBREGION_MASK_SIDEWALK](#) | [DIGUY_SUBREGION_MASK_CROSSWALK](#).

For `cost_bias_for_neutral_regions`, specifying a non-zero value for this lowers the likelihood that searches fail on disjointed subregions. A value < 1 will cause the planner to run faster but explore fewer points, possibly missing preferred regions. Values greater than 1 will explore more points but will be slower.

NOTE:

THIS API has changed for DI-GUY 11.

`cost_bias_for_preferred_regions` has been added and the neutral and repulsed default values have changed.

Returns:

0 on success, -1 on failure

5.12.573 function `diguyCharacter::agent_move_to_point_via_subregions_bg`

Prototype:

```
int agent_move_to_point_via_subregions_bg(float x, float y, float z,
    const char* via_region = NULL,
    int preferred_subregions_mask = DIGUY_SUBREGION_MASK_NONE,
    float cost_bias_for_preferred_regions = 0.25f,
    float cost_bias_for_neutral_regions = 1.1f,
    int repulsed_regions_mask = DIGUY_SUBREGION_MASK_NONE,
    float cost_bias_for_repulsed_regions = 10.0f,
    int cost_rule = 0);
```

Description:

A multi-threaded version of `agent_move_to_point_via_subregions()`. We recommend switching to using this over the single threaded version. When the result is found a `CALLBACK_ID_AGENT_PATH_PLAN_COMPLETED` is triggered and the result is stored in `get_path_planning_result()`, you can also poll for completion via `get_is_path_planning()`.

Returns:

`diguyNavPathSearchResults` enum value

The return value is typically `DIGUY_NAV_PATH_ERROR_NONE` (0) on success, `DIGUY_NAV_PATH_ERROR_SEARCH_QUEUED` if this has been queued for multi-threading, or a negative value on failure.

5.12.574 function `diguyCharacter::get_path_planning_result`

Prototype:

```
int get_path_planning_result();
```

Returns:

`diguyNavPathSearchResults` value after `agent_move_to_point_bg( )` or `agent_move_to_point_via_subregions_bg( )` has been called when the multi-threaded path planner has completed.

5.12.575 function diguyCharacter::get_is_path_planning**Prototype:**

```
int get_is_path_planning();
```

Returns:

1 if character is currently waiting for a result from the multi-threaded path planner, else 0

5.12.576 function diguyCharacter::agent_move_to_region**Prototype:**

```
int agent_move_to_region(const char* region,
    const char* subregion_name = "base",
    const char* via_region = NULL);
```

Description:

Agent will attempt to create a navigation path on the via_region to a point in the target region, and then travel it. An A* path planning algorithm is used to find the path. Cost preferences from the character's current [diguyAgentParams](#) are used in the search.

Arguments:

region	target region
subregion_name	target subregion
via_region	name of the region travel in

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.577 function diguyCharacter::agent_move_to_region_via_subregions

Prototype:

```
int agent_move_to_region_via_subregions(const char* region_name,
    const char* subregion_name = "base",
    const char* via_region = NULL,
    float cost_bias_for_preferred_regions = 0.25f,
    int preferred_subregions_mask = DIGUY_SUBREGION_MASK_NONE,
    float cost_bias_for_neutral_regions = 1.1f,
    int repulsed_regions_mask = DIGUY_SUBREGION_MASK_NONE,
    float cost_bias_for_repulsed_regions = 10.0f);
```

Description:

Similar to [agent_move_to_point_via_subregions\(\)](#), but for moving to another region.

NOTE:

THIS API has changed for DI-GUY 11.

`cost_bias_for_preferred_regions` has been added and the neutral and repulsed default values have changed.

See Also:

```
diguyScenario::set_max_distance_planning_multiplier();
diguyScenario::get_time_spent_path_planning_this_frame();
```

Returns:

0 on success, -1 on failure

Path Follow Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.578 function `diguyCharacter::agent_path_follow`

Prototype:

```
int agent_path_follow(const char* path_name);
```

Description:

The agent will begin following the specified path, using the path follow behavior (`DIGUY_AGENT_BEHAVIOR_PATH_FOLLOW`).

The path follow behavior allows an agent to approximately follow a `diguyCharacterPath` without being in path position mode. The agent can respond to external stimuli (e.g., run away from gunfire), and later return to and resume the path.

Currently the path must be a path previously defined for the character, through either DI-Guy Scenario or via path API calls.

Arguments:

`path_name` name of a path of this character

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.579 function diguyCharacter::agent_resume_path_follow

Prototype:

```
int agent_resume_path_follow();
```

Description:

The agent will resume following the path specified by a previous call to [agent_path_follow\(\)](#). This will put the agent back into the path follow behavior.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.580 function diguyCharacter::agent_get_path_follow_path**Prototype:**

```
diguyCharacterPath* agent_get_path_follow_path();
```

Returns:

the path being followed in path behavior mode

5.12.581 function diguyCharacter::agent_get_path_follow_path_t**Prototype:**

```
float agent_get_path_follow_path_t();
```

Description:

In path follow behavior the agent moves along the path specified in [agent_path_follow\(\)](#). How far along the agent is on the path is specified in seconds. This function returns that time.

Note that advancing along the path may be paused for some reason, typically because the agent is resuming path follow behavior after being in a different behavior, or the agent is too far away from the current desired position on the path. In this case the returned time may not advance for awhile until the agent is close enough to the path to resume advancing along it.

Call

```
diguyAgentParams::set\_path\_follow\_pause\_t\_when\_off\_path\(\)
```

to set whether time will be paused when off of the path.

Call

`diguyAgentParams::set_path_follow_unpause_t_at_radius()`

to set how close the agent must be to the last point on the path before time will resume.

Returns:

how far, in seconds, the agent is along its path follow path

5.12.582 function

`diguyCharacter::agent_get_path_follow_path_t_paused`

Prototype:

```
int agent_get_path_follow_path_t_paused();
```

Returns:

1 if time along the path is not advancing, else 0

Other Agent Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.583 function diguyCharacter::agent_get_crowd**Prototype:**

```
diguyCrowd* agent_get_crowd();
```

Returns:

the crowd this agent is a part of, if any; returns NULL if not in a crowd

5.12.584 function diguyCharacter::agent_begin_behavior**Prototype:**

```
int agent_begin_behavior(const char* behavior_name);
```

Description:

This function allows the end user to directly change an agent's current behavior. This function currently the only way to start an agent "idling".

5.12.585 function diguyCharacter::agent_stop_behavior**Prototype:**

```
int agent_stop_behavior();
```

Description:

This function sets the behavior of the agent to "none".

5.12.586 function diguyCharacter::agent_set_current_focus_character**Prototype:**

```
int agent_set_current_focus_character(const char* character_name);
```

Description:

This function is a convenience function that changes the focus character of this agent's current crowd params.

This can also be done by getting the character's current params using `get_agent_current_params( )`, and then calling `set_focus_character( )` on that object. This function call, however, is available in decisions.

The agent currently need not be part of a crowd for this function to work.

Arguments:

`character_name` name of character to become new focus character;
pass NULL or "" to un-set the focus character

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.587 function diguyCharacter::agent_get_current_focus_character

Prototype:

```
diguyCharacter* agent_get_current_focus_character();
```

Description:

This function returns the current character this agent is focused on. This will either be the character named by the focus character, or the character chosen from the focus group to be the current focus character.

The agent currently need not be part of a crowd for this function to work.

pointer to type [diguyCharacter](#)

5.12.588 function diguyCharacter::agent_set_current_focus_group

Prototype:

```
int agent_set_current_focus_group(const char* group_name);
```

Description:

Similar to `set_current_focus_character( )`, but sets the current focus group instead of character.

The agent currently need not be part of a crowd for this function to work.

Arguments:

`group_name` name of group to become new focus group; pass NULL or "" to un-set the focus group

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.589 function diguyCharacter::agent_get_initial_params**Prototype:**

```
diguyAgentParams* agent_get_initial_params();
```

Description:

This function returns the initial agent parameters of this agent. The initial parameters of this agent can then be edited through this object.

Changes to the initial parameters will have no effect until the scenario is reset.

The current agent parameters will be set to these initial parameters whenever the scenario is reset.

The agent currently need not be part of a crowd for this function to work.

pointer to type [diguyAgentParams](#)

5.12.590 function diguyCharacter::agent_get_current_params**Prototype:**

```
diguyAgentParams* agent_get_current_params();
```

Description:

This function returns the current agent parameters of this agent. The current parameters of this agent can then be edited through this object.

Changes to the current parameters should take effect immediately.

The current agent parameters will be reset to the initial parameters whenever the scenario is reset.

The agent currently need not be part of a crowd for this function to work.

pointer to type [diguyAgentParams](#)

5.12.591 function**diguyCharacter::agent_get_current_params_profile_name****Prototype:**

```
const char* agent_get_current_params_profile_name();
```

Description:

This function gets the name of the agent's current profile.

Returns:

the name of the agent's current profile

5.12.592 function**diguyCharacter::agent_set_current_params_from_profile****Prototype:**

```
int agent_set_current_params_from_profile(const char* profile_name,
    int retain_focus_objects = 1,
    int retain_navigation_values = 1,
    int retain_behavior = 0);
```

Description:

This function sets this agent's current agent parameters from the specified crowd profile.

The agent currently need not be part of a crowd for this function to work.

Arguments:

<code>profile_name</code>	name of existing crowd profile
<code>retain_focus_objects</code>	pass 1 to not change focus object from current focus

`retain_navigation_values` pass 1 to not change current navigation parameters
`retain_behavior` pass 1 to not change current behavior

Navigation values are things like:

- ◇ behavior path
- ◇ behavior region and subregion
- ◇ region masks
- ◇ cost biases
- ◇ path planner settings

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.593 function `diguyCharacter::agent_set_initial_params_from_profile`

Prototype:

```
int agent_set_initial_params_from_profile(const char* profile_name);
```

Description:

Similar to `agent_set_current_params_from_profile()`, but initial parameters are changed.

5.12.594 function**diguyCharacter::agent_set_current_params_from_initial_params****Prototype:**

```
void agent_set_current_params_from_initial_params();
```

Description:

This function sets the current parameters from the agent's initial parameters.

5.12.595 function**diguyCharacter::agent_set_current_behavior_region_border_is_s****Prototype:**

```
void agent_set_current_behavior_region_border_is_solid(int is_solid);
```

Description:

This function sets whether the borders of the current behavior region of the agent is solid.

If they are, the agent will treat the walls as if they are solid walls in the environment. If not, the agent is free to walk through them.

The agent currently need not be part of a crowd for this function to work.

Arguments:

`is_solid` 1 if region border should be solid, 0 if not

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.596 function diguyCharacter::agent_move_on

Prototype:

```
void agent_move_on();
```

Description:

If the agent is in a behavior that stays in one place before moving to another, such as wander or mingle, this function will request that the agent move on to the next location.

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.12.597 function**diguyCharacter::agent_set_auto_variant_selection****Prototype:**

```
int agent_set_auto_variant_selection(int selection_enabled);
```

Description:

This function controls if the AI will attempt to automatically change variants depending on the behavior. This should only change how still motions are performed when the default variant is Normal. Characters in wander and idle will automatically shift to ambient still motions, mingle will alternate between socialize and ambient motions. Attack should automatically switch from aiming to ambient. defaults to being on.

5.12.598 function**diguyCharacter::agent_get_auto_variant_selection****Prototype:**

```
int agent_get_auto_variant_selection();
```

Description:

Returns if automatic variant selection is enabled, defaults to being on.

User-Defined Feelers

Feelers are effectively ray traces that an agent uses to detect intersections with obstacles in the environment.

In DI-Guy Scenario, DI-Guy AI has a built-in feeler function that allows agents to detect collisions with the environment. In a DI-Guy SDK application, a user-defined feeler function must be supplied.

User-defined functions can be added to individual characters by calling

`agent_set_feeler_function()`, or to all characters by calling `set_default_character_feeler_function()`.

The following prototype should be used for the feeler functions:

```
int diguyFeelerFunction(diguyCharacter* character,
    float origin_x,
    float origin_y,
    float origin_z,
    float dir_x,
    float dir_y,
    float dir_z,
    float max_distance_to_test,
    float* intersection_x,
    float* intersection_y,
    float* intersection_z,
    float* normal_x,
    float* normal_y,
    float* normal_z);
```

Arguments:

`origin_x, origin_y, origin_z` starting location of the feeler ray

`dir_x, dir_y, dir_z` direction of feeler ray

`max_distance_to_test` max distance feeler will test

`intersection_x, _y, and _z` world location of intersection

`normal_x, normal_y, normal_z` normal of intersection surface

Smaller values for `max_distance_to_test` will result in better performance.

The location of the intersection point is returned in the intersection x, y, and z pointers.

The normal of the surface intersected is returned in the normal x, y, and z pointers. The normal is needed for agents using feelers to know how to move around and avoid the detected surface.

Returns:

1 if intersection occurred, 0 if not

5.12.599 function diguyCharacter::agent_set_feeler_function**Prototype:**

```
int agent_set_feeler_function(diguyFeelerFunction* feeler_function);
```

Description:

This function sets a feeler function that will be used by this character for environment collision detection. The feeler will replace any existing feeler function (including the default function used in DI-Guy Scenario).

See the "User-Defined Feelers" information for more information.

Arguments:

`feeler_function` pointer to user-defined feeler function

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.12.600 function diguyCharacter::agent_get_feeler_function**Prototype:**

```
diguyFeelerFunction* agent_get_feeler_function();
```

Returns:

current feeler function; see `set_feeler_function( )`

Callable From:

◇ C++

Lua Mind Functions

5.12.601 function diguyCharacter::create_lua_mind**Prototype:**

```
int create_lua_mind(const char* base_class, const char* init_arg1 = NULL,  
    const char* init_arg2 = NULL);
```

Description:

Creates a Lua mind object based off of the contents of base_class.

Arguments:

base_class the class to use as the basis for the character's mind
init_arg1-2 experimental optional arguments passed to the init
function; your lua init function will need to take arguments

5.12.602 function diguyCharacter::get_mind_name**Prototype:**

```
const char *get_mind_name();
```

Description:

Returns the name of the mind, e.g. "luaPedestrian". This is, in other words,
the mind's base class

5.12.603 function diguyCharacter::get_mind_current_state_name**Prototype:**

```
const char* get_mind_current_state_name();
```

Description:

Agent characters with minds will typically have the mind be in some state, such as "patrolling", "fleeing", "mobbing", etc. This function returns the name of that state.

Currently this is only applicable to characters with Lua minds.

Returns:

the name of the agent's current mind state

5.12.604 function diguyCharacter::do_mind_current_state_changed_callback**Prototype:**

```
void do_mind_current_state_changed_callback();
```

Description:

This function causes callback functions with the callback id `CALLBACK_ID_AGENT_MIND_CURRENT_STATE_CHANGED` to be called. It's intended use is for mind implementations that need to signal that the mind's current state has changed.

Currently this is only applicable to characters with Lua minds.

5.12.605 function diguyCharacter::set_sleep_timer**Prototype:**

```
int set_sleep_timer(float sleep_length);
```

Description:

How long until the character's main coroutine thread should be resumed. This function is typically used by various luaCharacter sleep functions.

Arguments:

`sleep_length` how long relative to the current time the character should sleep

5.12.606 function diguyCharacter::get_sleep_timer**Prototype:**

```
float get_sleep_timer();
```

Returns:

How long until the character's main coroutine thread resumes, relative to the current time.

5.12.607 function diguyCharacter::set_heartbeat_dt**Prototype:**

```
int set_heartbeat_dt(float heartbeat_dt);
```

Description:

The frequency at which the luaCharacter's heartbeat function should run. This defaults to once per second. The character must have a Lua function named heartbeat.

Arguments:

heartbeat_dt frequency of heartbeat

5.12.608 function diguyCharacter::add_secondary_heartbeat**Prototype:**

```
int add_secondary_heartbeat(const char* function_name,
float heartbeat_dt);
```

Description:

This function allows the addition of additional heartbeat functions.

Arguments:

function_name the name of the Lua function to treat as a heartbeat; the function_name should not include the name of the character object

heartbeat_dt frequency of heartbeat

Returns:

0 on success, -1 on failure

5.12.609 function diguyCharacter::remove_secondary_heartbeat**Prototype:**

```
int remove_secondary_heartbeat(const char* function_name);
```

Description:

This function removes an added heartbeat function.

Arguments:

function_name the name of the Lua function to treat as a heartbeat; the function_name should not include the name of the character object

Returns:

0 on success, -1 on failure

5.12.610 function diguyCharacter::set_secondary_heartbeat_dt**Prototype:**

```
int set_secondary_heartbeat_dt(const char* function_name, float heartbeat_dt);
```

Description:

The frequency at which a luaCharacter's secondary heartbeat function should run. The character must have a Lua function named function_name.

Arguments:

function_name heartbeat function to adjust
heartbeat_dt frequency of heartbeat

Returns:

0 on success, -1 on failure

5.12.611 function diguyCharacter::add_wakeup_callback**Prototype:**

```
int add_wakeup_callback(int callback);
```

Description:

This function creates a request that whenever the specified callback is triggered, the character is woken up with a message. The message will be of the form:

- message_type = "callback"
- message = the string representation of the callback

A good example of the use for this is a flexible "move to" command:

Lua Example:

```
self.character:add_wakeup_callback(diguyCharacter_CALLBACK_ID_AGENT_TRAVEL_FORWARD_
-- begin to move to a location
self.character:agent_move_to_region_via_subregions("navmesh", "green", "navmesh",
DIGUY_SUBREGION_MASK_BASE, 12);
-- sleep till arrived
self:sleep(100);
self.character:remove_wakeup_callback(diguyCharacter_CALLBACK_ID_AGENT_TRAVEL_FORWA
```

Effectively the character will wake up either after 100 seconds or when they arrive at their desired destination.

Arguments:

`callback` the callback id that should wake up the character

Returns:

0 on success, -1 on failure

5.12.612 function diguyCharacter::remove_wakeup_callback**Prototype:**

```
int remove_wakeup_callback(int callback);
```

Description:

This function removes a request to trigger a message when a callback is triggered.

Arguments:

`callback` callback id that should no longer wake up the character

Returns:

0 on success, -1 on failure

5.12.613 function diguyCharacter::remove_all_wakeup_callbacks**Prototype:**

```
int remove_all_wakeup_callbacks();
```

Description:

This function removes all requests to trigger a message when callbacks are triggered.

Returns:

0 on success, -1 on failure

5.12.614 function diguyCharacter::has_wakeup_callback**Prototype:**

```
int has_wakeup_callback(int callback);
```

Description:

This function check for a requests to trigger a message when a given callback is triggered.

Returns:

0 on false, 1 on true

5.12.615 function diguyCharacter::agent_accept_message**Prototype:**

```
int agent_accept_message(const char* sender,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);
```

Description:

This function wakes up a character and sends them a message. Currently this is only applicable to characters with Lua minds.

Arguments:

sender	who is sending the message
message_type	what type of message is being sent
message	the message string
message_params	optional parameter string

Returns:

number of times message was successfully parsed

5.12.616 function diguyCharacter::agent_broadcast_message

Prototype:

```
int agent_broadcast_message(float radius,  
    const char* message_type,  
    const char* message,  
    const char* message_params = NULL);
```

Description:

This function wakes up nearby characters and sends them a message. The sender of the message is implicitly set to the individual calling the broadcast function

Currently this is only applicable to characters with Lua minds.

Arguments:

radius	how far to broadcast the message
message_type	what type of message is being sent
message	the message string

message_params optional parameter string

Returns:

number of times message was successfully parsed

5.12.617 function diguyCharacter::agent_broadcast_message_to_group

Prototype:

```
int agent_broadcast_message_to_group(const char* group_name,
    float radius,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);
```

Description:

This function wakes up nearby characters in the group specified and sends them a message. The sender of the message is implicitly set to the individual calling the broadcast function.

Currently this is only applicable to characters with Lua minds.

Arguments:

group_name group to broadcast message to

radius	how far to broadcast the message
message_type	what type of message is being sent
message	the message string
message_params	optional parameter string

Returns:

number of times message was successfully parsed, -1 on failure

5.12.618 function diguyCharacter::get_lua_object_name

Prototype:

```
const char* get_lua_object_name();
```

Returns:

name of character's lua mind object; typically this is the character's name with '-' replaced by '_'

5.12.619 function diguyCharacter::set_mind_field_as_string**Prototype:**

```
void set_mind_field_as_string(const char* field_name,  
                             const char* field_value);
```

Description:

Sets the specified field of the mind to the string field_value.

5.12.620 function diguyCharacter::get_mind_field_as_string**Prototype:**

```
const char* get_mind_field_as_string(const char* field_name,  
                                     int warn_if_no_field = 1);
```

Description:

Queries the luaCharacter object for a specific field, and attempts to convert it into a string. This works for strings and numbers.

This returned pointer will stay valid until the next time this function is called, so if the results need to be saved they must be copied.

Returns:

field_name as string

5.12.621 function diguyCharacter::set_mind_field_as_float**Prototype:**

```
void set_mind_field_as_float(const char* field_name, float field_value);
```

Description:

Sets the specified field of the mind to the floating point number field_value.

5.12.622 function diguyCharacter::get_mind_field_as_float**Prototype:**

```
float get_mind_field_as_float(const char* field_name,  
    int warn_if_no_field = 1,  
    int* found_field = NULL);
```

Description:

Similar to [get_mind_field_as_string\(\)](#), but returns field as a float.

Returns:

field_name as double

5.12.623 function diguyCharacter::set_mind_field_as_int**Prototype:**

```
void set_mind_field_as_int(const char* field_name, int field_value);
```

Description:

Sets the specified field of the mind to the integer number field_value.

5.12.624 function diguyCharacter::get_mind_field_as_int**Prototype:**

```
int get_mind_field_as_int(const char* field_name,  
    int warn_if_no_field = 1,  
    int* found_field = NULL);
```

Description:

Similar to [get_mind_field_as_int\(\)](#), but returns field as an integer.

Returns:

field_name as int

5.12.625 function diguyCharacter::evaluate_mind_function

Prototype:

```
const char* evaluate_mind_function(const char* function_name,
    const char* argument = NULL,
    int has_return_string = 0);
```

Description:

Attempts to run a luaCharacter member function, and can optionally return the function's result. Lua return values can be converted to strings and numbers -- other data types will cause an error.

This returned pointer will stay valid until the next time this function is called, so if the results need to be saved they must be copied.

***Note:** This function should be considered experimental, as the

API may change.

Arguments:

function_name	luaCharacter function field to invoke
argument	optional string argument to the function
has_return_string	0 or 1; determines if lua interface should attempt to retrieve a string value

Returns:

Lua function return value

5.12.626 function diguyCharacter::get_ai_state_string**Prototype:**

```
const char* get_ai_state_string(int include_stack = 1, int include_locals = 1);
```

Description:

This function produces a string with a long list of character information, including: current path, crowd, behavior, lua state, lua call stack, wakeup callbacks and lua object fields. It is meant to help with debugging agents.

This returned pointer will stay valid until the next time this function is called, so if the results need to be saved they must be copied.

Arguments:

`include_stack` pass 1 to include callstack info, 0 to not

`include_locals` pass 1 to include local variable info, 0 to not

Returns:

string containing character information

Visibility Functions

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

Visibility Tests

5.12.627 function diguyCharacter::check_character_visibility**Prototype:**

```
int check_character_visibility(diguyCharacter* character,
    int los_check_type = 0);
```

Description:

This function tests whether this character can see the passed character. Currently this tests if the cervical link of this character has a clear line of sight to either the cervical or the position links of the target. Currently the orientation of this character isn't factored in.

This function does not fail if other characters are between the two characters in question.

Checking visibility is only an option if an octtree is generated from scene objects; typically only DI-Guy Scenario does this. Outside of DI-Guy Scenario [set_character_los_function\(\)](#) should be used if no octtree is present.

Arguments:

<code>character</code>	character to check
<code>los_check_type</code>	currently unused but passed to callback function if end users wish to pass an additional parameter

Returns:

1 if character is visible, 0 if not

5.12.628 function diguyCharacter::check_point_visibility**Prototype:**

```
int check_point_visibility(float x, float y, float z);
```

Description:

This function tests whether this character can see the x, y, z location specified. Currently this tests if the cervical link of this character has a clear line of sight to the x, y, z location. Currently the orientation of this character isn't factored in.

This function does not fail if other characters are between the character and the location in question.

Checking visibility is only an option if an octree is generated from scene objects; typically only DI-Guy Scenario does this.

Arguments:

x, y, z location to check

Returns:

1 if location is visible, 0 if not

AI Callback Functions

Agents use visibility tests to assess if they should path plan or attack. These functions should be filled out by SDK users who want intelligent behavior and decision making from agents.

These functions require that end users have a fast static intersection test infrastructure.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.629 function diguyCharacter::set_point_los_function**Prototype:**

```
int set_point_los_function(diguyPointLOSFunction* point_los_function);
```

Description:

When a character needs to check if it can see a location, DI-Guy calls this function with a pointer to the [diguyCharacter](#) in question, and the coordinates of the point that is being looked at. This currently happens when an agent is told to move to a new region.

The following prototype should be used for the point line-of-sight function:

```
int diguyPointLOSFunction(diguyCharacter* character,
float x,
float y,
float z);
```

The function should then check if the desired location is visible. This can be done in a number of ways; a typical approach is to do an intersection test with the static scene.

The function should return 1 if the path to the target is clear, 0 if there is an obstruction.

A default point line-of-sight function can be set for all subsequently created characters by calling [diguyScenario::set_default_point_los_function\(\)](#).

Callable From:

◇ C++

5.12.630 function diguyCharacter::get_point_los_function**Prototype:**

```
diguyPointLOSFunction* get_point_los_function();
```

Returns:

current point line-of-sight function; see [set_point_los_function\(\)](#)

Callable From:

◇ C++

5.12.631 function diguyCharacter::set_character_los_function**Prototype:**

```
int set_character_los_function(diguyCharacterLOSFunction* character_los_function);
```

Description:

When a character needs to check if it can see another character, DI-Guy calls this function with a pointer to the [diguyCharacter](#) in question. This can happen when [check_character_visibility\(\)](#) is invoked or when AI attack code assesses if it should take a shot.

The following prototype should be used for the character line-of-sight function:

```
int diguyCharacterLOSFunction(diguyCharacter* character,
    diguyCharacter* target_character,
    int los_test_type) // (currently unused)
```

Currently the `los_test_type` argument defaults to 0 and is not used internally. It is, however, available in [check_character_visibility\(\)](#) and will be passed to the callback. This can be useful if the end user wants to write

scripting code that has alternative Line of Sight tests depending on agent conditions. (e.g. a test that factors in the character's field of view.)

The function then checks if the target character is visible. This can be done in a number of ways; a typical approach is to do an intersection test with the static scene. The default implementation tests the path from the characters head to the target's head and then to the target's feet.

The function should return 1 if the path to the target is clear, and 0 if there is an obstruction.

A default character line-of-sight function can be set for all subsequently created characters by calling
[diguyScenario::set_default_character_los_function\(\)](#).

Callable From:

◇ C++

5.12.632 function [diguyCharacter::get_character_los_function](#)

Prototype:

```
diguyCharacterLOSFunction* get_character_los_function();
```

Returns:

current character line-of-sight function; see
[set_character_los_function\(\)](#)

Callable From:

◇ C++

Simulation Functions

The functions in this section control physics simulator settings for characters.

For many of the functions a DI-Guy physics module must be active.

Unless otherwise specified, all functions callable from:

- ◆ C++
- ◆ Script

5.12.633 function `diguyCharacter::set_initial_character_simulator`

Prototype:

```
int set_initial_character_simulator(diguyCharacterSimulator character_simulator);
```

Description:

This function sets the initial character simulator this character will use. The current simulator will be set to this value on scenario reset.

The current simulator can be set by calling
`set_current_character_simulator()`.

The default simulator is
`DIGUY_CHARACTER_SIMULATOR_DIGUY_MOTION_ENGINE`.

Requires active physics module if character simulator is set to
`DIGUY_CHARACTER_SIMULATOR_PHYSICS_SIM_MODULE`.

Arguments:

`character_simulator` which simulator to use at reset

Returns:

0 on success, -1 on failure

5.12.634 function `diguyCharacter::get_initial_character_simulator`

Prototype:

```
diguyCharacterSimulator get_initial_character_simulator();
```

Returns:

initial character simulator

5.12.635 function diguyCharacter::set_current_character_simulator

Prototype:

```
int set_current_character_simulator(diguyCharacterSimulator character_simulator);
```

Description:

This function sets the current character simulator this character will use. The current simulator will be reset to the initial character simulator on scenario reset.

The initial simulator can be set by calling
[set_initial_character_simulator\(\)](#).

Arguments:

`character_simulator` new current character simulator

Returns:

0 on success, -1 on failure

5.12.636 function diguyCharacter::get_current_character_simulator

Prototype:

```
diguyCharacterSimulator get_current_character_simulator();
```

Returns:

current character simulator

5.12.637 function

`diguyCharacter::set_initial_physics_simulation_mode`

Prototype:

```
int set_initial_physics_simulation_mode(diguyPhysicsSimulationMode physics_simulation_mode)
```

Description:

This function sets the initial physics simulation mode this character will use. The current simulation mode will be set to this value on scenario reset.

The current simulation mode can be set by calling `set_current_physics_simulation_mode()`.

The default simulation mode is `DIGUY_PHYSICS_SIMULATION_MODE_SIM_DRIVES_CHARACTER`.

The physics simulation mode will have no effect unless the current character simulator is `DIGUY_CHARACTER_SIMULATOR_PHYSICS_SIM_MODULE`.

Arguments:

`physics_simulation_mode` which simulation mode to use at reset

Returns:

0 on success, -1 on failure

5.12.638 function**diguyCharacter::get_initial_physics_simulation_mode****Prototype:**

```
diguyPhysicsSimulationMode get_initial_physics_simulation_mode();
```

Returns:

initial physics simulation mode

5.12.639 function**diguyCharacter::set_current_physics_simulation_mode****Prototype:**

```
int set_current_physics_simulation_mode(diguyPhysicsSimulationMode physics_simulation_mode);
```

Description:

This function sets the current physics simulation mode this character will use. The current simulation mode will be reset to the initial physics simulator mode on scenario reset.

The initial simulation mode can be set by calling `set_initial_physics_simulation_mode()`.

The physics simulation mode will have no effect unless the current character simulator is `DIGUY_CHARACTER_SIMULATOR_PHYSICS_SIM_MODULE`.

Arguments:

`physics_simulation_mode` new current physics simulation mode

Returns:

0 on success, -1 on failure

5.12.640 function `diguyCharacter::get_current_physics_simulation_mode`

Prototype:

```
diguyPhysicsSimulationMode get_current_physics_simulation_mode();
```

Returns:

initial physics simulation mode

5.12.641 function diguyCharacter::set_physics_collision_group**Prototype:**

```
int set_physics_collision_group(diguyPhysicsCollisionGroup collision_group);
```

Description:

This function sets the physics collision group this character will be in when being simulated. For improved performance, collision detection between various characters and scene objects will not be performed, based on which collision groups the objects are in.

By default:

- ◇ characters in character group n collide against props in prop group n and scene objects in scene object group n in group n
- ◇ characters in group ghost don't collide against any other characters, but do against scene objects and props
- ◇ characters in group everything collide against all characters, props, and scene objects
- ◇ all props collide against all other props and scene objects

Collision group relationships can be modified using [diguyApp::set_collision_group_detection_flag\(\)](#).

Arguments:

`physics_simulation_mode` new current physics simulation mode

Returns:

0 on success, -1 on failure

5.12.642 function diguyCharacter::get_physics_collision_group**Prototype:**

```
diguyPhysicsCollisionGroup get_physics_collision_group();
```

Returns:

physics collision group

5.12.643 function diguyCharacter::set_stop_physics_sim_at_t**Prototype:**

```
void set_stop_physics_sim_at_t(float t);
```

Description:

This function sets a time at which active simulation of a physics controlled character will stop. This is useful for:

- ◇ reducing simulation processing overhead
- ◇ freezing a simulation that is mostly in steady state, but some small part remains oscillating

The physics controlled character will remain frozen in the last computed simulated pose.

Arguments:

t scenario t at which simulation will freeze

t should be large enough for the simulation to reach a reasonably steady state, such as a human lying on the ground. A value that is usually reasonable is around 5 seconds.

5.12.644 function `diguyCharacter::set_stop_physics_sim_in_t_seconds`

Prototype:

```
void set_stop_physics_sim_in_t_seconds(float relative_t);
```

Description:

Similar to `set_stop_physics_sim_at_t( )`, but the passed time will be relative to the current simulation time as returned by `diguyScenario::get_t()`.

5.12.645 function diguyCharacter::get_stop_physics_sim_at_t**Prototype:**

```
float get_stop_physics_sim_at_t();
```

Returns:

time at which physics simulation will freeze

5.12.646 function diguyCharacter::apply_impulse_to_physics_sim**Prototype:**

```
void apply_impulse_to_physics_sim(const char* link,
    float impulse_magnitude_x,
    float impulse_magnitude_y,
    float impulse_magnitude_z,
    float link_offset_x = 0.0f,
    float link_offset_y = 0.0f,
    float link_offset_z = 0.0f,
    int set_simulation_drives_character = 1);
```

Description:

This function applied an impulse force to the character. An impulse is a force that is applied all at one time, such as an explosion shock wave.

Arguments:

link	link to which impulse should be applied
impulse_magnitude_[xyz]	magnitude of impulse in x direction

<code>link_offset_[xyz]</code>	offset on link at which to apply impulse
<code>set_simulation_drives_character</code>	pass 1 to automatically change the character's current simulation mode to <code>DIGUY_PHYSICS_SIMULATION_MODE_SIM_DRIVE</code>

5.12.647 function

`diguyCharacter::apply_impulse_to_physics_sim_from_world_pos`

Prototype:

```
void apply_impulse_to_physics_sim_from_world_position(const char* link,
    float impulse_magnitude_x,
    float impulse_magnitude_y,
    float impulse_magnitude_z,
    float impulse_generator_position_x,
    float impulse_generator_position_y,
    float impulse_generator_position_z,
    float max_distance = 20.0f,
    float impulse_rolloff = 0.0f,
    float link_offset_x = 0.0f,
    float link_offset_y = 0.0f,
    float link_offset_z = 0.0f,
    int set_simulation_drives_character = 1);
```

Description:

Similar to `apply_impulse_to_physics_sim()`, but impulse can be calculated relative to a world position. This is useful if, for example, a character should react to a nearby explosion.

Arguments:

<code>link</code>	link to which impulse should be applied
<code>impulse_magnitude_[xyz]</code>	magnitude of impulse
<code>impulse_generator_position_[xyz]</code>	world coordinates of impulse generating object
<code>max_distance</code>	maximum distance from above position at which the character
<code>impulse_rolloff</code>	how much the impulse magnitude should decrease as source increases
<code>link_offset_[xyz]</code>	offset on link at which to apply impulse
<code>set_simulation_drives_character</code>	pass 1 to automatically change the character's current DIGUY_PHYSICS_SIMULATION_MODE_SIM_I

Experimental Functions

The following functions are experimental and may change or disappear with no warning.

5.12.648 function diguyCharacter::set_texture_load_behavior**Prototype:**

```
static int set_texture_load_behavior(diguyGraphicsTextureLoadingBehavior mode);
```

Description:

Sets texture load behavior to [DIGUY_LOAD_INSTANTLY](#), [DIGUY_LOAD_PRIORITY_ASYNC](#), or [DIGUY_LOAD_ASYNC](#)

5.12.649 function diguyCharacter::get_texture_load_behavior**Prototype:**

```
static diguyGraphicsTextureLoadingBehavior get_texture_load_behavior();
```

Returns current behavior, as set in [set_texture_load_behavior\(\)](#)

5.12.650 function diguyCharacter::get_has_bump_maps**Prototype:**

```
int get_has_bump_maps();
```

Returns 1 if the current appearance has bump maps

5.12.651 function diguyCharacter::get_action_num_channels**Prototype:**

```
int get_action_num_channels(const char* action_name);
```

Description:

This function returns how many channels the specified action has. If an action has more than one channel, the relative weights of its channels can be set using the [set_action_channel_weight\(\)](#) function.

Arguments:

`action_name` name of action to be queried

Returns:

number of channels in specified action

5.12.652 function diguyCharacter::set_action_channel_weight**Prototype:**

```
int set_action_channel_weight(const char* action_name,  
                             char channel,  
                             float weight);
```

Description:

This function sets the weight one channel of a multi-channel action.

The passed weight should be between 0 and 1. The weights of the other channels in the action will be adjusted so that the total weight of all channels is 1.

Arguments:

<code>action_name</code>	name of action to be affected
<code>channel</code>	which channel's weight to set
<code>weight</code>	weight of the channel, between 0 and 1

Returns:

0 on success, -1 on failure

5.12.653 function diguyCharacter::get_action_channel_weight**Prototype:**

```
float get_action_channel_weight(const char* action_name,
                               char channel);
```

Description:

This function returns the weight of one channel of a multi-channel action.

Arguments:

<code>action_name</code>	name of action to be queried
<code>channel</code>	which channel's weight to return

Returns:

weight of a channel

5.12.654 function diguyCharacter::get_time_to_reach_desired_action**Prototype:**

```
float get_time_to_reach_desired_action();
```

Description:

This function returns the number of seconds before the transition to the desired action begins. If the character is already performing the desired action the function returns 0.

THIS FUNCTION DOES NOT YET RETURN VALID RESULTS.

Returns:

time to reach desired action, in seconds

5.12.655 function diguyCharacter::get_type_map_field**Prototype:**

```
const char* get_type_map_field(diguyCharacterTypeMapField field);
```

Description:

Maps a character type map field to its value. See [diguyCharacterTypeMap](#).

Returns:

The value, as a string.

5.12.656 function diguyCharacter::get_type_map_field_using_string**Prototype:**

```
const char* get_type_map_field_using_string(const char* field);
```

Description:

Maps a character type map field to its value, using a string version (e.g. "age") of the field name. See [diguyCharacterTypeMap](#).

Returns:

The value, as a string.

5.12.657 function diguyCharacter::merge_object**Prototype:**

```
int merge_object(const char* object);
```

Description:

This function performs a "micro merge", allowing a [diguyCharacter](#) to have objects added to it by parsing a .dss file fragment in the form of a string.

This function can be used, for example, to add paths to dynamically created agents. A Lua Package can contain a number of text strings representing local paths for agents. On character creation these strings can be merged in and turned into per-character objects.

The DI-Guy Scenario path page has a "Display Asset" button for generating [merge_object\(\)](#) compatible strings.

There is also a [diguyScenario::merge_object\(\)](#) function that allows the merging of text blocks that represent sounds and particle systems.

Note: the parser is very sensitive to tabs and formatting!

5.12.658 function diguyCharacter::is_active_character_in_group_visible**Prototype:**

```
int is_active_character_in_group_visible(const char* group_name,
float max_distance_to_check = 100000.0f);
```

Description:

Returns 1 if any active character in the named group is visible, otherwise 0.

group_name max_distance_to_check

5.12.659 function diguyCharacter::get_local_space_bounding_box

Prototype:

```
int get_local_space_bounding_box(float* x1, float* y1, float* z1,
    float* x2, float* y2, float* z2);
```

Description:

Gets local space bounding box, this function is currently expensive and should be used with care.

Arguments:

x1, y1, z1, x2, y2, z2 output parameters

Returns:

0 on success, -1 on failure (float * arguments converted to return values in Lua)

5.12.660 function diguyCharacter::get_geometry_tag

Prototype:

```
int get_geometry_tag(int tag,
    int world_space,
    float* x1, float* y1, float* z1, float* radius);
```

Description:

Get a spherical area containing geometry with specified tag, in specified world space.

Arguments:

tag	the tag
world_space	the world space
x1, y1, z1, radius	output parameters returning sphere definition

Returns:

0 on success, -1 on failure (float * arguments converted to return values in Lua)

5.12.661 function diguyCharacter::set_mind_always_updates

Prototype:

```
void set_mind_always_updates(int force_update);
```

Description:

This causes a character's mind to get an update call even if time isn't passing in the scenario. This is occasionally useful for characters that are managing the UI.

Note that there is now a luaUIMachine that might be a better solution.

5.12.662 function diguyCharacter::get_mind_always_updates**Prototype:**

```
int get_mind_always_updates();
```

Description:

Returns value set by `set_mind_always_updates()`

5.12.663 function diguyCharacter::get_angle_to_path**Prototype:**

```
float get_angle_to_path(const char* path_name,  
                        const char* action_bead_name = NULL);
```

Description:

Returns angle in degrees from character's "forward facing" vector to the specified action bead, on the specified path

5.12.664 function diguyCharacter::get_aim_at_link**Prototype:**

```
const char* get_aim_at_link();
```

Returns:

Name of a reasonable link that other characters could aim at.

5.12.665 function diguyCharacter::get_aim_at_link_offset**Prototype:**

```
int get_aim_at_link_offset(float* offset_x, float* offset_y, float* offset_z);
```

Returns:

xyz offset from link returned by [get_aim_at_link\(\)](#)

5.12.666 function diguyCharacter::get_gaze_at_link**Prototype:**

```
const char* get_gaze_at_link();
```

Returns:

Name of a reasonable link that other characters could look at.

5.12.667 function diguyCharacter::get_gaze_at_link_offset**Prototype:**

```
int get_gaze_at_link_offset(float* offset_x, float* offset_y, float* offset_z);
```

Returns:

xyz offset from link returned by [get_gaze_at_link\(\)](#)

5.12.668 function diguyCharacter::set_scene_graph_mask**Prototype:**

```
void set_scene_graph_mask(unsigned int mask);
```

5.12.669 function diguyCharacter::set_use_texture_variations**Prototype:**

```
int set_use_texture_variations(int value);
```

Description:

Turns on texture modulation system that tries to make all character look unique only supported by some appearances currently.

5.12.670 function diguyCharacter::get_use_texture_variations**Prototype:**

```
int get_use_texture_variations();
```

Returns:

If texture variations have been enabled for this character.

5.12.671 function diguyCharacter::set_emit_light**Prototype:**

```
int set_emit_light(int emit_light);
```

Description:

Sets whether or not character emits light.

Arguments:

emit_light 0 or 1

5.12.672 function diguyCharacter::get_emit_light**Prototype:**

```
int get_emit_light();
```

Returns:

0 or 1, depending on whether or not character emits light.

5.12.673 function diguyCharacter::set_max_shader_quality_level**Prototype:**

```
static int set_max_shader_quality_level(int quality_level);
```

Description:

Sets maximum quality of shader to use (actual quality varies by distance from camera). See [diguyGraphicsShaderQualityLevel](#) for possible values.

Arguments:

`quality_level` a value between
DIGUY_GRAPHICS_SHADER_QUALITY_UNLIT

5.12.674 function diguyCharacter::get_max_shader_quality_level**Prototype:**

```
static int get_max_shader_quality_level();
```

Returns:

max shader quality level, as set by
`set_max_shader_quality_level()`

Deprecated Functions

The following functions are deprecated.

As these functions may disappear in a future version of DI-Guy, it is recommended that you remove calls to these functions from your application.

5.12.675 function diguyCharacter::get_base_link**Prototype:**

```
diguyGraphicsLink* get_base_link();
```

Deprecated as of 9.1.3; use
[diguyCharacter::get_position_link\(\)](#) instead.

5.12.676 function diguyCharacter::set_apply_actor_scale_to_action_bead_xy_travel**Prototype:**

```
void set_apply_actor_scale_to_action_bead_xy_travel(int apply_scale_flag);
```

Deprecated as of 9.0.1; use [set_apply_actor_scale_to_action_bead_travel\(\)](#)

instead.

5.12.677 function**diguyCharacter::get_apply_actor_scale_to_action_bead_xy_trave****Prototype:**

```
int get_apply_actor_scale_to_action_bead_xy_travel();
```

Deprecated as of 9.0.1; use `get_apply_actor_scale_to_action_bead_travel()`

instead.

5.12.678 function diguyCharacter::set_default_weapon_sound**Prototype:**

```
int set_default_weapon_sound(const char* sound_name);
```

Deprecated as of 9.1.4. In general setting the weapon sound should no longer be necessary, as weapon sounds are now specified by the weapon's munition type. If this function is called, it will have the same effect as calling `diguyCharacter::set_weapon_sound_override()`.

5.12.679 function diguyCharacter::get_default_weapon_sound**Prototype:**

```
const char* get_default_weapon_sound();
```

Deprecated as of 9.1.4. See comments for [set_default_weapon_sound\(\)](#).

5.12.680 function diguyCharacter::set_default_weapon_sound_gain**Prototype:**

```
int set_default_weapon_sound_gain(float gain);
```

Deprecated as of 9.1.4. See comments for [set_default_weapon_sound\(\)](#).

5.12.681 function `diguyCharacter::get_default_weapon_sound_gain`

Prototype:

```
float get_default_weapon_sound_gain();
```

Deprecated as of 9.1.4. See comments for
[set_default_weapon_sound\(\)](#).

5.12.682 function `diguyCharacter::set_weapon_munition_type`

Prototype:

```
int set_weapon_munition_type(const char* munition_type);
```

Deprecated as of 9.1.4. Call
[set_weapon_munition_type_override\(\)](#) instead.

5.12.683 function diguyCharacter::get_weapon_munition_type**Prototype:**

```
const char* get_weapon_munition_type();
```

Deprecated as of 9.1.4. Call

[get_weapon_munition_type_override\(\)](#) instead.

5.12.684 function diguyCharacter::set_weapon_horizontal_error**Prototype:**

```
int set_weapon_horizontal_error(float h_error);
```

Deprecated as of 10.5.2. Call [set_weapon_max_azimuth_error\(\)](#) instead.

5.12.685 function diguyCharacter::get_weapon_horizontal_error**Prototype:**

```
float get_weapon_horizontal_error();
```

Deprecated as of 10.5.2. Call [get_weapon_max_azimuth_error\(\)](#) instead.

5.12.686 function diguyCharacter::set_weapon_vertical_error**Prototype:**

```
int set_weapon_vertical_error(float v_error);
```

Deprecated as of 10.5.2. Call [set_weapon_max_elevation_error\(\)](#) instead.

5.12.687 function diguyCharacter::get_weapon_vertical_error**Prototype:**

```
float get_weapon_vertical_error();
```

Deprecated as of 10.5.2. Call [get_weapon_max_elevation_error\(\)](#) instead.

5.12.688 function diguyCharacter::begin_legacy_appearance_effect**Prototype:**

```
int begin_legacy_appearance_effect(const char* effect,
    const char* override_link = "",
    float override_scale = DIGUY_DEFAULT_FLOAT,
    float override_offset_x = DIGUY_DEFAULT_FLOAT,
    float override_offset_y = DIGUY_DEFAULT_FLOAT,
    float override_offset_z = DIGUY_DEFAULT_FLOAT);
```

Added as of 10.1.6. Invokes the old implementation of [begin_appearance_effect\(\)](#).

5.12.689 function**diguyCharacter::end_legacy_appearance_effect****Prototype:**

```
int end_legacy_appearance_effect(const char* effect);
```

Added as of 10.1.6. Invokes the old implementation of [end_appearance_effect\(\)](#).

5.12.690 function**diguyCharacter::legacy_appearance_effect_is_active****Prototype:**

```
int legacy_appearance_effect_is_active(const char* effect);
```

Added as of 10.1.6. Invokes the old implementation of [appearance_effect_is_active\(\)](#).

5.12.691 function diguyCharacter::merge_asset**Prototype:**

```
int merge_asset(const char* asset);
```

Deprecated as of 10.5.1. Use [merge_object\(\)](#) instead.

5.12.692 function diguyCharacter::set_DIS_smoke_trail_state**Prototype:**

```
void set_DIS_smoke_trail_state(int state);
```

Deprecated as of 10.5.2. Use [get_DIS_trailing_effect_state\(\)](#) instead.

5.12.693 function diguyCharacter::get_DIS_smoke_trail_state**Prototype:**

```
int get_DIS_smoke_trail_state();
```

Deprecated as of 10.5.2. Use [get_DIS_trailing_effect_state\(\)](#) instead.

5.12.694 function diguyCharacter::speak**Prototype:**

```
int speak(const char* string);
```

Deprecated as of 12.0.0. Not supported in FaceFX.

5.12.695 function diguyCharacter::set_viseme**Prototype:**

```
int set_viseme(const char* viseme_name, float weight);
```

Deprecated as of 12.0.0. Not supported in FaceFX.

5.12.696 function diguyCharacter::get_viseme**Prototype:**

```
float get_viseme(const char* viseme_name);
```

Deprecated as of 12.0.0. Not supported in FaceFX.

5.12.697 function diguyCharacter::set_orientation_eye_left**Prototype:**

```
int set_orientation_eye_left(float azimuth, float elevation);
```

Deprecated as of 12.0.0. Not supported in FaceFX.

5.12.698 function diguyCharacter::set_orientation_eye_right**Prototype:**

```
int set_orientation_eye_right(float azimuth, float elevation);
```

Deprecated as of 12.0.0. Not supported in FaceFX.

5.12.699 function diguyCharacter::get_azimuth_eye_left**Prototype:**

```
float get_azimuth_eye_left();
```

Deprecated as of 12.0.0. Not supported in FaceFX.

5.12.700 function diguyCharacter::get_elevation_eye_left**Prototype:**

```
float get_elevation_eye_left();
```

Deprecated as of 12.0.0. Not supported in FaceFX.

5.12.701 function diguyCharacter::get_azimuth_eye_right**Prototype:**

```
float get_azimuth_eye_right();
```

Returns:

the current azimuth of the right eye; see [set_orientation_eyes\(\)](#)

5.12.702 function diguyCharacter::get_elevation_eye_right**Prototype:**

```
float get_elevation_eye_right();
```

Returns:

the current elevation of the right eye; see [set_orientation_eyes\(\)](#)

5.13 diguyCharacterFaceExpression

diguyCharacterFaceExpression.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

Class acts as a wrapper around a individual face pose like mad or angry All functions in this class are callable from:

- ◆ C++
- ◆ Script

5.13.1 class diguyCharacterFaceExpression

```
class BDI_DECLSPEC_diguy diguyCharacterFaceExpression
{
public:
    const char* get_name();
    void clear();
    int get_register_count();
    const char* get_register_name_by_index(int register_index);
    float get_register_value(const char* name);
    int set_register_value(const char* name, float value);
    float get_register_value_by_index(int register_index);
    int set_register_value_by_index(int register_index, float value);
    int get_register_enabled(const char* name);
    int set_register_enabled(const char* name, int enabled);
    int set_all_registers_enabled(int enabled);
    int get_register_enabled_by_index(int register_index);
    int set_register_enabled_by_index(int register_index, int enabled);
};
```

5.13.2 function diguyCharacterFaceExpression::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of this face expression. This pointer will never be NULL.

Returns:

name of this face expression

5.13.3 function diguyCharacterFaceExpression::clear

Prototype:

```
void clear();
```

Description:

Sets all of this face expression's registers to zero.

5.13.4 function `diguyCharacterFaceExpression::get_register_count`

Prototype:

```
int get_register_count();
```

Returns:

number of registers in this face expression

5.13.5 function `diguyCharacterFaceExpression::get_register_name_by_index`

Prototype:

```
const char* get_register_name_by_index(int register_index);
```

Returns:

a register name, given the index of that register

Arguments:

`register_index` index of face_expression register

5.13.6 function diguyCharacterFaceExpression::get_register_value

Prototype:

```
float get_register_value(const char* name);
```

Description:

Gets the value of a face expression register, given the name of the register.

Arguments:

name	name of register

Returns:

value of the specified register

5.13.7 function diguyCharacterFaceExpression::set_register_value

Prototype:

```
int set_register_value(const char* name, float value);
```

Description:

Sets the value of a face expression register.

Arguments:

name	name of register to set
value	new value for register

Returns:

0 on success, -1 on failure

5.13.8 function diguyCharacterFaceExpression::get_register_value_by_index

Prototype:

```
float get_register_value_by_index(int register_index);
```

Description:

Gets the value of a face expression register, given the index of the register.

Arguments:

`register_index` index of the register

Returns:

value of the specified register

5.13.9 function diguyCharacterFaceExpression::set_register_value_by_index

Prototype:

```
int set_register_value_by_index(int register_index, float value);
```

Description:

Sets the value of a face expression register.

Arguments:

`register_index` index of the register
`value` new value for register

Returns:

0 on success, -1 on failure

5.13.10 function diguyCharacterFaceExpression::get_register_enabled

Prototype:

```
int get_register_enabled(const char* name);
```

Description:

Gets whether a face expression register is enabled.

A nonzero value means that the register's value will be assigned to the character's face's corresponding attribute when the character applies the face expression using `diguyCharacter::set_face_expression()`.

A zero value means that the register will have no effect on the character's face when the character applies the face expression using `diguyCharacter::set_face_expression()`.

So, each face expression can affect some, all, or no attributes of the character's face.

Arguments:

name	name of the register

Returns:

whether the register is enabled

See Also:

`set_register_enabled()`

5.13.11 function diguyCharacterFaceExpression::set_register_enabled

Prototype:

```
int set_register_enabled(const char* name, int enabled);
```

Description:

Sets whether a face expression register is enabled. See [get_register_enabled\(\)](#).

Arguments:

name	name of the register to check
enabled	whether to enable the register

Returns:

0 on success, -1 on failure

See Also:

[get_register_enabled\(\)](#), [set_all_registers_enabled\(\)](#)

5.13.12 function diguyCharacterFaceExpression::set_all_registers_enabled

Prototype:

```
int set_all_registers_enabled(int enabled);
```

Description:

Sets the enabled status of all face expression registers. This is used to turn all registers on, or all registers off.

Arguments:

enabled	whether to enable or disable all registers

Returns:

0 on success, -1 on failure

C++ Example:

```
diguyCharacterFaceExpression* fx;

fx = scenario->create_face_expression("Smile_exp");
if (fx)
{
fx->set_register_value("Smile", 1);
fx->set_all_registers_enabled(1);
}

fx = scenario->create_face_expression("Smile_cl_exp");
if (fx)
{
fx->set_register_value("Smile_cl", 1);
fx->set_all_registers_enabled(1);
}

fx = scenario->create_face_expression("Mad_exp");
if (fx)
{
fx->set_register_value("Mad", 1);
fx->set_all_registers_enabled(1);
}
```

5.13.13 function diguyCharacterFaceExpression::get_register_enabled_by_index

Prototype:

```
int get_register_enabled_by_index(int register_index);
```

Description:

Gets whether a face expression register is enabled.

Arguments:

index	index of the register to check

Returns:

whether the register is enabled

5.13.14 function `diguyCharacterFaceExpression::set_register_enabled_by_index`

Prototype:

```
int set_register_enabled_by_index(int register_index, int enabled);
```

Description:

Sets whether a face expression register is enabled.

Arguments:

<code>index</code>	index of the register to modify
<code>enabled</code>	whether to enable or disable the register

Returns:

0 on success, -1 on failure

5.14 diguyCharacterGesture

diguyCharacterGesture.h

DI-Guy SDK Version 12.5.2

Includes: [diguyCharacterGestureType.h](#) | [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Common Functions](#)
- ◆ [Progression Gesture Functions](#)
- ◆ [Table Gesture Functions](#)
- ◆ [Callback Functions](#)
- ◆ [Deprecated Functions](#)

5.14.1 class diguyCharacterGesture

```
class BDI_DECLSPEC_diguy diguyCharacterGesture
{
public:
    const char* get_name();
    diguyCharacterGestureType get_gesture_type();
    int begin_now();
    float get_tin();
    int get_has_begun_flag();
    int end_gesture();
    int get_has_completed_flag();
    int abort_now(float max_rampdown_interval);
    int set_automatic_destroy_flag(int automatic_destroy_flag);
    int set_interpolation_flag(int flag);
    int get_interpolation_flag();
    int get_num_channels();
    int set_channel_weight(char channel, float channel_weight);
    float get_channel_weight(char channel);
    int set_channel_weight_target(char channel,
        float channel_weight_target,
        float blend_tin,
        float blend_tout,
        const char* blend_function_name = 0);
    int add_channel_weight_target(char channel,
        float channel_weight_target,
        float blend_tin,
        float blend_tout,
        const char* blend_function_name = 0);
    int set_rampup_duration(float duration);
    float get_rampup_duration();
    int set_rampup_blend_function_name(const char* blend_function_name);
    const char* get_rampup_blend_function_name();
    int set_rampdown_duration(float duration);
    float get_rampdown_duration();
    int set_rampdown_blend_function_name(const char* blend_function_name);
    const char* get_rampdown_blend_function_name();
    float get_default_blend_duration();
```

```

int set_max_reps(int max_reps,
    int derive_duration_flag = 1);
int get_max_reps();
int set_overall_duration(float overall_duration,
    int derive_max_reps_flag = 1);
float get_overall_duration();
int get_num_progression_stages();
int set_progression_stage_duration(int stage, float duration);
float get_progression_stage_duration(int stage);
float get_progression_stage_optimal_duration(int stage);
int set_table_action_desired(const char* action_name);
const char* get_table_action_desired();
float get_time_required_to_reach_table_action_desired();
int get_num_table_actions();
const char* get_table_action_at_index(int index);
float get_table_action_rep_duration(const char* action_name);
float get_table_transition_duration(const char* action_from,
    const char* action_to);
int set_table_action_begin(const char* action_name);
const char* get_table_action_begin();
int set_table_action_end(const char* action_name);
const char* get_table_action_end();
const char* get_table_action_now();
const char* get_table_action_next();
int force_table_action(const char* action_name,
    int include_transition_arc = 1,
    float max_rampdown_interval = 0.5f);
enum {
    CALLBACK_ID_CREATE = 1,
    CALLBACK_ID_DESTROY,
    CALLBACK_ID_ATTAINED_DESIRED_ACTION
};
#ifdef CPLUSPLUS_ONLY
int add_callback(int callback_id,
    diguyCharacterGestureCallback* callback,
    void* callback_params = 0,
    void* callback_user_data = 0);
int remove_callback(int callback_id,
    diguyCharacterGestureCallback* callback);
int remove_callback_with_user_data(int callback_id,
    void* callback_user_data);
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int remove_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int get_num_stages();
int set_stage_duration(int stage, float duration);
float get_stage_duration(int stage);
float get_stage_optimal_duration(int stage);

```

Common Functions

5.14.2 function diguyCharacterGesture::get_name

Prototype:

```
const char* get_name();
```

Returns:

name of the gesture

Callable For Gesture Types:

◇ all

Callable From:

◇ C++
◇ Script

5.14.3 function diguyCharacterGesture::get_gesture_type

Prototype:

```
diguyCharacterGestureType get_gesture_type();
```

Returns:

type of the gesture

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.4 function diguyCharacterGesture::begin_now

Prototype:

```
int begin_now();
```

Description:

This function causes the gesture to begin immediately. The gesture will continue until it has completed or is aborted.

Calling this function on a gesture that has already begun will result in a warning message and have no other effect.

Progression Gestures:

The overall duration of the gesture can be set by calling [set_overall_duration\(\)](#). The durations of each stage will be set proportionately over the overall duration.

Alternatively, the durations of individual stages of the gesture can be set by calling [set_stage_duration\(\)](#).

The number of repetitions of the gesture can be set by calling `set_max_reps()`.

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.5 function diguyCharacterGesture::get_tin

Prototype:

```
float get_tin();
```

Description:

This function returns the time at which the gesture began.

Returns:

time in seconds when scenario began

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.6 function diguyCharacterGesture::get_has_begun_flag

Prototype:

```
int get_has_begun_flag();
```

Returns:

1 if the gesture has begun; 0 if not

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.7 function diguyCharacterGesture::end_gesture

Prototype:

```
int end_gesture();
```

Description:

This function notifies the gesture that it should complete as soon as possible. To end a gesture immediately, call [abort_now\(\)](#).

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.8 function diguyCharacterGesture::get_has_completed_flag**Prototype:**

```
int get_has_completed_flag();
```

Returns:

1 if the gesture has completed; 0 if not

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.9 function diguyCharacterGesture::abort_now**Prototype:**

```
int abort_now(float max_rampdown_interval);
```

Description:

This function causes this gesture to abort. Aborting a gesture doesn't destroy it. The gesture can be begun again if desired. To end a gesture gracefully call [end_gesture\(\)](#).

Arguments:

<code>max_rampdown_interval</code>	max amount of time spent trying to smooth over any motion seams; pass 0.5 for a smooth transition, pass 0 for an immediate abort

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ all

Callable From:

◇ C++
◇ Script

5.14.10 function diguyCharacterGesture::set_automatic_destroy_flag

Prototype:

```
int set_automatic_destroy_flag(int automatic_destroy_flag);
```

Description:

This function causes the gesture object to be automatically deleted when the gesture has completed. No further calls through the [diguyCharacterGesture](#) object can or should be made, so this function should only be called once all durations, reps, etc. have been set.

`automatic_destroy_flag` pass 1 to enable automatic deletion

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.11 function diguyCharacterGesture::set_interpolation_flag

Prototype:

```
int set_interpolation_flag(int flag);
```

Description:

This function sets whether joint angles in the gesture will be interpolated. This takes more time, but results in smoother motion at high frame rates.

Arguments:

<code>flag</code>	pass 1 to turn on interpolation, pass 0 to turn off interpolation

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.12 function diguyCharacterGesture::get_interpolation_flag

Prototype:

```
int get_interpolation_flag();
```

Returns:

the most recent setting of [set_interpolation_flag\(\)](#)

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.13 function diguyCharacterGesture::get_num_channels**Prototype:**

```
int get_num_channels();
```

Returns:

how many channels are in the gesture

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.14 function diguyCharacterGesture::set_channel_weight**Prototype:**

```
int set_channel_weight(char channel, float channel_weight);
```

Description:

This function sets the weight of the specified channel. The other channel weights in a multi-channel gesture will be affected by this call so that all channel weight add to 1.

If the gesture does not have the channel, this function has no effect.

Arguments:

<code>channel</code>	which channel to affect; pass a single upper case character (e.g., 'A')
<code>channel_weight</code>	value between 0 and 1; 1 means gesture is entirely driven by specified channel, 0 means specified channel has no effect on gesture

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.15 function diguyCharacterGesture::get_channel_weight

Prototype:

```
float get_channel_weight(char channel);
```

Returns:

the current weight of the specified channel; see [set_channel_weight\(\)](#)

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.16 function

diguyCharacterGesture::set_channel_weight_target

Prototype:

```
int set_channel_weight_target(char channel,
                             float channel_weight_target,
                             float blend_tin,
                             float blend_tout,
                             const char* blend_function_name = 0);
```

Description:

This function schedules a smooth ramp of the channel weight from its current value to the specified target value.

This function is equivalent to calling `set_channel_weight()` many times over the course of a gesture.

Calling this function removes any pending channel weight targets added by calls to either `set_channel_weight_target()` or `add_channel_weight_target()`.

Arguments:

<code>channel</code>	which channel to affect; pass a single upper case character (e.g., 'A')
<code>channel_weight_target</code>	target channel weight; a value between 0 and 1
<code>blend_tin</code>	when, relative to the start of the gesture, the channel weight should begin moving toward <code>channel_weight_target</code>
<code>blend_tout</code>	when, relative to the start of the gesture, the channel weight should arrive at <code>channel_weight_target</code>
<code>blend_function_name</code>	name of the curve to use to move between current channel weight and target channel weight

The possible blend function names are:

"HalfSine"	half of a sine curve going from 0 to 1; good blend function for smooth entry and exit
"Linear"	linear line from 0 to 1

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.17 function

`diguyCharacterGesture::add_channel_weight_target`

Prototype:

```
int add_channel_weight_target(char channel,
    float channel_weight_target,
    float blend_tin,
    float blend_tout,
    const char* blend_function_name = 0);
```

Description:

This function is similar to `set_channel_weight_target()`, but adds the channel weight target to a queue of pending targets. The gesture will not begin moving the current channel weight to the specified target until all pending channel weight targets from previous calls to `set_channel_weight_target()` and `add_channel_weight_target()` have been reached.

Arguments:

`channel`

	which channel to affect; pass a single upper case character (e.g., 'A')
<code>channel_weight_target</code>	target channel weight; a value between 0 and 1
<code>blend_tin</code>	when, relative to the start of the gesture, the channel weight should begin moving toward <code>channel_weight_target</code>
<code>blend_tout</code>	when, relative to the start of the gesture, the channel weight should arrive at <code>channel_weight_target</code>
<code>blend_function_name</code>	name of the curve to use to move between current channel weight and target channel weight

Lua Example:

```
-- Create a gesture.
local gesture1 = character:create_gesture("example_gesture");

-- Set the starting channel weight of channel 'A' to 0.2.
gesture1:set_channel_weight('A', 0.2);

--
-- Set a channel weight target. This call essentially does this:
--
-- Smoothly change the channel A weight of the gesture from its
-- current value (0.2) to 0, starting at 1.5 seconds into the
-- gesture, finishing 3.5 seconds into the gesture.
--
gesture1:set_channel_weight_target(
    'A',          -- channel to change
    0.0,          -- target weight
    1.5,          -- when to start changing
    3.5,          -- when to stop changing
    "HalfSine");  -- curve to use for blend

--
-- Add another channel weight target. This call essentially does this:
--
-- Smoothly change the channel A weight of the gesture from its
-- last set channel weight (0) to 0.8, starting at 3.5 seconds into
-- the gesture, finishing 5.5 seconds into the gesture.
--
gesture1:add_channel_weight_target(
    'A',
    0.8,
    3.5,
    5.5,
    "HalfSine");

--
-- We're done tweaking the gesture. Set its automatic destroy flag
-- so that it will be deleted when the gesture has completed.
--
gesture1:set_automatic_destroy_flag(1);
```


Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.18 function diguyCharacterGesture::set_rampup_duration

Prototype:

```
int set_rampup_duration(float duration);
```

Description:

This function sets how long the gesture will take to "rampup" control of joints from the base motion to the gesture. A longer rampup will result in a smoother transition into the gesture.

If the rampup duration is longer than the first motion of the gesture, it will be shortened to be the same duration as the first motion. Motion motions are 0.5 seconds or shorter.

The default duration is 0.5 seconds.

Arguments:

`duration` rampup duration, in seconds

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.19 function diguyCharacterGesture::get_rampup_duration

Prototype:

```
float get_rampup_duration();
```

Returns:

the rampup duration as set by [set_rampup_duration\(\)](#)

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.20 function**diguyCharacterGesture::set_rampup_blend_function_name****Prototype:**

```
int set_rampup_blend_function_name(const char* blend_function_name);
```

Description:

This function sets which blend function will be used during rampup to transition control of joints from the base motion to the gesture.

The default blend function is "HalfSine".

Arguments:

`blend_function_name` new rampup blend function

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.21 function**diguyCharacterGesture::get_rampup_blend_function_name****Prototype:**

```
const char* get_rampup_blend_function_name();
```

Returns:

the rampup blend function as set by
[set_rampup_blend_function_name\(\)](#)

Callable For Gesture Types:

◇ all

Callable From:

◇ C++
 ◇ Script

5.14.22 function**diguyCharacterGesture::set_rampdown_duration****Prototype:**

```
int set_rampdown_duration(float duration);
```

Description:

This function sets how long the gesture will take to "rampdown" control of joints from the gesture to the base motion. A longer rampdown will result in a smoother transition out of the gesture.

Arguments:

`duration` rampdown duration, in seconds

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.23 function diguyCharacterGesture::get_rampdown_duration

Prototype:

```
float get_rampdown_duration();
```

Returns:

the rampdown duration as set by [set_rampdown_duration\(\)](#)

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.24 function

diguyCharacterGesture::set_rampdown_blend_function_name

Prototype:

```
int set_rampdown_blend_function_name(const char* blend_function_name);
```

Description:

This function sets which blend function will be used during rampdown to transition control of joints from the gesture to the base motion.

The default blend function is "HalfSine".

Arguments:

`blend_function_name` new rampdown blend function

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ all

Callable From:

◇ C++

◇ Script

5.14.25 function

diguyCharacterGesture::get_rampdown_blend_function_name

Prototype:

```
const char* get_rampdown_blend_function_name();
```

Returns:

the rampdown blend function as set by
[set_rampdown_blend_function_name\(\)](#)

Callable For Gesture Types:

◇ all

Callable From:

◇ C++
 ◇ Script

5.14.26 function

diguyCharacterGesture::get_default_blend_duration

Prototype:

```
float get_default_blend_duration();
```

Description:

This function returns the default blend duration. This is the amount of time the gesture will take to blend motions together to produce more seamless motions.

Returns:

default blend duration, in seconds

Callable For Gesture Types:

◇ all

Callable From:

- ◇ C++
- ◇ Script

Progression Gesture Functions

5.14.27 function diguyCharacterGesture::set_max_reps**Prototype:**

```
int set_max_reps(int max_reps,
                 int derive_duration_flag = 1);
```

Description:

This function sets the number of times the gesture will be repeated. The optimal overall duration can be optionally computed based on the number of repetitions specified.

Table Gestures:

Table gestures do not yet make use of the overall duration. The `derive_duration_flag` should be given a value of 0.

Arguments:

<code>max_reps</code>	number of times gesture will be repeated
<code>derive_duration_flag</code>	pass 1 to derive optimal overall duration from passed <code>max_reps</code>

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ progression

Callable From:

◇ C++
◇ Script

5.14.28 function diguyCharacterGesture::get_max_reps**Prototype:**

```
int get_max_reps();
```

Returns:

the number of times the gesture will be repeated

Callable For Gesture Types:

◇ progression

Callable From:

◇ C++

◇ Script

5.14.29 function diguyCharacterGesture::set_overall_duration**Prototype:**

```
int set_overall_duration(float overall_duration,
    int derive_max_reps_flag = 1);
```

Description:

This function sets the overall duration the gesture will have. The optimal number of repetitions can be optionally computed based on the specified duration.

Calling this function will override any previous calls that set specific stage durations (e.g., as set by a call to [set_stage_duration\(\)](#)).

Table Gestures:

Table gestures do not yet make use of the overall duration.

Arguments:

overall_duration

duration of all stages of the gesture, in seconds

`derive_max_reps_flag` pass 1 to derive optimal number of reps for the given duration

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ progression

Callable From:

◇ C++
◇ Script

5.14.30 function `diguyCharacterGesture::get_overall_duration`

Prototype:

```
float get_overall_duration();
```

Returns:

the overall duration in seconds of the gesture

Table Gestures:

Table gestures do not yet make use of the overall duration.

Callable For Gesture Types:

◇ progression

Callable From:

◇ C++
◇ Script

5.14.31 function diguyCharacterGesture::get_num_progression_stages

Prototype:

```
int get_num_progression_stages();
```

Returns:

the number of stages in the gesture

Callable For Gesture Types:

◇ progression

Callable From:

◇ C++

◇ Script

5.14.32 function diguyCharacterGesture::set_progression_stage_duration

Prototype:

```
int set_progression_stage_duration(int stage, float duration);
```

Description:

This function sets the exact duration of the specified stage of the gesture.

Arguments:

<code>stage</code>	stage; an integer between 1 and 3
<code>duration</code>	duration of stage, in seconds

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ progression

Callable From:

◇ C++

◇ Script

5.14.33 function diguyCharacterGesture::get_progression_stage_duration

Prototype:

```
float get_progression_stage_duration(int stage);
```

Returns:

the duration in seconds of specified stage of the gesture

Arguments:

stage stage; an integer between 1 and 3

Callable For Gesture Types:

◇ progression

Callable From:

◇ C++

◇ Script

5.14.34 function diguyCharacterGesture::get_progression_stage_optimal_duration

Prototype:

```
float get_progression_stage_optimal_duration(int stage);
```

Returns:

the optimal duration in seconds of specified stage of the gesture

Arguments:

stage stage; an integer between 1 and 3

Callable For Gesture Types:

◇ progression

Callable From:

◇ C++

◇ Script

Table Gesture Functions

5.14.35 function diguyCharacterGesture::set_table_action_desired

Prototype:

```
int set_table_action_desired(const char* action_name);
```

Description:

This function sets the desired action for table gestures. The gesture will begin a transition to the desired action as soon as it can.

The time required to reach the desired action can be determined by calling the [get_time_required_to_reach_table_action_desired\(\)](#) function.

Arguments:

`action_name` name of desired action

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ table

Callable From:

◇ C++

◇ Script

5.14.36 function diguyCharacterGesture::get_table_action_desired

Prototype:

```
const char* get_table_action_desired();
```

Description:

This function returns the desired action for table gestures.

If this gesture is not a table gesture the string "error" will be returned.

The desired action can be set using the
[set_table_action_desired\(\)](#) function.

Returns:

name of desired action, "error" on error

Callable For Gesture Types:

◇ table

Callable From:

◇ C++

◇ Script

5.14.37 function diguyCharacterGesture::get_time_required_to_reach_table_action

Prototype:

```
float get_time_required_to_reach_table_action_desired();
```

Description:

This function returns how much time it will take to reach the desired action for table gestures.

Returns:

time in seconds needed to reach desired action

Callable For Gesture Types:

◇ table

Callable From:

◇ C++
◇ Script

5.14.38 function diguyCharacterGesture::get_num_table_actions

Prototype:

```
int get_num_table_actions();
```

Description:

This function returns how many actions a table gesture contains. The function [get_table_action_at_index\(\)](#) can be called to get the name of the actions.

Returns:

number of actions in table gesture

Callable For Gesture Types:

◇ table

Callable From:

◇ C++

◇ Script

5.14.39 function diguyCharacterGesture::get_table_action_at_index

Prototype:

```
const char* get_table_action_at_index(int index);
```

Description:

This function returns the name of the table action at the specified index.

Returns:

name of table action; NULL if no action at given index

Callable For Gesture Types:

◇ table

Callable From:

◇ C++

◇ Script

5.14.40 function diguyCharacterGesture::get_table_action_rep_duration

Prototype:

```
float get_table_action_rep_duration(const char* action_name);
```

Description:

This function returns how long a single repetition of a table action takes.

Arguments:

`action_name` name of action

Returns:

duration of single rep, in seconds

Callable For Gesture Types:

◇ table

Callable From:

- ◇ C++
- ◇ Script

5.14.41 function diguyCharacterGesture::get_table_transition_duration

Prototype:

```
float get_table_transition_duration(const char* action_from,
                                   const char* action_to);
```

Description:

This function returns how long the transition motion(s) take between two table actions.

Arguments:

`action_from` name of origin action
`action_to` name of destination action

Returns:

duration of transition, in seconds

Callable For Gesture Types:

- ◇ table

Callable From:

- ◇ C++
- ◇ Script

5.14.42 function diguyCharacterGesture::set_table_action_begin**Prototype:**

```
int set_table_action_begin(const char* action_name);
```

Description:

This function sets which action a table gesture should begin at when [begin_now\(\)](#) is called. The gesture's first motion will be the transition motion from the beginning action to the desired action if the actions differ, or the looping motion of the desired action if they are the same.

The default beginning action is the action at index 0.

Arguments:

`action_name` name of beginning action

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ table

Callable From:

◇ C++

◇ Script

5.14.43 function diguyCharacterGesture::get_table_action_begin

Prototype:

```
const char* get_table_action_begin();
```

Returns:

the name of the beginning action of a table gesture, as set by [set_table_action_begin\(\)](#)

Callable For Gesture Types:

◇ table

Callable From:

◇ C++

◇ Script

5.14.44 function diguyCharacterGesture::set_table_action_end**Prototype:**

```
int set_table_action_end(const char* action_name);
```

Description:

This function sets which action a table gesture should end with when `end_gesture()` is called. The default ending action is the action at index 0.

Arguments:

`action_name` name of ending action

Returns:

0 on success, -1 on failure

Callable For Gesture Types:

◇ table

Callable From:

◇ C++
◇ Script

5.14.45 function diguyCharacterGesture::get_table_action_end**Prototype:**

```
const char* get_table_action_end();
```

Returns:

the name of the ending action of a table gesture, as set by [set_table_action_end\(\)](#)

Callable For Gesture Types:

◇ table

Callable From:

◇ C++

◇ Script

5.14.46 function diguyCharacterGesture::get_table_action_now**Prototype:**

```
const char* get_table_action_now();
```

Returns:

the name of the action the table gesture is currently playing

Callable For Gesture Types:

◇ table

Callable From:

◇ C++

◇ Script

5.14.47 function diguyCharacterGesture::get_table_action_next**Prototype:**

```
const char* get_table_action_next();
```

Returns:

the name of the action the table gesture will play next

Callable For Gesture Types:

◇ table

Callable From:

◇ C++

◇ Script

5.14.48 function diguyCharacterGesture::force_table_action**Prototype:**

```
int force_table_action(const char* action_name,
    int include_transition_arc = 1,
    float max_rampdown_interval = 0.5f);
```

Description:

This function forces the current action of the table gesture to be the action identified by action_name.

Arguments:

action_name	name of the action to be performed by the gesture
include_transition_arc	flag for whether the transition motion from the current action to the new action is included; pass 1 for best chance of good looking transition; pass 0 for fastest response

`max_rampdown_interval` max amount of time spent trying to smooth over any motion seams; set to 0.5 for a smooth transition, set to 0 for a potentially rough transition

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Callback Functions

5.14.49 enumeration

```
enum {
    CALLBACK_ID_CREATE = 1,
    CALLBACK_ID_DESTROY,
    CALLBACK_ID_ATTAINED_DESIRED_ACTION
};
```

Description:

This is an enumeration of the different callbacks that can be registered with [add_callback\(\)](#) and [add_callback_script\(\)](#).

Usable From:

- ◇ C++
- ◇ Script

5.14.50 function diguyCharacterGesture::add_callback

Prototype:

```
int add_callback(int callback_id,
                diguyCharacterGestureCallback* callback,
                void* callback_params = 0,
                void* callback_user_data = 0);
```

Description:

This function adds a user callback.

Arguments:

callback	pointer to function with prototype diguyCharacterGestureCallback (typedefed above)
callback_id	integer id of when this callback is to be called
callback_params	not currently used; pass NULL
callback_user_data	pointer for user's own use; DI-Guy will do nothing to the contents of this pointer beyond passing it back when the callback is invoked

callback_id should be one of the following values:

CALLBACK_ID_CREATE

This callback will be called when a new gesture is created.

CALLBACK_ID_DESTROY

This callback will be called when the gesture is destroyed.

CALLBACK_ID_ATTAINED_DESIRED_ACTION

This callback will be called when a table gesture has reached its desired action as set by [set_table_action_desired\(\)](#).

Callback Return Values:

Callbacks return a value of type [diguyCallbackReturn](#), which will be [DIGUY_CALLBACK_STOP](#) or [DIGUY_CALLBACK_CONTINUE](#). If the callback returns [DIGUY_CALLBACK_STOP](#), the default handler of the function will not be called; the callback is asserting that it has done everything necessary for the function call. If the callback returns [DIGUY_CALLBACK_CONTINUE](#), the default handler for the function will be called after the callback.

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.14.51 function diguyCharacterGesture::remove_callback

Prototype:

```
int remove_callback(int callback_id,
                    diguyCharacterGestureCallback* callback);
```

Description:

This function removes a user callback. All callbacks matching the specified `callback_id` and callback function will be removed.

Arguments:

<code>callback_id</code>	integer id of when this callback is to be called
<code>callback</code>	pointer to function with prototype diguyCharacterGestureCallback (typedefed above)

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.14.52 function diguyCharacterGesture::remove_callback_with_user_data

Prototype:

```
int remove_callback_with_user_data(int callback_id,
    void* callback_user_data);
```

Description:

This function removes a user callback. All callbacks matching the specified `callback_id` and `callback_user_data` pointer will be removed.

Arguments:

<code>callback_id</code>	integer id of when this callback is to be called
<code>callback_user_data</code>	pointer for user's own use

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.14.53 function diguyCharacterGesture::add_callback_script**Prototype:**

```
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function adds a user callback script. Callback scripts can be removed with [remove_callback_script\(\)](#).

See [diguyCharacter::add_callback_script\(\)](#) for an example of use.

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback to be added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

lua specific:

When the script is called, the object for which it is being called will be in the `$callback_object` scalar.

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.14.54 function diguyCharacterGesture::remove_callback_script

Prototype:

```
int remove_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function removes a user callback script previously added with [add_callback_script\(\)](#).

See [diguyCharacter::remove_callback_script\(\)](#) for an example of use.

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback previously added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script`, all callback scripts whose ids match `callback_id` and whose types match `callback_script_type` will be removed.

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

lua specific:

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Deprecated Functions

The following functions are deprecated.

As these functions may disappear in a future version of DI-Guy, it is recommended that you replace calls to these functions from your application.

5.14.55 function diguyCharacterGesture::get_num_stages**Prototype:**

```
int get_num_stages();
```

Description:

This function is deprecated. Use [get_num_progression_stages\(\)](#) instead.

5.14.56 function diguyCharacterGesture::set_stage_duration**Prototype:**

```
int set_stage_duration(int stage, float duration);
```

Description:

This function is deprecated. Use [set_progression_stage_duration\(\)](#) instead.

5.14.57 function diguyCharacterGesture::get_stage_duration**Prototype:**

```
float get_stage_duration(int stage);
```

Description:

This function is deprecated. Use `get_progression_stage_duration()` instead.

5.14.58 function diguyCharacterGesture::get_stage_optimal_duration**Prototype:**

```
float get_stage_optimal_duration(int stage);
```

Description:

This function is deprecated. Use `get_progression_stage_optimal_duration()` instead.

5.15 diguyCharacterGestureType Type Declaration

diguyCharacterGestureType.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

5.15.1 typedef diguyCharacterGestureType

```
typedef int diguyCharacterGestureType;
```

5.15.2 enumeration

```
enum {  
    DIGUY_CHARACTER_GESTURE_TYPE_UNKNOWN = 0,  
    DIGUY_CHARACTER_GESTURE_TYPE_PROGRESSION,  
    DIGUY_CHARACTER_GESTURE_TYPE_TABLE,  
    DIGUY_CHARACTER_GESTURE_TYPE_COUNT  
};
```

5.15.3 function diguy_gesture_type_to_string

Prototype:

```
const char* diguy_gesture_type_to_string(diguyCharacterGestureType gesture_type);
```

Description:

This function converts the passed gesture type enumeration into a NULL-terminated C-style string.

Returns:

C-style string identifying gesture type

Callable From:

- ◇ C++
- ◇ Script

5.15.4 function diguy_gesture_type_from_string

Prototype:

```
diguyCharacterGestureType diguy_gesture_type_from_string(const char* string);
```

Description:

This function converts the passed C-style string into an enumerated value.

Returns:

enumeration value of gesture type

Callable From:

- ◇ C++
- ◇ Script

5.16 diguyCharacterGroup

diguyCharacterGroup.h

DI-Guy SDK Version 12.5.2

Includes: stdio.h | declspec_diguy.h

Contents:

- ◆ [Label Functions](#)
- ◆ [Variable Functions](#)
- ◆ [Experimental Functions](#)

5.16.1 class diguyCharacterGroup

```
class BDI_DECLSPEC_diguy diguyCharacterGroup
{
public:
    const char* get_name();
    int set_name(const char* name);
    int add_member(diguyCharacter* character);
    int add_all_characters_as_members();
    int remove_member(diguyCharacter* character);
    int remove_all_members();
    int is_member(diguyCharacter* character);
    int get_num_members();
    diguyCharacter* get_member_at_index(int index);
    int get_index_of_member(diguyCharacter* character);
    int get_subgroup_number(diguyFormation* formation, int index);
    int get_index_in_subgroup(diguyFormation* formation, int index);
    int set_all_members_enabled(int enabled);
    int update_all_members(float t);
#ifdef CPLUSPLUS_ONLY
    int draw_all_members();
    int draw_pass1_all_members();
    int draw_pass2_all_members();
#endif
    int set_all_members_current_tin(float tin);
    int set_all_members_current_tout(float tout);
    diguyViewLabel* get_label();
    void set_label_offset_above_centroid(float offset);
    float get_label_offset_above_centroid();
    int get_num_variables();
    diguyVariable* get_variable_at_index(int index);
    diguyVariable* find_variable(const char* name);
    diguyVariable* find_or_create_variable(const char* name);
    int destroy_variable(diguyVariable* variable);
    int translate_all_paths(float tx, float ty, float tz);
    int rotate_all_paths_about_point(float rz, float rx, float ry,
        float rotation_pt_x, float rotation_pt_y, float rotation_pt_z);
    int rotate_all_paths_about_current_waypoint(float rz, float rx, float ry);
    diguyCharacter* get_nearest_character_to_point(float x, float y, float z);
    diguyCharacter* get_farthest_character_to_point(float x, float y, float z);
    int get_group_centroid(float* x, float* y, float* z);
};
```

```

int get_group_min_circle(float* x, float* y, float* z, float* radius);
int send_message_to_all_members(const char* sender,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);
int apply_formation_to_group_with_dir(diguyCharacter* leader,
    diguyFormation* formation,
    float dir_x,
    float dir_y,
    float desired_radius = -1.0f,
    int require_sort = 1);
int apply_formation_to_group_with_goal(diguyCharacter* leader,
    diguyFormation* formation,
    float goal_x,
    float goal_y,
    float desired_radius = -1.0f,
    int require_sort = 1);
int apply_formation_to_group(diguyCharacter* leader,
    diguyFormation* formation,
    float desired_radius = -1.0f,
    int require_sort = 1);
diguyFormation* get_current_formation();
int sort_group_based_on_pos(float x,
    float y,
    float z,
    diguyFormation* formation = NULL,
    diguyCharacter* leader = NULL);
diguyCharacter* get_sorted_member_at_index(int index);
int get_num_active_members();

```

5.16.2 function diguyCharacterGroup::get_name

Prototype:

```
const char* get_name();
```

Returns:

name of the group; this value will never be NULL

Callable From:

- ◇ C++
- ◇ Script

5.16.3 function diguyCharacterGroup::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.16.4 function diguyCharacterGroup::add_member

Prototype:

```
int add_member(diguyCharacter* character);
```

Description:

Adds the passed character to the group if not already in it.

Returns:

0 if the character is now a part of the group, -1 if not

Callable From:

- ◇ C++
- ◇ Script

5.16.5 function diguyCharacterGroup::add_all_characters_as_members

Prototype:

```
int add_all_characters_as_members();
```

Description:

Adds all characters of the scenario as members of the group.

Returns:

Always return 0.

Callable From:

- ◇ C++
- ◇ Script

5.16.6 function diguyCharacterGroup::remove_member

Prototype:

```
int remove_member(diguyCharacter* character);
```

Description:

Removes the passed character from the group if it is a member.

Returns:

0 if the character was part of the group and removed; -1 if not

Callable From:

- ◇ C++
- ◇ Script

5.16.7 function diguyCharacterGroup::remove_all_members

Prototype:

```
int remove_all_members();
```

Description:

Removes all members of the group.

Returns:

Always return 0.

Callable From:

- ◇ C++
- ◇ Script

5.16.8 function diguyCharacterGroup::is_member

Prototype:

```
int is_member(diguyCharacter* character);
```

Description:

The function can be used to check if a specific character is in this group.

Also see [diguyCharacter::is_group_member\(\)](#).

Returns:

1 if the passed character is a member of the group; 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.16.9 function diguyCharacterGroup::get_num_members

Prototype:

```
int get_num_members();
```

Returns:

number of member characters in group

Callable From:

- ◇ C++
- ◇ Script

5.16.10 function diguyCharacterGroup::get_member_at_index

Prototype:

```
diguyCharacter* get_member_at_index(int index);
```

Returns:

group member at specified index; NULL if no character at specified index

Callable From:

- ◇ C++
- ◇ Script

5.16.11 function diguyCharacterGroup::get_index_of_member**Prototype:**

```
int get_index_of_member(diguyCharacter* character);
```

Returns:

index of member, or -1 if not found

Callable From:

- ◇ C++
- ◇ Script

5.16.12 function diguyCharacterGroup::get_subgroup_number**Prototype:**

```
int get_subgroup_number(diguyFormation* formation, int index);
```

Description:

See [diguyFormation](#) for a description of subgroups.

Returns:

subgroup number within formation of character at index, or -1

Callable From:

- ◇ C++
- ◇ Script

5.16.13 function diguyCharacterGroup::get_index_in_subgroup**Prototype:**

```
int get_index_in_subgroup(diguyFormation* formation, int index);
```

Description:

See [diguyFormation](#) for a description of subgroups.

Returns:

index within subgroup, or 0

Callable From:

- ◇ C++
- ◇ Script

5.16.14 function diguyCharacterGroup::set_all_members_enabled**Prototype:**

```
int set_all_members_enabled(int enabled);
```

Description:

Sets whether all characters in group are enabled. Equivalent to calling [diguyCharacter::set_enabled\(\)](#) for group members.

Returns:

Always return 0.

Callable From:

- ◇ C++
- ◇ Script

5.16.15 function diguyCharacterGroup::update_all_members**Prototype:**

```
int update_all_members(float t);
```

Description:

Updates all characters in the group. Equivalent to calling `diguyCharacter::update( )` for all group members.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.16.16 function diguyCharacterGroup::draw_all_members**Prototype:**

```
int draw_all_members();
```

Description:

This function draws all members of the group. See `diguyCharacter::draw( )`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++

5.16.17 function `diguyCharacterGroup::draw_pass1_all_members`

Prototype:

```
int draw_pass1_all_members();
```

Description:

This function draws pass 1 of all members of the group. See [`diguyCharacter::draw\_pass1\(\)`](#).

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.16.18 function `diguyCharacterGroup::draw_pass2_all_members`

Prototype:

```
int draw_pass2_all_members();
```

Description:

This function draws pass 2 all members of the group. See [`diguyCharacter::draw\_pass2\(\)`](#).

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.16.19 function diguyCharacterGroup::set_all_members_current_tin

Prototype:

```
int set_all_members_current_tin(float tin);
```

Description:

This function sets the current tin time of all characters in the group. See [diguyCharacter::set_current_tin\(\)](#).

Arguments:

tin	new value of current tin in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.16.20 function diguyCharacterGroup::set_all_members_current_tout

Prototype:

```
int set_all_members_current_tout(float tout);
```

Description:

This function sets the current tout time of all characters in the group. See [diguyCharacter::set_current_tout\(\)](#).

Arguments:

tout	new value of current tout in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Label Functions

The functions in this section control the text, visibility, and colors of group labels.

Currently group labels are only enabled in DI-Guy Scenario.

5.16.21 function diguyCharacterGroup::get_label

Prototype:

```
diguyViewLabel* get_label();
```

Description:

This function returns a pointer to the label object of the group.

Currently group labels are only enabled in DI-Guy Scenario.

Returns:

pointer to the group's label

Callable From:

- ◇ C++
- ◇ Script

5.16.22 function diguyCharacterGroup::set_label_offset_above_centroid

Prototype:

```
void set_label_offset_above_centroid(float offset);
```

Description:

This function sets how high the label of the group will float above the group's centroid.

Arguments:

offset	offset, in meters, of label above group

Callable From:

- ◇ C++
- ◇ Script

5.16.23 function**diguyCharacterGroup::get_label_offset_above_centroid****Prototype:**

```
float get_label_offset_above_centroid();
```

Returns:

how high the label of the group will float above the group centroid

Callable From:

- ◇ C++
- ◇ Script

Variable Functions

5.16.24 function diguyCharacterGroup::get_num_variables**Prototype:**

```
int    get_num_variables();
```

Returns:

number of variables the group has

Callable From:

- ◇ C++
- ◇ Script

5.16.25 function diguyCharacterGroup::get_variable_at_index**Prototype:**

```
diguyVariable* get_variable_at_index(int index);
```

Returns:

pointer of type [diguyVariable](#); NULL if no variable at the specified index

Arguments:

index index of the variable; indices start at 0

Callable From:

- ◇ C++
- ◇ Script

5.16.26 function diguyCharacterGroup::find_variable**Prototype:**

```
diguyVariable* find_variable(const char* name);
```

Description:

This function returns a pointer to the specified variable.

Arguments:

name	name of variable to be found

Returns:

pointer of type [diguyVariable](#); NULL if not found

Callable From:

- ◇ C++
- ◇ Script

5.16.27 function diguyCharacterGroup::find_or_create_variable**Prototype:**

```
diguyVariable* find_or_create_variable(const char* name);
```

Description:

This function finds the variable with the given name or creates it if it doesn't exist.

Arguments:

name	name of the variable to find or create

Returns:

pointer of type [diguyVariable](#); should never be NULL

Callable From:

- ◇ C++

◇ Script

5.16.28 function diguyCharacterGroup::destroy_variable

Prototype:

```
int destroy_variable(diguyVariable* variable);
```

Description:

This function destroys the passed variable.

Arguments:

`variable` pointer to a [diguyVariable](#)

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

Experimental Functions

5.16.29 function diguyCharacterGroup::translate_all_paths

Prototype:

```
int translate_all_paths(float tx, float ty, float tz);
```

5.16.30 function**diguyCharacterGroup::rotate_all_paths_about_point****Prototype:**

```
int rotate_all_paths_about_point(float rz, float rx, float ry,  
    float rotation_pt_x, float rotation_pt_y, float rotation_pt_z);
```

5.16.31 function**diguyCharacterGroup::rotate_all_paths_about_current_waypoint****Prototype:**

```
int rotate_all_paths_about_current_waypoint(float rz, float rx, float ry);
```


5.16.32 function diguyCharacterGroup::get_nearest_character_to_point

Prototype:

```
diguyCharacter* get_nearest_character_to_point(float x, float y, float z);
```

Description:

Get the character closest to the specified point.

Arguments:

`x, y, z` position in meters from the origin

Returns:

pointer to [diguyCharacter](#), or NULL if none found

Callable From:

◇ C++
◇ Script

5.16.33 function diguyCharacterGroup::get_farthest_character_to_point

Prototype:

```
diguyCharacter* get_farthest_character_to_point(float x, float y, float z);
```

Description:

Get the character furthest from the specified point.

Arguments:

`x, y, z` position in meters from the origin

Returns:

pointer to [diguyCharacter](#), or NULL if none found

Callable From:

◇ C++

◇ Script

5.16.34 function diguyCharacterGroup::get_group_centroid

Prototype:

```
int get_group_centroid(float* x, float* y, float* z);
```

Description:

Get centroid of group (average of character positions).

Arguments:

`x, y, z` position in meters from the origin

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.16.35 function diguyCharacterGroup::get_group_min_circle**Prototype:**

```
int get_group_min_circle(float* x, float* y, float* z, float* radius);
```

Description:

Get minimum enclosing circle of group. Note that this is slower than getting the centroid.

Arguments:

x, y, z	center of circle of group in meters from the origin
radius	radius of circle

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.16.36 function diguyCharacterGroup::send_message_to_all_members**Prototype:**

```
int send_message_to_all_members(const char* sender,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);
```

Description:

Send a message to all characters in group. See [diguyCharacter::agent_accept_message\(\)](#).

Arguments:

sender	who is sending the message
message_type	what type of message is being sent
message	the message string
message_params	optional parameter string

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.16.37 function**diguyCharacterGroup::apply_formation_to_group_with_dir****Prototype:**

```
int apply_formation_to_group_with_dir(diguyCharacter* leader,
    diguyFormation* formation,
    float dir_x,
    float dir_y,
    float desired_radius = -1.0f,
    int require_sort = 1);
```

Description:

Puts the group into the specified formation, applying agent_pursue_character_with_offset() to each character except the leader.

Arguments:

leader	leader; can be NULL
formation	name of formation
dir_x	x component of the formation's heading

`dir_y` y component of the formation's heading
`desired_radius` if not given (-1.0), get radius from formation
`require_sort` if 1, group must be resorted; if 0, it may still be

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
 ◇ Script

5.16.38 function `diguyCharacterGroup::apply_formation_to_group_with_goal`

Prototype:

```
int apply_formation_to_group_with_goal(diguyCharacter* leader,
    diguyFormation* formation,
    float goal_x,
    float goal_y,
    float desired_radius = -1.0f,
    int require_sort = 1);
```

Description:

Puts the group into the specified formation, applying
`agent_pursue_character_with_offset( )` to each character except the leader.

Arguments:

<code>leader</code>	leader; can be NULL
<code>formation</code>	name of formation
<code>goal_x</code>	x component of the formation's goal point

`goal_y` y component of the formation's goal point
`desired_radius` if not given (-1.0), get radius from formation
`require_sort` if 1, group must be resorted; if 0, it may still be

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
 ◇ Script

5.16.39 function `diguyCharacterGroup::apply_formation_to_group`

Prototype:

```
int apply_formation_to_group(diguyCharacter* leader,
                             diguyFormation* formation,
                             float desired_radius = -1.0f,
                             int require_sort = 1);
```

Description:

Puts the group into the specified formation, applying
`agent_pursue_character_with_offset( )` to each character except the leader.

Arguments:

`leader` leader; can be NULL
`formation` name of formation
`desired_radius` if not given (-1.0), get radius from formation
`require_sort` if 1, group must be resorted; if 0, it may still be

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.16.40 function diguyCharacterGroup::get_current_formation

Prototype:

```
diguyFormation* get_current_formation();
```

Description:

Returns the last-set formation

Returns:

The formation or NULL

Callable From:

- ◇ C++
- ◇ Script

5.16.41 function

diguyCharacterGroup::sort_group_based_on_pos

Prototype:

```
int sort_group_based_on_pos(float x,
    float y,
    float z,
    diguyFormation* formation = NULL,
    diguyCharacter* leader = NULL);
```

Description:

Sorts group relative to the point given. The point should be on the left or in the front, depending on what order the formation is in. This updates an internal list of sorted group members.

Arguments:

x	x component of point
y	y component of point
z	z component of point
formation	the formation; pass NULL for no formation
leader	the leader; pass NULL for no formation leader

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.16.42 function `diguyCharacterGroup::get_sorted_member_at_index`

Prototype:

```
diguyCharacter* get_sorted_member_at_index(int index);
```

Returns:

member of formation-sorted group; NULL if no character at index

Callable From:

- ◇ C++
- ◇ Script

5.16.43 function `diguyCharacterGroup::get_num_active_members`

Prototype:

```
int get_num_active_members();
```

Returns:

number of active members

Callable From:

- ◇ C++
- ◇ Script

5.17 diguyCharacterGuide

diguyCharacterGuide.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

- [Guide Algorithms](#)
- [Exact](#)
- [Follow1](#)
- [Follow2](#)
- [Drift1](#)
- ◆ [General Functions](#)

[diguyCharacterGuide](#) Overview

A guide is something that can influence the position, orientation, or action of a character that is in free position mode and free action mode.

Most guides will look at the desired position and orientation of the character and do what is necessary to get the character there (or at least closer). How a guide accomplishes this depends on:

- ◆ the guide algorithm being used
- ◆ the parameters set for that algorithm
- ◆ the difference between current and desired settings

The desired position and orientation for a character can be explicitly set by the calls `set_desired_position( )` and `set_desired_orientation( )`. They may also be implicitly set if that character is in a formation.

Guides can be explicitly added to a character by calling `add_guide( )` or `create_guide( )`. They can also be implicitly added to a character if that character is called into a formation.

Guide parameters are set by calls to `diguyCharacterGuide::set_guide_algorithm_float_parameter( )`.

Once a guide has been added to a character it can potentially affect the position, orientation, and/or action of a character until the guide has been removed. Guides can be explicitly removed from a character by calling `remove_guide( )` or `remove_all_guides( )`. They can be implicitly removed by the break-up of a formation, or by an automatic removal guide acquiring the target destination.

Guide Algorithms

Below are descriptions of each guide algorithm, including an overview, its effects (if any) on action, position, and orientation, and the parameters that can be set for the algorithm.

Exact

Overview:

This is a trivial guide algorithm that simply sets the current position and orientation of the character to be equal to the desired position and orientation.

It can be used with or without a formation.

Effects on Action:

If the character is following a formation leader, and the leader has the same character type as the character, the character will match its action to that of the leader.

If the character types do not match, the character will try to use a suitable alternative action.

If the character is not in a formation, the action of the character is not affected.

Effects on Position and Orientation:

The character's position and orientation will be set to exactly equal the desired values without delay. If there is a substantial difference between current and desired values, a large jump in will be evident.

Parameters:

(none)

Follow1

Overview:

The Follow1 guide algorithm attempts to reach the desired position by changing the character's action and orientation.

The Follow1 algorithm is commonly used by followers in formations. It can also be used by characters not in a formation.

Effects on Action:

If the character is in distance_zone_0 and is following a formation leader that is stopped, the character will match its action to that of the leader. Otherwise characters in distance_zone_0 will switch to their default still action (usually stand).

DI-Guy SDK Version 12.5.2 Index

If the character is outside distance_zone_0 and is following a formation leader that is moving, the character will match its action to that of the leader. Otherwise characters outside distance_zone_0 will switch to their default moving action (usually walk).

If the character is outside distance_zone_1 the character will scale its speed by up to max_speed_scale.

Effects on Position:

There is no direct effect on position. The character attempts to reach its desired position by changing its current action and orientation.

Effects on Orientation:

If the character is in distance_zone_0 there is no effect on orientation.

If the character is beyond distance_zone_0 the character's orientation will be set to turn the character directly toward the desired position.

Note that currently the desired orientation is not used; instead the character orients toward the desired position.

Parameters:

max_speed_scale (description pending)

distance_zone_0 (description pending)

distance_zone_1 (description pending)

Follow2

Overview:

The Follow2 guide algorithm attempts to reach the desired position by changing the character's action and orientation.

The Follow2 algorithm is commonly used by followers in formations. It can also be used by characters not in a formation.

Effects on Action:

If following a formation leader:

- In distance_zone_0, leader stopped: Character will stop and match its still action to that of the leader if

- they are the same character type, otherwise the character will pick a different still action.
- In distance_zone_0, leader moving: Character will stop and use its default still action.
 - In distance_zone_1, leader stopped: Character will use an appropriate moving action.
 - In distance_zone_1, leader moving: Character will match its action to that of the leader.
 - Beyond distance_zone_1, leader stopped or moving: Character will select a fast moving action, or speed up a slower action.

If not in a formation:

- In distance_zone_0: Character will stop using its default still action.
- In distance_zone_1: Character will use an appropriate moving action.
- Beyond distance_zone_1: Character will select a fast moving action, or speed up a slower action.

Effects on Position:

There is no direct effect on position. The character attempts to reach its desired position by changing its current action and orientation.

Effects on Orientation:

In general, if the character is moving it will turn toward its desired position. If stopped, the character will not turn.

Note that currently the desired orientation is not used; instead the character orients toward the desired position.

Parameters:

- distance_zone_0 distance zone 0 radius; default 0.5 meters
- distance_zone_1 distance zone 1 radius; default 2.0 meters
- max_speed_scale determines how much character can speed up outside distance_zone_1

Drift1

Overview:

The Drift1 guide algorithm servos the character toward its desired position and orientation, effectively "scooting" the

character toward where it should be.

The Drift1 algorithm can be used by formation followers, and can also be used for characters that are trying to match desired positions and orientations from outside sources.

Effects on Action:

If the character is following a formation leader, and the leader has the same character type as the character, the character will match its action to that of the leader.

If the character is not in a formation, the action of the character is not affected.

Effects on Position:

Drift 1 distance zones are spherical, centered on the character's current position.

If the character is in distance_zone_0, the character is considered "close enough", and no drifting occurs.

If the character is in distance_zone_1, the character will drift toward the desired position. The magnitude of the drift is proportional to the parameter position_time_constant.

If the character is in distance_zone_2 or higher, the character is considered to be too far away and will be teleported directly to the desired position and orientation.

Effects on Orientation:

Drift1 azimuth zones are symmetrical, extending from the character's forward direction symmetrically around to its left and right.

If the desired position is in azimuth_zone_0, the character is considered "close enough", and no drifting occurs.

If the desired position is in azimuth_zone_1, the character will drift toward the desired orientation. The magnitude of the drift is proportional to the parameter orientation_time_constant.

If the desired position is in azimuth_zone_2 or higher, the character is considered to be too far away and will be immediately reoriented directly toward the desired orientation.

Parameters:

<code>distance_zone_0</code>	distance zone 0 radius; default 0.0 meters
<code>distance_zone_1</code>	distance zone 1 radius; default 4.0 meters
<code>azimuth_zone_0</code>	azimuth zone 0 extent; default 0.0 degrees
<code>azimuth_zone_1</code>	azimuth zone 1 extent; default 180.0 degrees
<code>position_time_constant</code>	higher value results in slower drift; default is 1.0
<code>orientation_time_constant</code>	higher value results in slower drift; default is 1.0

5.17.1 class `diguyCharacterGuide`

```
class BDI_DECLSPEC diguy diguyCharacterGuide
{
public:
    const char* get_name();
    int set_name(const char* name);
    int set_enabled(int enabled);
    int get_enabled();
    int get_position_acquired();
    void set_position_acquired(int acquired);
    int get_orientation_acquired();
    void set_orientation_acquired(int acquired);
    int set_guide_algorithm_float_parameter(const char* parameter,
        float value);
    float get_guide_algorithm_float_parameter(const char* parameter);
    int set_guide_algorithm_string_parameter(const char* parameter,
        const char * value);
    const char * get_guide_algorithm_string_parameter(const char* parameter);
```

General Functions

All functions are Callable From:

- ◆ C++
- ◆ Script

5.17.2 function diguyCharacterGuide::get_name

Prototype:

```
const char* get_name();
```

Returns:

name of the guide

5.17.3 function diguyCharacterGuide::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

5.17.4 function diguyCharacterGuide::set_enabled

Prototype:

```
int set_enabled(int enabled);
```

Description:

This function enables or disables this guide.

Arguments:

enabled pass 1 to enable to guide, 0 to disable

Returns:

0 on success, -1 on failure

5.17.5 function diguyCharacterGuide::get_enabled

Prototype:

```
int get_enabled();
```

Returns:

whether this guide is enabled; see [set_enabled\(\)](#)

5.17.6 function diguyCharacterGuide::get_position_acquired

Prototype:

```
int get_position_acquired();
```

Description:

This function returns whether the character controlled by this guide has reached its desired position.

Returns:

1 if position reached, 0 if not

5.17.7 function diguyCharacterGuide::set_position_acquired

Prototype:

```
void set_position_acquired(int acquired);
```

Description:

This function manually sets whether the character controlled by this guide has reached its desired position. Normally this is determined automatically by the guide.

Note that calling this function may trigger a callback of diguyCharacter::CALLBACK_ID_GUIDE_POSITION_ACQUIRED or diguyCharacter::CALLBACK_ID_GUIDE_POSITION_UNACQUIRED.

Arguments:

<code>acquired</code>	pass 1 to set the desired position as acquired, 0 as unacquired

5.17.8 function diguyCharacterGuide::get_orientation_acquired

Prototype:

```
int get_orientation_acquired();
```

Description:

This function returns whether the character controlled by this guide has reached its desired orientation.

Returns:

1 if orientation reached, 0 if not

5.17.9 function diguyCharacterGuide::set_orientation_acquired**Prototype:**

```
void set_orientation_acquired(int acquired);
```

Description:

Similar to `set_position_acquired()`, but for orientation.

Arguments:

<code>acquired</code>	pass 1 to set the desired orientation as acquired, 0 as unacquired

5.17.10 function diguyCharacterGuide::set_guide_algorithm_float_parameter**Prototype:**

```
int set_guide_algorithm_float_parameter(const char* parameter,
float value);
```

Description:

This function sets a parameter for this guide's guide algorithm.

Arguments:

<code>parameter</code>	name of the parameter to set
<code>value</code>	new value of the parameter

Returns:

0 on success, -1 on failure

5.17.11 function diguyCharacterGuide::get_guide_algorithm_float_parameter

Prototype:

```
float get_guide_algorithm_float_parameter(const char* parameter);
```

Description:

This function gets the current setting for a parameter of this guide's guide algorithm.

Arguments:

parameter name of the parameter to get

Returns:

current value of parameter

5.17.12 function diguyCharacterGuide::set_guide_algorithm_string_parameter

Prototype:

```
int set_guide_algorithm_string_parameter(const char* parameter,  
const char * value);
```

Description:

This function sets a parameter for this guide's guide algorithm.

Arguments:

parameter name of the parameter to set
value new value of the parameter

Returns:

0 on success, -1 on failure

5.17.13 function `diguyCharacterGuide::get_guide_algorithm_string_parameter`

Prototype:

```
const char * get_guide_algorithm_string_parameter(const char* parameter);
```

Description:

This function gets the current setting for a parameter of this guide's guide algorithm.

Arguments:

`parameter` name of the parameter to get

Returns:

current value of parameter

5.18 diguyCharacterPath

diguyCharacterPath.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Waypoint Functions](#)
- ◆ [Action Bead Functions](#)
- ◆ [Script Bead Functions](#)
- ◆ [Decision Bead Functions](#)
- ◆ [Gaze Bead Functions](#)
- ◆ [Aim Bead Functions](#)
- ◆ [Callback Functions](#)
- ◆ [Experimental Functions](#)

Link against: **libdiguy**

5.18.1 class diguyCharacterPath

```
class BDI_DECLSPEC_diguy diguyCharacterPath
{
public:
    const char* get_name();
    int set_name(const char* name);
    long get_uid();
    diguyCharacter* get_character();
    int update();
    float get_length();
    float get_duration();
    int get_point_at_distance(float distance_into_path,
        float* x, float* y, float* z,
        float* rz, float* rx, float* ry);
    diguyPathShape* get_path_shape();
    int get_num_waypoints();
    diguyWaypoint* get_waypoint_at_index(int index);
    diguyWaypoint* find_waypoint(const char* name);
    diguyWaypoint* create_waypoint(float x = 0.0f, float y = 0.0f, float z = 0.0f,
        float yaw = 0.0f, float roll = 0.0f, float pitch = 0.0f,
        float weight = 1.0f,
        int index = -1);
    int add_waypoint(diguyWaypoint* waypoint);
    int get_num_action_beads();
    diguyCharacterPathActionBead* get_action_bead_at_index(int index);
    diguyCharacterPathActionBead* find_action_bead(const char* name);
    diguyCharacterPathActionBead* get_final_action_bead();
    diguyCharacterPathActionBead* create_action_bead(const char* action,
        int index = -1);
    int destroy_action_bead(diguyCharacterPathActionBead* action_bead);
    int get_num_script_beads();
};
```

```

diguyCharacterPathScriptBead* get_script_bead_at_index(int index);
diguyCharacterPathScriptBead* find_script_bead(const char* name);
diguyCharacterPathScriptBead* create_script_bead(const char* script,
    int index = -1,
    const char* script_type = "lua");
int destroy_script_bead(diguyCharacterPathScriptBead* script_bead);
int get_num_decision_beads();
diguyCharacterPathDecisionBead* get_decision_bead_at_index(int index);
diguyCharacterPathDecisionBead* find_decision_bead(const char* name);
int get_num_gaze_beads();
diguyCharacterPathGazeBead* get_gaze_bead_at_index(int index);
diguyCharacterPathGazeBead* find_gaze_bead(const char* name);
int get_num_aim_beads();
diguyCharacterPathAimBead* get_aim_bead_at_index(int index);
diguyCharacterPathAimBead* find_aim_bead(const char* name);
#ifdef CPLUSPLUS_ONLY
enum {
    CALLBACK_ID_TIME_LEFT_DEPRECATED = 1,
    CALLBACK_ID_TIME_INT0_DEPRECATED
};
int add_callback(int callback_id,
    diguyCharacterPathCallback* callback,
    void* callback_params = 0,
    void* callback_user_data = 0);
int remove_callback(int callback_id,
    diguyCharacterPathCallback* callback);
int remove_callback_with_user_data(int callback_id,
    void* callback_user_data);
int translate(float tx, float ty, float tz);
int rotate_about_point(float rz, float rx, float ry,
    float rotation_pt_x, float rotation_pt_y, float rotation_pt_z);
int set_ground_clamp_when_on_path(int ground_clamp_when_on_path);
int get_ground_clamp_when_on_path();
int get_position_at_distance(float distance,
    float* tx, float* ty, float* tz,
    float* rz = NULL, float* rx = NULL, float* ry = NULL);
int get_position_at_distance_double(float distance,
    double* tx, double* ty, double* tz,
    float* rz = NULL, float* rx = NULL, float* ry = NULL);
int get_position_at_t(float t,
    float* tx, float* ty, float* tz,
    float* rz = NULL, float* rx = NULL, float* ry = NULL);
int get_position_at_t_double(float t,
    double* tx, double* ty, double* tz,
    float* rz = NULL, float* rx = NULL, float* ry = NULL);

```


5.18.2 function diguyCharacterPath::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the path. This pointer will never be NULL.

Returns:

name of the path

Callable From:

- ◇ C++
- ◇ Script

5.18.3 function diguyCharacterPath::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.18.4 function diguyCharacterPath::get_uid

Prototype:

```
long get_uid();
```

Description:

All character paths are assigned a unique identifier, or uid. This function returns this path's uid.

Note: unique identifiers will change between DI-Guy runs!

Returns:

unique identifier of object

Callable From:

- ◇ C++
- ◇ Script

5.18.5 function diguyCharacterPath::get_character

Prototype:

```
diguyCharacter* get_character();
```

Returns:

character that owns this path

Callable From:

- ◇ C++
- ◇ Script

5.18.6 function `diguyCharacterPath::update`

Prototype:

```
int update();
```

Description:

This function should be called after all modifications have been made to the path and it is time to re-derive the overall path shape and when actions happen.

This update is deferred for reasons of performance; if there are a lot of modifications to be made to the path they can all be done in a group with only a single final update.

A call to `update( )` is typically required after certain calls are made on objects owned by the path, such as a `diguyWaypoint` or a `diguyCharacterPathActionBead`. The documentation for function calls of these objects will state whether a subsequent `update( )` call is necessary.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.18.7 function diguyCharacterPath::get_length

Prototype:

```
float get_length();
```

Description:

This function returns the total length of the path, in meters.

Returns:

length of path in meters

Callable From:

- ◇ C++
- ◇ Script

5.18.8 function diguyCharacterPath::get_duration

Prototype:

```
float get_duration();
```

Description:

This function returns how long this path is

Returns:

duration of this path, in seconds

Callable From:

- ◇ C++
- ◇ Script

5.18.9 function diguyCharacterPath::get_point_at_distance**Prototype:**

```
int get_point_at_distance(float distance_into_path,
    float* x, float* y, float* z,
    float* rz, float* rx, float* ry);
```

Description:

This function returns, via the passed variable pointers, the position and orientation of the point on the path at the specified distance.

Arguments:

<code>distance_into_path</code>	distance in meters into path of the desired point
<code>x, y, z</code>	position of point in meters from the origin
<code>rz, rx, ry</code>	orientation of point in degrees counter-clockwise from the positive X axis

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.18.10 function diguyCharacterPath::get_path_shape

Prototype:

```
diguyPathShape* get_path_shape();
```

Description:

This function returns the underlying path shape of the path. The path shape defines only the waypoints and associated spline shape of the path; it doesn't not contain any beads or other information.

The returned path shape should be considered READ ONLY. DO NOT edit the path shape using [diguyPathShape](#) member functions!

Returns:

pointer to object of type [diguyPathShape](#)

Callable From:

- ◇ C++
- ◇ Script

Waypoint Functions

5.18.11 function diguyCharacterPath::get_num_waypoints**Prototype:**

```
int get_num_waypoints();
```

Returns:

the number of waypoints belonging to this path

Callable From:

- ◇ C++
- ◇ Script

5.18.12 function diguyCharacterPath::get_waypoint_at_index**Prototype:**

```
diguyWaypoint* get_waypoint_at_index(int index);
```

Description:

This function gets the waypoint at the specified index and returns a pointer to it. The returned waypoint can then be queried for information or modified.

Note that if the waypoint is modified, the [update\(\)](#) function must be called to update the path's shape and other information.

Arguments:

index	index of the waypoint; indices start at 0

Returns:

pointer of type [diguyWaypoint](#); NULL if no waypoint at the specified index

Callable From:

- ◇ C++
- ◇ Script

Lua Example:

This example gets the first waypoint of the first character's first path. It then changes the x coordinate of the waypoint and updates the path to reflect the change.

```
local character = this_scenario:get_character_at_index(0);
local path0 = character:get_path_at_index(0);

local wp0 = path0:get_waypoint_at_index(0);
local wp0_x = wp0:get_x();
wp0:set_x(wp0_x + 1.0);

path0:update();
```

5.18.13 function diguyCharacterPath::find_waypoint

Prototype:

```
diguyWaypoint* find_waypoint(const char* name);
```

Description:

This function returns a pointer to the specified waypoint.

Arguments:

name	name of waypoint to be found

Returns:

pointer of type [diguyWaypoint](#); NULL if not found

Callable From:

- ◇ C++
- ◇ Script

5.18.14 function `diguyCharacterPath::create_waypoint`

Prototype:

```
diguyWaypoint* create_waypoint(float x = 0.0f, float y = 0.0f, float z = 0.0f,
    float yaw = 0.0f, float roll = 0.0f, float pitch = 0.0f,
    float weight = 1.0f,
    int index = -1);
```

Description:

This function creates a new waypoint that is added to the path. Do not call `diguyScenario::destroy_waypoint( )` on the returned waypoint; it will be destroyed automatically with the path.

Arguments:

<code>tx, ty, tz</code>	position in meters from the origin
<code>rz, rx, ry</code>	orientations in degrees counter-clockwise from the positive X axis
<code>weight</code>	how much influence this waypoint exerts over the path curve
<code>index</code>	where the waypoint should be inserted in the path; indices start at 0; pass -1 to add to end of path

Returns:

pointer to type `diguyWaypoint`

Callable From:

- ◇ C++
- ◇ Script

5.18.15 function diguyCharacterPath::add_waypoint

Prototype:

```
int add_waypoint(diguyWaypoint* waypoint);
```

Description:

This function adds a waypoint to the path.

The waypoint should be created using the function `diguyScenario::create_waypoint()`, and destroyed using the function `diguyScenario::destroy_waypoint()`.

An internal copy of the waypoint is made, so it can be destroyed right after this function call or used for other path shapes.

Arguments:

`waypoint` waypoint to be appended to end of path

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Action Bead Functions

5.18.16 function diguyCharacterPath::get_num_action_beads

Prototype:

```
int get_num_action_beads();
```

Returns:

the number of action beads belonging to this path

Callable From:

- ◇ C++
- ◇ Script

5.18.17 function diguyCharacterPath::get_action_bead_at_index**Prototype:**

```
diguyCharacterPathActionBead* get_action_bead_at_index(int index);
```

Description:

This function gets the action bead at the specified index and returns a pointer to it. The returned action bead can then be queried for information or modified.

Note that if the action bead is modified, the [update\(\)](#) function must be called to update the path's information.

Returns:

pointer of type [diguyCharacterPathActionBead](#); NULL if no action bead at the specified index

Arguments:

index	index of the action bead; indices start at 0

Callable From:

- ◇ C++
- ◇ Script

5.18.18 function diguyCharacterPath::find_action_bead**Prototype:**

```
diguyCharacterPathActionBead* find_action_bead(const char* name);
```

Description:

This function returns a pointer to the specified bead.

Arguments:

name	name of bead to be found

Returns:

pointer of type [diguyCharacterPathActionBead](#); NULL if not found

Callable From:

- ◇ C++
- ◇ Script

5.18.19 function diguyCharacterPath::get_final_action_bead**Prototype:**

```
diguyCharacterPathActionBead* get_final_action_bead();
```

Description:

This function returns a pointer to the last action bead of the path. The last action bead is slightly different than the rest. It has no inherent duration or length; instead, it effectively specifies what the character should do once this path has completed.

Calling this function is the same as the following:

```
int num_action_beads = path->get_num_action_beads();
diguyCharacterPathActionBead* ab;
ab = path->get_action_bead_at_index(num_action_beads-1);
```

Do not call delete or [destroy_action_bead\(\)](#) on the returned pointer. All paths must have a final action bead.

Returns:

pointer of type [diguyCharacterPathActionBead](#); this pointer should never be NULL

Callable From:

- ◇ C++
- ◇ Script

5.18.20 function [diguyCharacterPath::create_action_bead](#)

Prototype:

```
diguyCharacterPathActionBead* create_action_bead(const char* action,
int index = -1);
```

Description:

This function inserts a new action bead at the specified index and returns a pointer to it. The returned action bead can then be queried for information or modified.

Note that the [update\(\)](#) function must be called after this function.

Do not call delete on the returned pointer. Use [destroy_action_bead\(\)](#) instead.

Returns:

pointer of type [diguyCharacterPathActionBead](#); NULL if insertion failed

Arguments:

<code>action</code>	action new bead should perform
<code>index</code>	index of the action bead; indices start at 0; pass -1 to append the action bead to the end of the path

Callable From:

- ◇ C++
- ◇ Script

5.18.21 function `diguyCharacterPath::destroy_action_bead`

Prototype:

```
int destroy_action_bead(diguyCharacterPathActionBead* action_bead);
```

Description:

This function removes the passed action bead from the path and destroys it.

Note that the `update( )` function must be called after this function.

Returns:

0 on success, -1 on failure

Arguments:

`action_bead` object to destroy; must be part of this path

Callable From:

- ◇ C++
- ◇ Script

Script Bead Functions

5.18.22 function diguyCharacterPath::get_num_script_beads**Prototype:**

```
int get_num_script_beads();
```

Returns:

the number of script beads belonging to this path

Callable From:

- ◇ C++
- ◇ Script

5.18.23 function diguyCharacterPath::get_script_bead_at_index**Prototype:**

```
diguyCharacterPathScriptBead* get_script_bead_at_index(int index);
```

Description:

This function gets the script bead at the specified index and returns a pointer to it. The returned script bead can then be queried for information or modified.

Note that if the script bead is modified, the [update\(\)](#) function must be called to update the path's information.

Returns:

pointer of type [diguyCharacterPathScriptBead](#); NULL if no script bead at the specified index

Arguments:

index index of the script bead; indices start at 0

Callable From:

- ◇ C++
- ◇ Script

5.18.24 function diguyCharacterPath::find_script_bead

Prototype:

```
diguyCharacterPathScriptBead* find_script_bead(const char* name);
```

Description:

This function returns a pointer to the specified bead.

Arguments:

name name of bead to be found

Returns:

pointer of type [diguyCharacterPathScriptBead](#); NULL if not found

Callable From:

- ◇ C++
- ◇ Script

5.18.25 function diguyCharacterPath::create_script_bead

Prototype:

```
diguyCharacterPathScriptBead* create_script_bead(const char* script,
        int index = -1,
        const char* script_type = "lua");
```

Description:

This function inserts a new script bead at the specified index and returns a pointer to it. The returned script bead can then be queried for information or modified.

Note that the [update\(\)](#) function must be called after this function.

Do not call delete on the returned pointer. Use [destroy_script_bead\(\)](#) instead.

Returns:

pointer of type [diguyCharacterPathScriptBead](#); NULL if insertion failed

Arguments:

<code>script</code>	script new bead should perform
<code>index</code>	index of the script bead; indices start at 0; pass -1 to create the bead at the last index
<code>script_type</code>	defaults to "lua"

Callable From:

- ◇ C++
- ◇ Script

5.18.26 function diguyCharacterPath::destroy_script_bead

Prototype:

```
int destroy_script_bead(diguyCharacterPathScriptBead* script_bead);
```

Description:

This function removes the passed script bead from the path and destroys it.

Note that the [update\(\)](#) function must be called after this function.

Returns:

0 on success, -1 on failure

Arguments:

`script_bead` object to destroy; must be part of this path

Callable From:

- ◇ C++
- ◇ Script

Decision Bead Functions

5.18.27 function diguyCharacterPath::get_num_decision_beads**Prototype:**

```
int get_num_decision_beads();
```

Returns:

the number of decision beads belonging to this path

Callable From:

- ◇ C++
- ◇ Script

5.18.28 function diguyCharacterPath::get_decision_bead_at_index**Prototype:**

```
diguyCharacterPathDecisionBead* get_decision_bead_at_index(int index);
```

Description:

This function gets the decision bead at the specified index and returns a pointer to it. The returned decision bead can then be queried for information.

Note that if the decision bead is modified, the [update\(\)](#) function must be called to update the path's information.

Returns:

pointer of type [diguyCharacterPathDecisionBead](#); NULL if no script bead at the specified index

Arguments:

index index of the decision bead; indices start at 0

Callable From:

- ◇ C++
- ◇ Script

5.18.29 function diguyCharacterPath::find_decision_bead

Prototype:

```
diguyCharacterPathDecisionBead* find_decision_bead(const char* name);
```

Description:

This function returns a pointer to the specified bead.

Arguments:

name name of bead to be found

Returns:

pointer of type [diguyCharacterPathDecisionBead](#); NULL if not found

Callable From:

- ◇ C++

◇ Script

Gaze Bead Functions

5.18.30 function diguyCharacterPath::get_num_gaze_beads**Prototype:**

```
int get_num_gaze_beads();
```

Returns:

the number of gaze beads belonging to this path

Callable From:

- ◇ C++
- ◇ Script

5.18.31 function diguyCharacterPath::get_gaze_bead_at_index**Prototype:**

```
diguyCharacterPathGazeBead* get_gaze_bead_at_index(int index);
```

Description:

This function gets the gaze bead at the specified index and returns a pointer to it. The returned gaze bead can then be queried for information.

Note that if the gaze bead is modified, the [update\(\)](#) function must be called to update the path's information.

Returns:

pointer of type [diguyCharacterPathDecisionBead](#); NULL if no script bead at the specified index

Arguments:

index	index of the gaze bead; indices start at 0

Callable From:

- ◇ C++
- ◇ Script

5.18.32 function diguyCharacterPath::find_gaze_bead

Prototype:

```
diguyCharacterPathGazeBead* find_gaze_bead(const char* name);
```

Description:

This function returns a pointer to the specified bead.

Arguments:

name	name of bead to be found

Returns:

pointer of type [diguyCharacterPathGazeBead](#); NULL if not found

Callable From:

- ◇ C++
- ◇ Script

Aim Bead Functions

5.18.33 function diguyCharacterPath::get_num_aim_beads

Prototype:

```
int get_num_aim_beads();
```

Returns:

the number of aim beads belonging to this path

Callable From:

- ◇ C++
- ◇ Script

5.18.34 function diguyCharacterPath::get_aim_bead_at_index**Prototype:**

```
diguyCharacterPathAimBead* get_aim_bead_at_index(int index);
```

Description:

This function gets the aim bead at the specified index and returns a pointer to it. The returned aim bead can then be queried for information.

Note that if the aim bead is modified, the [update\(\)](#) function must be called to update the path's information.

Returns:

pointer of type [diguyCharacterPathDecisionBead](#); NULL if no script bead at the specified index

Arguments:

<code>index</code>	index of the aim bead; indices start at 0

Callable From:

- ◇ C++
- ◇ Script

5.18.35 `function diguyCharacterPath::find_aim_bead`**Prototype:**

```
diguyCharacterPathAimBead* find_aim_bead(const char* name);
```

Description:

This function returns a pointer to the specified bead.

Arguments:

name	name of bead to be found

Returns:

pointer of type `diguyCharacterPathAimBead`; NULL if not found

Callable From:

- ◇ C++
- ◇ Script

Callback Functions

5.18.36 `enumeration`

```
enum {
    CALLBACK_ID_TIME_LEFT_DEPRECATED = 1,
    CALLBACK_ID_TIME_INTO_DEPRECATED
};
```

Description:

This is an enumeration of the different callbacks that can be registered with `add_callback()`.

Usable From:

- ◇ C++
- ◇ Script

5.18.37 function diguyCharacterPath::add_callback

Prototype:

```
int add_callback(int callback_id,
                 diguyCharacterPathCallback* callback,
                 void* callback_params = 0,
                 void* callback_user_data = 0);
```

Description:

This function adds a user callback.

Arguments:

<code>callback</code>	pointer to function with prototype <code>diguyCharacterPathCallback</code> (typedefed above)
<code>callback_id</code>	integer id of when this callback is to be called
<code>callback_params</code>	not currently used; pass NULL
<code>callback_user_data</code>	pointer for user's own use; DI-Guy will do nothing to the contents of this pointer beyond passing it back when the callback is invoked

`callback_id` should be one of the following values:

CALLBACK_ID_TIME_LEFT_DEPRECATED

This callback id was formerly `CALLBACK_ID_TIME_LEFT`, which has been deprecated. Use a decision bead or script bead located near the end of the path for new development.

CALLBACK_ID_TIME_INTO_DEPRECATED

This callback id was formerly `CALLBACK_ID_TIME_INTO`, which has been deprecated. Use a decision bead or script bead located near the beginning of the path for new development.

Callbacks return a value of type `diguyCallbackReturn`, which will be `DIGUY_CALLBACK_STOP` or `DIGUY_CALLBACK_CONTINUE`. If the callback returns `DIGUY_CALLBACK_STOP`, the default handler of the function will not be called; the callback is asserting that it has done everything necessary for the function call. If the callback returns `DIGUY_CALLBACK_CONTINUE`, the default handler for the function will be called after the callback.

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.18.38 function diguyCharacterPath::remove_callback**Prototype:**

```
int remove_callback(int callback_id,  
                    diguyCharacterPathCallback* callback);
```

Description:

This function removes a user callback. All callbacks matching the specified `callback_id` and callback function will be removed.

Arguments:

<code>callback_id</code>	integer id of when this callback is to be called
<code>callback</code>	pointer to function with prototype <code>diguyCharacterCallback</code> (typedefed above)

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.18.39 function diguyCharacterPath::remove_callback_with_user_data

Prototype:

```
int remove_callback_with_user_data(int callback_id,
    void* callback_user_data);
```

Description:

This function removes a user callback. All callbacks matching the specified `callback_id` and `callback_user_data` pointer will be removed.

Arguments:

<code>callback_id</code>	integer id of when this callback is to be called
<code>callback_user_data</code>	pointer for user's own use

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

Experimental Functions

The following functions are experimental and may change or disappear with no warning.

5.18.40 function diguyCharacterPath::translate**Prototype:**

```
int translate(float tx, float ty, float tz);
```

5.18.41 function diguyCharacterPath::rotate_about_point**Prototype:**

```
int rotate_about_point(float rz, float rx, float ry,  
    float rotation_pt_x, float rotation_pt_y, float rotation_pt_z);
```

5.18.42 function

diguyCharacterPath::set_ground_clamp_when_on_path

Prototype:

```
int set_ground_clamp_when_on_path(int ground_clamp_when_on_path);
```

5.18.43 function

diguyCharacterPath::get_ground_clamp_when_on_path

Prototype:

```
int get_ground_clamp_when_on_path();
```

5.18.44 function diguyCharacterPath::get_position_at_distance**Prototype:**

```
int get_position_at_distance(float distance,  
    float* tx, float* ty, float* tz,  
    float* rz = NULL, float* rx = NULL, float* ry = NULL);
```

**5.18.45 function
diguyCharacterPath::get_position_at_distance_double****Prototype:**

```
int get_position_at_distance_double(float distance,  
    double* tx, double* ty, double* tz,  
    float* rz = NULL, float* rx = NULL, float* ry = NULL);
```

5.18.46 function diguyCharacterPath::get_position_at_t**Prototype:**

```
int get_position_at_t(float t,  
    float* tx, float* ty, float* tz,  
    float* rz = NULL, float* rx = NULL, float* ry = NULL);
```

5.18.47 function diguyCharacterPath::get_position_at_t_double**Prototype:**

```
int get_position_at_t_double(float t,  
    double* tx, double* ty, double* tz,  
    float* rz = NULL, float* rx = NULL, float* ry = NULL);
```

5.19 diguyCharacterPathActionBead

diguyCharacterPathActionBead.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

Link against: libdiguy

5.19.1 class diguyCharacterPathActionBead

```
class BDI_DECLSPEC_diguy diguyCharacterPathActionBead
{
public:
    const char* get_name();
    int set_name(const char* name);
    const char* get_action();
    int set_action(const char* action);
    float get_distance_into_path();
    int set_distance_into_path(float distance_into_path);
    float get_length();
    int set_length(float length);
    int get_derive_length_flag();
    int set_derive_length_flag(int derive_length_flag);
    float get_tin();
    int set_tin(float tin);
    float get_duration();
    int set_duration(float duration);
    int get_derive_duration_flag();
    int set_derive_duration_flag(int derive_duration_flag);
    float get_desired_duration();
    int set_desired_duration(float desired_duration);
    int get_derive_desired_duration_flag();
    int set_derive_desired_duration_flag(int derive_desired_duration_flag);
    int get_motion_reps();
    int set_motion_reps(int motion_reps);
    int get_derive_motion_reps_flag();
    int set_derive_motion_reps_flag(int derive_motion_reps);
    int get_transition_before_bead_flag();
    int set_transition_before_bead_flag(int transition_before_bead_flag);
    const char* get_companion_waypoint_name();
    int set_companion_waypoint_name(const char* companion_waypoint_name);
    int create_or_update_companion_waypoint();
```

5.19.2 function diguyCharacterPathActionBead::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the object

Callable From:

- ◇ C++
- ◇ Script

5.19.3 function diguyCharacterPathActionBead::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.19.4 function `diguyCharacterPathActionBead::get_action`

Prototype:

```
const char* get_action();
```

Description:

This function returns the action this bead specifies. This pointer will never be NULL.

Returns:

action of this bead

Callable From:

- ◇ C++
- ◇ Script

5.19.5 function `diguyCharacterPathActionBead::set_action`

Prototype:

```
int set_action(const char* action);
```

Description:

This function sets the action this bead specifies. When a character is traversing the path containing this bead, the specified action will be performed when this bead is reached.

If this action bead is part of a [diguyCharacterPath](#), the [diguyCharacterPath::update\(\)](#) function must be called for changes to take effect.

Arguments:

<code>action</code>	the action the character should perform when this bead is reached

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.19.6 function `diguyCharacterPathActionBead::get_distance_into_path`

Prototype:

```
float get_distance_into_path();
```

Description:

This function returns how far into the path this bead is.

Returns:

distance into path of this bead, in meters

Callable From:

- ◇ C++
- ◇ Script

5.19.7 function `diguyCharacterPathActionBead::set_distance_into_path`

Prototype:

```
int set_distance_into_path(float distance_into_path);
```

Description:

This function sets how far into the path this bead is.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting when on the path the bead is located, as if to a call to `set_tin()`.

If this script bead is part of a `diguyCharacterPath`, the `diguyCharacterPath::update()` function must be called for changes to take effect.

Arguments:

`distance_into_path` how far into the path this bead is, in meters

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.19.8 function diguyCharacterPathActionBead::get_length

Prototype:

```
float get_length();
```

Description:

This function returns where on the path this bead is.

Returns:

location of this bead, in meters

Callable From:

- ◇ C++
- ◇ Script

5.19.9 function diguyCharacterPathActionBead::set_length

Prototype:

```
int set_length(float length);
```

Description:

This function sets how long this bead will have an effect. The bead will have an effect from its initial distance into the path (as set by [set_distance_into_path\(\)](#)) to (distance + length) seconds.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting when on the path the bead's effect ends, as if to a call to [set_duration\(\)](#).

If this script bead is part of a [diguyCharacterPath](#), the [diguyCharacterPath::update\(\)](#) function must be called for changes to take effect.

Arguments:

`duration` duration of this bead, in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.19.10 function `diguyCharacterPathActionBead::get_derive_length_flag`

Prototype:

```
int get_derive_length_flag();
```

Description:

This function returns whether the length of this bead is derived from other information in the bead. If so, calling `set_length( )` will have no effect.

See `set_derive_length_flag( )` for information on how length is derived.

Returns:

integer flag; 1 means length is derived, 0 means it is not

Callable From:

- ◇ C++

◇ Script

5.19.11 function diguyCharacterPathActionBead::set_derive_length_flag

Prototype:

```
int set_derive_length_flag(int derive_length_flag);
```

Description:

This function sets whether the length of this bead is derived from other information in the bead. If so, calling [set_length\(\)](#) will have no effect.

(More documentation needed to explain how length is derived.)

Arguments:

`derive_length_flag` pass 1 for length to be derived, else 0

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.19.12 function diguyCharacterPathActionBead::get_tin

Prototype:

```
float get_tin();
```

Description:

This function returns when on the path this bead is.

Returns:

time location of this bead, in seconds

Callable From:

- ◇ C++
- ◇ Script

5.19.13 function diguyCharacterPathActionBead::set_tin**Prototype:**

```
int set_tin(float tin);
```

Description:

This function sets when this bead will begin to have an effect.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting where on the path the bead is located, as if to a call to [set_distance_into_path\(\)](#).

If this script bead is part of a [diguyCharacterPath](#), the [diguyCharacterPath::update\(\)](#) function must be called for changes to take effect.

Arguments:

<code>tin</code>	when this bead begins to have an effect, in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.19.14 function diguyCharacterPathActionBead::get_duration**Prototype:**

```
float get_duration();
```

Description:

This function returns how long this bead will have an effect.

Returns:

duration of this bead, in seconds

Callable From:

- ◇ C++
- ◇ Script

5.19.15 function diguyCharacterPathActionBead::set_duration**Prototype:**

```
int set_duration(float duration);
```

Description:

This function sets how long this bead will have an effect. The bead will have an effect from tin seconds (as set by `set_tin( )`) to (tin + duration) seconds.

Note that if duration is derived (as by a call to `set_derive_duration_flag( )`), this function will have no effect.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting where on the path the bead's effect ends, as if to a call to `set_length( )`.

If this script bead is part of a `diguyCharacterPath`, the `diguyCharacterPath::update( )` function must be called for changes to take effect.

Arguments:

`duration` duration of this bead, in seconds

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.19.16 function `diguyCharacterPathActionBead::get_derive_duration_flag`

Prototype:

```
int get_derive_duration_flag();
```

Description:

This function returns whether the duration of this bead is derived from other information in the bead. If so, calling `set_duration( )` will have no effect.

See `set_derive_duration_flag( )` for information on how length is derived.

Returns:

integer flag; 1 means duration is derived, 0 means it is not

Callable From:

◇ C++
◇ Script

5.19.17 function `diguyCharacterPathActionBead::set_derive_duration_flag`

Prototype:

```
int set_derive_duration_flag(int derive_duration_flag);
```

Description:

This function sets whether the duration of this bead is derived from other information in the bead. If so, calling `set_duration( )` will have no effect.

(More documentation needed to explain how duration is derived.)

Arguments:

`derive_length_flag` pass 1 for duration to be derived, else 0

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.19.18 function `diguyCharacterPathActionBead::get_desired_duration`

Prototype:

```
float get_desired_duration();
```

Returns:

the desired duration of this bead

Callable From:

- ◇ C++
- ◇ Script

5.19.19 function diguyCharacterPathActionBead::set_desired_duration

Prototype:

```
int set_desired_duration(float desired_duration);
```

Description:

This function sets the desired duration of the action bead.

Arguments:

`desired_duration` desired duration of action in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.19.20 function diguyCharacterPathActionBead::get_derive_desired_duration_flag

Prototype:

```
int get_derive_desired_duration_flag();
```

Description:

This function returns whether the desired duration of this bead is derived from other information in the bead. If so, calling [set_desired_duration\(\)](#) will have no effect.

Returns:

integer flag; 1 means duration is derived, 0 means it is not

Callable From:

- ◇ C++
- ◇ Script

5.19.21 function diguyCharacterPathActionBead::set_derive_desired_duration_flag

Prototype:

```
int set_derive_desired_duration_flag(int derive_desired_duration_flag);
```

Description:

This function sets whether the desired duration of this bead is derived from other information in the bead. If so, calling `set_derived_duration()` will have no effect.

Arguments:

`derive_desired_duration_flag` pass 1 for duration to be derived,
else 0

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.19.22 function diguyCharacterPathActionBead::get_motion_reps

Prototype:

```
int get_motion_reps();
```

Returns:

how many repetitions of the action's looping motion will be played

Callable From:

- ◇ C++
- ◇ Script

5.19.23 function diguyCharacterPathActionBead::set_motion_reps

Prototype:

```
int set_motion_reps(int motion_reps);
```

Description:

This function sets how many repetitions of the action's looping motion will be played.

Arguments:

`motion_reps` how many times to play the looping motion

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.19.24 function

diguyCharacterPathActionBead::get_derive_motion_reps_flag

Prototype:

```
int get_derive_motion_reps_flag();
```

Description:

This function returns whether the number of motion repetitions for this action is derived from other data in the bead. If so, calling [set_motion_reps\(\)](#) will have no effect.

Returns:

integer flag; 1 means number of motion reps is derived, 0 means it is not

Callable From:

◇ C++
◇ Script

5.19.25 function

diguyCharacterPathActionBead::set_derive_motion_reps_flag

Prototype:

```
int set_derive_motion_reps_flag(int derive_motion_reps);
```

Description:

This function sets whether the number of motion repetitions for this bead is derived from other information in the bead. If so, calling `set_motion_reps( )` will have no effect.

Arguments:

`derive_motion_reps` pass 1 for number of motion reps to be derived,
else 0

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.19.26 function diguyCharacterPathActionBead::get_transition_before_bead_flag

Prototype:

```
int get_transition_before_bead_flag();
```

Description:

This function returns whether the transition motion for this action bead from the previous action bead (if any) occurs before this action bead or after.

Returns:

integer flag; 1 means transition happens before, 0 means it happens after

Callable From:

- ◇ C++
- ◇ Script

5.19.27 function diguyCharacterPathActionBead::set_transition_before_bead_flag

Prototype:

```
int set_transition_before_bead_flag(int transition_before_bead_flag);
```

Description:

This function sets whether the transition motion for this action bead from the previous action bead (if any) occurs before this action bead or after.

Arguments:

transition_before_bead_flag pass 1 for transition to happen before, 0 for it to happen after

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.19.28 function diguyCharacterPathActionBead::get_companion_waypoint_name

Prototype:

```
const char* get_companion_waypoint_name();
```

Description:

This function returns the name of this bead's companion waypoint.

Returns:

C string identifying name of companion waypoint; NULL if there is no companion waypoint

Callable From:

◇ C++
◇ Script

5.19.29 function diguyCharacterPathActionBead::set_companion_waypoint_name

Prototype:

```
int set_companion_waypoint_name(const char* companion_waypoint_name);
```

Description:

This function sets the name of this action bead's companion waypoint. The action bead will be constrained to stay at the companion waypoint as the waypoint is moved around.

Arguments:

`companion_waypoint_name` name of companion waypoint; pass NULL to un-set the companion waypoint

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.19.30 function diguyCharacterPathActionBead::create_or_update_companion_w

Prototype:

```
int create_or_update_companion_waypoint();
```

Description:

This function creates a companion waypoint for this action bead if it doesn't already have one. The created waypoint will be at the action bead's current position. If this action bead already has a companion waypoint, it may be modified to show facing direction changes if this action or the previous action have changed since the companion waypoint was assigned.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.20 diguyCharacterPathAimBead

diguyCharacterPathAimBead.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

◆ [Experimental Functions](#)

Link against: libdiguy

5.20.1 class diguyCharacterPathAimBead

```
class BDI_DECLSPEC_diguy diguyCharacterPathAimBead
{
public:
    const char* get_name();
    int set_name(const char* name);
    float get_distance_into_path();
    int set_distance_into_path(float distance_into_path);
    float get_length();
    int set_length(float length);
    float get_tin();
    int set_tin(float tin);
    float get_duration();
    int set_duration(float duration);
    const char* get_begin_at_object_name();
    int set_begin_at_object_name(const char* begin_at_object_name);
    float get_begin_at_object_offset();
    int set_begin_at_object_offset(float begin_at_object_offset);
    const char* get_end_at_object_name();
    int set_end_at_object_name(const char* end_at_object_name);
    float get_end_at_object_offset();
    int set_end_at_object_offset(float end_at_object_offset);
};
```

5.20.2 function diguyCharacterPathAimBead::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the object

Callable From:

- ◇ C++
- ◇ Script

5.20.3 function diguyCharacterPathAimBead::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.20.4 function `diguyCharacterPathAimBead::get_distance_into_path`

Prototype:

```
float get_distance_into_path();
```

Description:

This function returns how far into the path this bead is.

Returns:

distance into path of this bead, in meters

Callable From:

- ◇ C++
- ◇ Script

5.20.5 function `diguyCharacterPathAimBead::set_distance_into_path`

Prototype:

```
int set_distance_into_path(float distance_into_path);
```

Description:

This function sets how far into the path this bead is.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting when on the path the bead is located, as if to a call to `set_tin()`.

If this script bead is part of a `diguyCharacterPath`, the `diguyCharacterPath::update()` function must be called for changes to take effect.

Arguments:

`distance_into_path` how far into the path this bead is, in meters

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.20.6 function diguyCharacterPathAimBead::get_length

Prototype:

```
float get_length();
```

Description:

This function returns where on the path this bead is.

Returns:

location of this bead, in meters

Callable From:

- ◇ C++
- ◇ Script

5.20.7 function diguyCharacterPathAimBead::set_length

Prototype:

```
int set_length(float length);
```

Description:

This function sets how long this bead will have an effect. The bead will have an effect from its initial distance into the path (as set by [set_distance_into_path\(\)](#)) to (distance + length) seconds.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting when on the path the bead's effect ends, as if to a call to [set_duration\(\)](#).

If this script bead is part of a [diguyCharacterPath](#), the [diguyCharacterPath::update\(\)](#) function must be called for changes to take effect.

Arguments:

`duration` duration of this bead, in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.20.8 function diguyCharacterPathAimBead::get_tin

Prototype:

```
float get_tin();
```

Description:

This function returns when on the path this bead is.

Returns:

time location of this bead, in seconds

Callable From:

- ◇ C++
- ◇ Script

5.20.9 function diguyCharacterPathAimBead::set_tin

Prototype:

```
int set_tin(float tin);
```

Description:

This function sets when this bead will begin to have an effect.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting where on the path the bead is located, as if to a call to [set_distance_into_path\(\)](#).

If this script bead is part of a [diguyCharacterPath](#), the [diguyCharacterPath::update\(\)](#) function must be called for changes to take effect.

Arguments:

`tin` when this bead begins to have an effect, in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.20.10 function diguyCharacterPathAimBead::get_duration

Prototype:

```
float get_duration();
```

Description:

This function returns how long this bead will have an effect.

Returns:

duration of this bead, in seconds

Callable From:

- ◇ C++
- ◇ Script

5.20.11 function diguyCharacterPathAimBead::set_duration

Prototype:

```
int set_duration(float duration);
```

Description:

This function sets how long this bead will have an effect. The bead will have an effect from tin seconds (as set by `set_tin( )`) to (tin + duration) seconds.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting where on the path the bead's effect ends, as if to a call to `set_length( )`.

If this script bead is part of a `diguyCharacterPath`, the `diguyCharacterPath::update( )` function must be called for changes to take effect.

Arguments:

`duration` duration of this bead, in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Experimental Functions

The following functions are experimental and may change or disappear with no warning.

5.20.12 function

diguyCharacterPathAimBead::get_begin_at_object_name

Prototype:

```
const char* get_begin_at_object_name();
```

5.20.13 function

diguyCharacterPathAimBead::set_begin_at_object_name

Prototype:

```
int set_begin_at_object_name(const char* begin_at_object_name);
```

5.20.14 function

diguyCharacterPathAimBead::get_begin_at_object_offset

Prototype:

```
float get_begin_at_object_offset();
```

5.20.15 function**diguyCharacterPathAimBead::set_begin_at_object_offset****Prototype:**

```
int set_begin_at_object_offset(float begin_at_object_offset);
```

5.20.16 function**diguyCharacterPathAimBead::get_end_at_object_name****Prototype:**

```
const char* get_end_at_object_name();
```

5.20.17 function

diguyCharacterPathAimBead::set_end_at_object_name

Prototype:

```
int set_end_at_object_name(const char* end_at_object_name);
```

5.20.18 function

diguyCharacterPathAimBead::get_end_at_object_offset

Prototype:

```
float get_end_at_object_offset();
```

5.20.19 function

diguyCharacterPathAimBead::set_end_at_object_offset

Prototype:

```
int set_end_at_object_offset(float end_at_object_offset);
```


5.21 diguyCharacterPathDecisionBead

diguyCharacterPathDecisionBead.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

◆ [Experimental Functions](#)

Link against: libdiguy

5.21.1 class diguyCharacterPathDecisionBead

```
class BDI_DECLSPEC_diguy diguyCharacterPathDecisionBead
{
public:
    const char* get_name();
    int set_name(const char* name);
    float get_distance_into_path();
    int set_distance_into_path(float distance_into_path);
    float get_length();
    int set_length(float length);
    float get_tin();
    int set_tin(float tin);
    float get_duration();
    int set_duration(float duration);
    const char* get_begin_at_object_name();
    int set_begin_at_object_name(const char* begin_at_object_name);
    float get_begin_at_object_offset();
    int set_begin_at_object_offset(float begin_at_object_offset);
    const char* get_end_at_object_name();
    int set_end_at_object_name(const char* end_at_object_name);
    float get_end_at_object_offset();
    int set_end_at_object_offset(float end_at_object_offset);
};
```

5.21.2 function diguyCharacterPathDecisionBead::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the object

Callable From:

- ◇ C++
- ◇ Script

5.21.3 function diguyCharacterPathDecisionBead::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.21.4 function**diguyCharacterPathDecisionBead::get_distance_into_path****Prototype:**

```
float get_distance_into_path();
```

Description:

This function returns how far into the path this bead is.

Returns:

distance into path of this bead, in meters

Callable From:

- ◇ C++
- ◇ Script

5.21.5 function**diguyCharacterPathDecisionBead::set_distance_into_path****Prototype:**

```
int set_distance_into_path(float distance_into_path);
```

Description:

This function sets how far into the path this bead is.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting when on the path the bead is located, as if to a call to [set_tin\(\)](#).

If this script bead is part of a [diguyCharacterPath](#), the [diguyCharacterPath::update\(\)](#) function must be called for changes to take effect.

Arguments:

`distance_into_path` how far into the path this bead is, in meters

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.21.6 function diguyCharacterPathDecisionBead::get_length

Prototype:

```
float get_length();
```

Description:

This function returns where on the path this bead is.

Returns:

location of this bead, in meters

Callable From:

- ◇ C++
- ◇ Script

5.21.7 function diguyCharacterPathDecisionBead::set_length

Prototype:

```
int set_length(float length);
```

Description:

This function sets how long this bead will have an effect. The bead will have an effect from its initial distance into the path (as set by `set_distance_into_path( )`) to (distance + length) seconds.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting when on the path the bead's effect ends, as if to a call to `set_duration( )`.

If this script bead is part of a `diguyCharacterPath`, the `diguyCharacterPath::update( )` function must be called for changes to take effect.

Arguments:

`duration` duration of this bead, in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.21.8 function diguyCharacterPathDecisionBead::get_tin

Prototype:

```
float get_tin();
```

Description:

This function returns when on the path this bead is.

Returns:

time location of this bead, in seconds

Callable From:

- ◇ C++
- ◇ Script

5.21.9 function diguyCharacterPathDecisionBead::set_tin

Prototype:

```
int set_tin(float tin);
```

Description:

This function sets when this bead will begin to have an effect.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting where on the path the bead is located, as if to a call to [set_distance_into_path\(\)](#).

If this script bead is part of a [diguyCharacterPath](#), the [diguyCharacterPath::update\(\)](#) function must be called for changes to take effect.

Arguments:

`tin` when this bead begins to have an effect, in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.21.10 function diguyCharacterPathDecisionBead::get_duration

Prototype:

```
float get_duration();
```

Description:

This function returns how long this bead will have an effect.

Returns:

duration of this bead, in seconds

Callable From:

- ◇ C++
- ◇ Script

5.21.11 function diguyCharacterPathDecisionBead::set_duration**Prototype:**

```
int set_duration(float duration);
```

Description:

This function sets how long this bead will have an effect. The bead will have an effect from tin seconds (as set by `set_tin( )`) to (tin + duration) seconds.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting where on the path the bead's effect ends, as if to a call to `set_length( )`.

If this script bead is part of a `diguyCharacterPath`, the `diguyCharacterPath::update( )` function must be called for changes to take effect.

Arguments:

`duration` duration of this bead, in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Experimental Functions

The following functions are experimental and may change or disappear with no warning.

5.21.12 function

diguyCharacterPathDecisionBead::get_begin_at_object_name

Prototype:

```
const char* get_begin_at_object_name();
```

5.21.13 function

diguyCharacterPathDecisionBead::set_begin_at_object_name

Prototype:

```
int set_begin_at_object_name(const char* begin_at_object_name);
```

5.21.14 function

diguyCharacterPathDecisionBead::get_begin_at_object_offset

Prototype:

```
float get_begin_at_object_offset();
```

5.21.15 function**diguyCharacterPathDecisionBead::set_begin_at_object_offset****Prototype:**

```
int set_begin_at_object_offset(float begin_at_object_offset);
```

5.21.16 function**diguyCharacterPathDecisionBead::get_end_at_object_name****Prototype:**

```
const char* get_end_at_object_name();
```

5.21.17 function**diguyCharacterPathDecisionBead::set_end_at_object_name****Prototype:**

```
int set_end_at_object_name(const char* end_at_object_name);
```

5.21.18 function**diguyCharacterPathDecisionBead::get_end_at_object_offset****Prototype:**

```
float get_end_at_object_offset();
```

5.21.19 function diguyCharacterPathDecisionBead::set_end_at_object_offset

Prototype:

```
int set_end_at_object_offset(float end_at_object_offset);
```

5.22 diguyCharacterPathEvent

diguyCharacterPathEvent.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

Link against: libdiguy

5.22.1 class diguyCharacterPathEvent

```
class BDI_DECLSPEC_diguy diguyCharacterPathEvent
{
public:
    const char* get_name();
    diguyCharacter* get_character();
    diguyCharacterPath* get_path();
    float get_how_far();
```

5.22.2 function diguyCharacterPathEvent::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the bead. This pointer will never be NULL.

Returns:

name of the bead

Callable From:

- ◇ C++
- ◇ Script

5.22.3 function diguyCharacterPathEvent::get_character

Prototype:

```
diguyCharacter* get_character();
```

Description:

This function returns a pointer to the character owning this event bead.

Returns:

pointer of type [diguyCharacter](#)

Callable From:

- ◇ C++
- ◇ Script

5.22.4 function diguyCharacterPathEvent::get_path

Prototype:

```
diguyCharacterPath* get_path();
```

Description:

This function returns a pointer to the path owning this event bead.

Returns:

pointer of type [diguyCharacterPath](#)

Callable From:

- ◇ C++
- ◇ Script

5.22.5 function diguyCharacterPathEvent::get_how_far

Prototype:

```
float get_how_far();
```

Returns:

returns how far the character is into this bead. This is a normalized value. 0 means beginning of bead, 0.5 means halfway through, 1 means end of bead

Callable From:

- ◇ C++
- ◇ Script

5.23 diguyCharacterPathGazeBead

diguyCharacterPathGazeBead.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

◆ [Experimental Functions](#)

Link against: libdiguy

5.23.1 class diguyCharacterPathGazeBead

```
class BDI_DECLSPEC_diguy diguyCharacterPathGazeBead
{
public:
    const char* get_name();
    int set_name(const char* name);
    float get_distance_into_path();
    int set_distance_into_path(float distance_into_path);
    float get_length();
    int set_length(float length);
    float get_tin();
    int set_tin(float tin);
    float get_duration();
    int set_duration(float duration);
    const char* get_begin_at_object_name();
    int set_begin_at_object_name(const char* begin_at_object_name);
    float get_begin_at_object_offset();
    int set_begin_at_object_offset(float begin_at_object_offset);
    const char* get_end_at_object_name();
    int set_end_at_object_name(const char* end_at_object_name);
    float get_end_at_object_offset();
    int set_end_at_object_offset(float end_at_object_offset);
};
```

5.23.2 function diguyCharacterPathGazeBead::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the object

Callable From:

- ◇ C++
- ◇ Script

5.23.3 function diguyCharacterPathGazeBead::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.23.4 function**diguyCharacterPathGazeBead::get_distance_into_path****Prototype:**

```
float get_distance_into_path();
```

Description:

This function returns how far into the path this bead is.

Returns:

distance into path of this bead, in meters

Callable From:

- ◇ C++
- ◇ Script

5.23.5 function**diguyCharacterPathGazeBead::set_distance_into_path****Prototype:**

```
int set_distance_into_path(float distance_into_path);
```

Description:

This function sets how far into the path this bead is.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting when on the path the bead is located, as if to a call to [set_tin\(\)](#).

If this script bead is part of a [diguyCharacterPath](#), the [diguyCharacterPath::update\(\)](#) function must be called for changes to take effect.

Arguments:

`distance_into_path` how far into the path this bead is, in meters

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.23.6 function diguyCharacterPathGazeBead::get_length

Prototype:

```
float get_length();
```

Description:

This function returns where on the path this bead is.

Returns:

location of this bead, in meters

Callable From:

- ◇ C++
- ◇ Script

5.23.7 function diguyCharacterPathGazeBead::set_length

Prototype:

```
int set_length(float length);
```

Description:

This function sets how long this bead will have an effect. The bead will have an effect from its initial distance into the path (as set by `set_distance_into_path( )`) to (distance + length) seconds.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting when on the path the bead's effect ends, as if to a call to `set_duration( )`.

If this script bead is part of a `diguyCharacterPath`, the `diguyCharacterPath::update( )` function must be called for changes to take effect.

Arguments:

`duration` duration of this bead, in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.23.8 function diguyCharacterPathGazeBead::get_tin

Prototype:

```
float get_tin();
```

Description:

This function returns when on the path this bead is.

Returns:

time location of this bead, in seconds

Callable From:

- ◇ C++
- ◇ Script

5.23.9 function diguyCharacterPathGazeBead::set_tin

Prototype:

```
int set_tin(float tin);
```

Description:

This function sets when this bead will begin to have an effect.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting where on the path the bead is located, as if to a call to [set_distance_into_path\(\)](#).

If this script bead is part of a [diguyCharacterPath](#), the [diguyCharacterPath::update\(\)](#) function must be called for changes to take effect.

Arguments:

`tin` when this bead begins to have an effect, in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.23.10 function diguyCharacterPathGazeBead::get_duration

Prototype:

```
float get_duration();
```

Description:

This function returns how long this bead will have an effect.

Returns:

duration of this bead, in seconds

Callable From:

- ◇ C++
- ◇ Script

5.23.11 function diguyCharacterPathGazeBead::set_duration

Prototype:

```
int set_duration(float duration);
```

Description:

This function sets how long this bead will have an effect. The bead will have an effect from tin seconds (as set by `set_tin( )`) to (tin + duration) seconds.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting where on the path the bead's effect ends, as if to a call to `set_length( )`.

If this script bead is part of a `diguyCharacterPath`, the `diguyCharacterPath::update( )` function must be called for changes to take effect.

Arguments:

`duration` duration of this bead, in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Experimental Functions

The following functions are experimental and may change or disappear with no warning.

5.23.12 function

diguyCharacterPathGazeBead::get_begin_at_object_name

Prototype:

```
const char* get_begin_at_object_name();
```

5.23.13 function

diguyCharacterPathGazeBead::set_begin_at_object_name

Prototype:

```
int set_begin_at_object_name(const char* begin_at_object_name);
```

5.23.14 function

diguyCharacterPathGazeBead::get_begin_at_object_offset

Prototype:

```
float get_begin_at_object_offset();
```

5.23.15 function**diguyCharacterPathGazeBead::set_begin_at_object_offset****Prototype:**

```
int set_begin_at_object_offset(float begin_at_object_offset);
```

5.23.16 function**diguyCharacterPathGazeBead::get_end_at_object_name****Prototype:**

```
const char* get_end_at_object_name();
```

5.23.17 function `diguyCharacterPathGazeBead::set_end_at_object_name`

Prototype:

```
int set_end_at_object_name(const char* end_at_object_name);
```

5.23.18 function `diguyCharacterPathGazeBead::get_end_at_object_offset`

Prototype:

```
float get_end_at_object_offset();
```

5.23.19 function diguyCharacterPathGazeBead::set_end_at_object_offset

Prototype:

```
int set_end_at_object_offset(float end_at_object_offset);
```

5.24 diguyCharacterPathScriptBead

diguyCharacterPathScriptBead.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

◆ [Experimental Functions](#)

Link against: libdiguy

5.24.1 class diguyCharacterPathScriptBead

```
class BDI_DECLSPEC_diguy diguyCharacterPathScriptBead
{
public:
    const char* get_name();
    int set_name(const char* name);
    const char* get_script();
    int set_script(const char* script);
    int set_script_type(const char* script);
    const char* get_script_type();
    float get_distance_into_path();
    int set_distance_into_path(float distance_into_path);
    float get_length();
    int set_length(float length);
    float get_tin();
    int set_tin(float tin);
    float get_duration();
    int set_duration(float duration);
    const char* get_begin_at_object_name();
    int set_begin_at_object_name(const char* begin_at_object_name);
    float get_begin_at_object_offset();
    int set_begin_at_object_offset(float begin_at_object_offset);
    const char* get_end_at_object_name();
    int set_end_at_object_name(const char* end_at_object_name);
    float get_end_at_object_offset();
    int set_end_at_object_offset(float end_at_object_offset);
};
```

5.24.2 function diguyCharacterPathScriptBead::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the object

Callable From:

- ◇ C++
- ◇ Script

5.24.3 function diguyCharacterPathScriptBead::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.24.4 function diguyCharacterPathScriptBead::get_script

Prototype:

```
const char* get_script();
```

Description:

This function returns the script this bead performs.

Returns:

script to perform; empty string if no script set

Callable From:

- ◇ C++
- ◇ Script

5.24.5 function diguyCharacterPathScriptBead::set_script

Prototype:

```
int set_script(const char* script);
```

Description:

This function sets the script this bead performs. When a character is traversing the path containing this bead, the specified script will be performed when this bead is reached.

If this script bead is part of a [diguyCharacterPath](#), the [diguyCharacterPath::update\(\)](#) function must be called for changes to take effect.

Arguments:

<code>script</code>	the script that should be evaluated when this bead is reached; pass NULL to un-set the script

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.24.6 function diguyCharacterPathScriptBead::set_script_type

Prototype:

```
int set_script_type(const char* script);
```

Description:

This function sets the script type of this bead. This can only be "lua".

Arguments:

`script_type` the language of the script

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.24.7 function diguyCharacterPathScriptBead::get_script_type

Prototype:

```
const char* get_script_type();
```

Returns:

The language of the script

Callable From:

- ◇ C++
- ◇ Script

5.24.8 function diguyCharacterPathScriptBead::get_distance_into_path

Prototype:

```
float get_distance_into_path();
```

Description:

This function returns how far into the path this bead is.

Returns:

distance into path of this bead, in meters

Callable From:

- ◇ C++
- ◇ Script

5.24.9 function

diguyCharacterPathScriptBead::set_distance_into_path

Prototype:

```
int set_distance_into_path(float distance_into_path);
```

Description:

This function sets how far into the path this bead is.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting when on the path the bead is located, as if to a call to `set_tin()`.

If this script bead is part of a `diguyCharacterPath`, the `diguyCharacterPath::update()` function must be called for changes to take effect.

Arguments:

`distance_into_path` how far into the path this bead is, in meters

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.24.10 function diguyCharacterPathScriptBead::get_length**Prototype:**

```
float get_length();
```

Description:

This function returns where on the path this bead is.

Returns:

location of this bead, in meters

Callable From:

- ◇ C++
- ◇ Script

5.24.11 function diguyCharacterPathScriptBead::set_length**Prototype:**

```
int set_length(float length);
```

Description:

This function sets how long this bead will have an effect. The bead will have an effect from its initial distance into the path (as set by [set_distance_into_path\(\)](#)) to (distance + length) seconds.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting when on the path the bead's effect ends, as if to a call to [set_duration\(\)](#).

If this script bead is part of a [diguyCharacterPath](#), the [diguyCharacterPath::update\(\)](#) function must be called for changes to take effect.

Arguments:

`duration` duration of this bead, in seconds

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.24.12 function `diguyCharacterPathScriptBead::get_tin`

Prototype:

```
float get_tin();
```

Description:

This function returns when on the path this bead is.

Returns:

time location of this bead, in seconds

Callable From:

◇ C++
◇ Script

5.24.13 function diguyCharacterPathScriptBead::set_tin**Prototype:**

```
int set_tin(float tin);
```

Description:

This function sets when this bead will begin to have an effect.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting where on the path the bead is located, as if to a call to [set_distance_into_path\(\)](#).

If this script bead is part of a [diguyCharacterPath](#), the [diguyCharacterPath::update\(\)](#) function must be called for changes to take effect.

Arguments:

<code>tin</code>	when this bead begins to have an effect, in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.24.14 function diguyCharacterPathScriptBead::get_duration**Prototype:**

```
float get_duration();
```

Description:

This function returns how long this bead will have an effect.

Returns:

duration of this bead, in seconds

Callable From:

- ◇ C++
- ◇ Script

5.24.15 function diguyCharacterPathScriptBead::set_duration**Prototype:**

```
int set_duration(float duration);
```

Description:

This function sets how long this bead will have an effect. The bead will have an effect from tin seconds (as set by `set_tin( )`) to (tin + duration) seconds.

Since distances and times on paths are interrelated, calling this function will have the side-effect of setting where on the path the bead's effect ends, as if to a call to `set_length( )`.

If this script bead is part of a `diguyCharacterPath`, the `diguyCharacterPath::update( )` function must be called for changes to take effect.

Arguments:

`duration` duration of this bead, in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Experimental Functions

The following functions are experimental and may change or disappear with no warning.

5.24.16 function diguyCharacterPathScriptBead::get_begin_at_object_name

Prototype:

```
const char* get_begin_at_object_name();
```

5.24.17 function diguyCharacterPathScriptBead::set_begin_at_object_name

Prototype:

```
int set_begin_at_object_name(const char* begin_at_object_name);
```

5.24.18 function**diguyCharacterPathScriptBead::get_begin_at_object_offset****Prototype:**

```
float get_begin_at_object_offset();
```

5.24.19 function**diguyCharacterPathScriptBead::set_begin_at_object_offset****Prototype:**

```
int set_begin_at_object_offset(float begin_at_object_offset);
```

5.24.20 function

diguyCharacterPathScriptBead::get_end_at_object_name

Prototype:

```
const char* get_end_at_object_name();
```

5.24.21 function

diguyCharacterPathScriptBead::set_end_at_object_name

Prototype:

```
int set_end_at_object_name(const char* end_at_object_name);
```

5.24.22 function**diguyCharacterPathScriptBead::get_end_at_object_offset****Prototype:**

```
float get_end_at_object_offset();
```

5.24.23 function**diguyCharacterPathScriptBead::set_end_at_object_offset****Prototype:**

```
int set_end_at_object_offset(float end_at_object_offset);
```

5.25 diguyCharacterPoseOverride

diguyCharacterPoseOverride.h

DI-Guy SDK Version 12.5.2

Includes: stdlib.h | declspec_diguy.h

Contents:

5.25.1 class diguyCharacterPoseOverride

```
class BDI_DECLSPEC_diguy diguyCharacterPoseOverride
{
public:
#ifdef CPLUSPLUS_ONLY
    float* allocate_pose_array();
    int free_pose_array(float* array);
#endif
    int get_num_vars();
#ifdef CPLUSPLUS_ONLY
    char** allocate_var_names();
    int get_var_names(char** var_names);
    int free_var_names(char** var_names);
    int get_var_index(const char* varname);
    int get_pose_in_radians(float* pose_array);
    int set_pose_in_radians(float* pose_array,
        float* weights_array = NULL,
        float default_weight = 1.0f);
#endif
    float get_default_weight();
    int set_default_weight(float default_weight);
};
```

5.25.2 function

diguyCharacterPoseOverride::allocate_pose_array

Prototype:

```
float* allocate_pose_array();
```

Description:

Allocates an array of floats to hold the joint angles for each degree of freedom, or weights to specify how pose angles will affect a character. The size of the array will be equal to the value returned by [get_num_vars\(\)](#).

The allocated array should be freed by calling [free_pose_array\(\)](#).

The allocated array will contain uninitialized values.

If the array is to hold weights, each entry in the array should be set to a valid value between 0.0 and 1.0.

If the array is to hold pose angles, each pose variable whose corresponding weight is going to be non-0 should be set to a valid value.

Returns:

pointer to array of floats, NULL if the array cannot be allocated

Callable From:

◇ C++

C++ Example:

See example for [set_pose_in_radians\(\)](#).

5.25.3 function diguyCharacterPoseOverride::free_pose_array

Prototype:

```
int free_pose_array(float* array);
```

Description:

Frees an array allocated with `allocate_pose_array()`.

Arguments:

array	array of pose angles to be freed

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.25.4 function diguyCharacterPoseOverride::get_num_vars

Prototype:

```
int get_num_vars();
```

Description:

This function returns the number of the variables representing joint angles.

Returns:

an integer representing the number of joint angles

Callable From:

◇ C++
◇ Script

5.25.5 function

`diguyCharacterPoseOverride::allocate_var_names`

Prototype:

```
char** allocate_var_names();
```

Description:

This function allocates an array of pointer to `char*s` (i.e., C-style strings).

This array can then be passed to `get_var_names()`.

It should be freed using the function `free_var_names()`. The size of the array will be equal to the value returned by `get_num_vars()`.

Returns:

an array of pointers to `char*s` (i.e. pointers to strings)

Callable From:

◇ C++

C++ Example:

See example for `get_var_names()`.

5.25.6 function diguyCharacterPoseOverride::get_var_names

Prototype:

```
int get_var_names(char** var_names);
```

Description:

This function fills out the passed array of strings with the list of the names of the variables representing joint angles.

Arguments:

var_names an array of strings as allocated by `allocate_var_names()`

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

C++ Example:

```
//
// Create a pose override for character.
//
pose_override = character1->create_pose_override();

//
// Allocate a list of joint angle names and fill it.
// Print the joint angle names, then destroy the array.
//
char** varnames = pose_override->allocate_var_names();
pose_override->get_var_names(varnames);

for (i = 0; i < character1->get_pose_array_size(); i++)
    bdi_log_printf(BDI_LOG_INFO, "%s\n", varnames[i]);

pose_override->free_var_names(varnames);
```

5.25.7 function diguyCharacterPoseOverride::free_var_names**Prototype:**

```
int free_var_names(char** var_names);
```

Description:

This function frees the array of strings allocated by `allocate_var_names()`.

Arguments:

`var_names` an array of strings as allocated by `allocate_var_names()`

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.25.8 function diguyCharacterPoseOverride::get_var_index**Prototype:**

```
int get_var_index(const char* varname);
```

Description:

Get the index of the given variable in the array of pose angles.

Arguments:

`varname` the name of the pose angle variable whose index is desired

Returns:

index of the pose variable, if it exists, -1 if not

Callable From:

◇ C++

C++ Example:

See example for [set_pose_in_radians\(\)](#).

5.25.9 function `diguyCharacterPoseOverride::get_pose_in_radians`

Prototype:

```
int get_pose_in_radians(float* pose_array);
```

Description:

This function fills out the passed array of floats with the most recent values passed in a call to [set_pose_in_radians\(\)](#).

Note that this may not be the actual pose of the character; this is only the pose angles for this specific pose override. Other pose overrides, gazing, etc., will also affect the pose of the character. Use the function [diguyCharacter::get_pose_in_radians\(\)](#) to read the "real" pose of the character, as computed by the DI-Guy motion engine.

Arguments:

`pose_array` array of floats allocated by `allocate_pose_array()`

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.25.10 function diguyCharacterPoseOverride::set_pose_in_radians

Prototype:

```
int set_pose_in_radians(float* pose_array,
    float* weights_array = NULL,
    float default_weight = 1.0f);
```

Description:

This function specifies the joint angle data that will modify or override the joint angles the character normally computes. The replacement data is in pose_array.

If weights array is non-NULL, then each element specifies the proportion of the value on pose array to mix in with the value normally computed by the character. A weight of 0.0 means to ignore the value in pose_array. A weight of 1.0 means fully override the value computed by the character.

If weights_array is NULL, then default_weight will be used for **all** of the joint angles.

Note that all weights for an Euler triple (e.g. - q.cervical_rzrxry) must be equal to ensure proper interpolation.

Also note that the first six variables that contain the character's overall position and orientation cannot be changed using a pose override. These values must be changed using the functions

`diguyCharacter::set_position()` and
`diguyCharacter::set_orientation()`.

Arguments:

<code>pose_array</code>	the array containing the pose angles for this override object
<code>weight_array</code>	the multiplier used to determine how the new <code>pose_array</code> values are to be combined with the existing pose
<code>default_weight</code>	the value to be used to set the weights if <code>weights_array</code> is NULL

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

C++ Example:

```
//
// Create a pose override for character.
//
pose_override = character1->create_pose_override();

//
// Allocate an array for pose angles.
//
pose_array = pose_override->allocate_pose_array();
for (i=0; i<pose_override->get_num_vars(); i++)
    pose_array[i] = 0.0f;

//
// Allocate a weights_array, and initialize it to all 0s.
//
weights_array = pose_override->allocate_pose_array();
for (i=0; i<pose_override->get_num_vars(); i++)
    weights_array[i] = 0.0f;

//
// Find the indices of the character's back angles. Set the
// weights for these angles in weights_array to 1.0. This
// means that when we call set_pose_in_radians(), the values
// in pose_array will completely override the original
// data for the back.
```

```
//
back_rz_index = pose_override->get_var_index("q.back_rz");
back_rx_index = pose_override->get_var_index("q.back_rx");
back_ry_index = pose_override->get_var_index("q.back_ry");

pose_array[back_rz_index] = 0.57f; // angle measured in radians
pose_array[back_rx_index] = 0.0f; // angle measured in radians
pose_array[back_ry_index] = 0.0f; // angle measured in radians

weights_array[back_rz_index] = 1.0f;
weights_array[back_rx_index] = 1.0f;
weights_array[back_ry_index] = 1.0f;

pose_override->set_pose_in_radians(pose_array,
weights_array,
0.0f);
```

5.25.11 function diguyCharacterPoseOverride::get_default_weight

Prototype:

```
float get_default_weight();
```

Description:

Get the default weight for pose angles.

Returns:

the default pose angle weight

Callable From:

- ◇ C++
- ◇ Script

5.25.12 function

`diguyCharacterPoseOverride::set_default_weight`

Prototype:

```
int set_default_weight(float default_weight);
```

Description:

Set the default weight for pose angles.

The default weight will be used to set values in a pose's weights array if the array passed in, for instance, `set_pose_in_radians( )`, is NULL.

Arguments:

`default_weight` the desired default weight

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.26 diguyCharacterTypeMap

diguyCharacterTypeMap.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Type Map Fields](#)
- ◆ [Character Type Fields](#)
- ◆ [Appearance Fields](#)
- ◆ [Type Map Ids](#)
- ◆ [Type Map Query Result Functions](#)

5.26.1 enumeration diguyCharacterTypeMapField

```
typedef enum {
    /*!
     ** As string: "unknown"
     */
    DIGUY_CHARACTER_TYPE_MAP_FIELD_UNKNOWN = -1,

    /*!
     ** As string: "character_class"
     */
    DIGUY_CHARACTER_TYPE_MAP_FIELD_CHARACTER_CLASS = 0,

    /*!
     ** As string: "category"
     */
    DIGUY_CHARACTER_TYPE_MAP_FIELD_CATEGORY,

    /*!
     ** As string: "subcategory"
     */
    DIGUY_CHARACTER_TYPE_MAP_FIELD_SUBCATEGORY,

    /*!
     ** As string: "sex"
     */
    DIGUY_CHARACTER_TYPE_MAP_FIELD_SEX,

    /*!
     ** As string: "age"
     */
    DIGUY_CHARACTER_TYPE_MAP_FIELD_AGE,

    /*!
     ** As string: "race"
     */
    DIGUY_CHARACTER_TYPE_MAP_FIELD_RACE,
```



```

    /**
     ** As string: "country"
     */
    DIGUY_CHARACTER_TYPE_MAP_FIELD_COUNTRY,

    /**
     ** As string: "military_branch"
     */
    DIGUY_CHARACTER_TYPE_MAP_FIELD_MILITARY_BRANCH,

    /**
     ** As string: "carried_object"
     */
    DIGUY_CHARACTER_TYPE_MAP_FIELD_CARRIED_OBJECT,

    /**
     ** How many fields there are.
     */
    DIGUY_CHARACTER_TYPE_MAP_FIELD_COUNT

} diguyCharacterTypeMapField;

```

Description:

This enumeration allows type map fields to be queried using a numerical value rather than a string name of the field.

See the top-level description of the [diguyCharacterTypeMap](#) for information about what each field means.

5.26.2 function diguy_character_type_map_field_to_string

Prototype:

```
const char* diguy_character_type_map_field_to_string(diguyCharacterTypeMapField field)
```

5.26.3 function diguy_character_type_map_field_from_string

Prototype:

```
diguyCharacterTypeMapField diguy_character_type_map_field_from_string(const char* string)
```

The character type map provides extra information about DI-Guy character types and appearances. This information allows users and DI-Guy Scenario to query DI-Guy about the best match character type and appearance based on certain criteria.

Type maps are broken down into fields, described below.

For example, the function

`diguyScenario::get_nearest_character_type_map( )` will return a `diguyCharacterTypeMap` that best matches the specified class, category, subcategory, sex, age, etc.

The Appearance Wizard in DI-Guy Scenario, provides a graphical user interface for finding the best character types and appearances based on the same information.

Type Map Fields

Each type map is identified by a number of fields. The value in each field is a string.

The available fields are:

Refer to the online documentation page "Character Type and Appearance Map Info" for information on exactly what values are available for each field.

Certain field values have special meanings. A "*" is a "wildcard", meaning that in match operations it will match any value. If a type map doesn't specify a value for a field, that field value is by convention set to "-". For example, a human with no weapon would have its `carried_object` field set to "-".

When calling functions, fields can be specified numerically using the `diguyCharacterTypeMapField` enumeration, or each field has a string name, as well.

Character Type Fields

These fields tend to specify the "what it can do" part of a type map. These fields tend to be set by the character type, as the selection of actions provided with different character types tend to be what limits what a DI-Guy character can do.

Sometimes the selected appearance of a character will override the `category` and `subcategory` fields. For example, the `soldier_07` character type is most often `categorysubcategory` `militarysoldier`. A police officer appearance can be selected for a soldier character, however, which will change the `categorysubcategory` to `emergencypolice`.

character_class

The high-level "class" of a character type. This list may grow as more character types are added to DI-Guy. Examples:

- ◆ human
- ◆ object
- ◆ vehicle
- ◆ animal

category

A category of a character class. Different classes tend to have different categories. This list may grow as more character types are added to DI-Guy. Human class examples:

- ◆ military
- ◆ civilian
- ◆ emergency
- ◆ (more)

subcategory

A further breakdown of categories. Different categories tend to have different subcategories. This list may grow as more character types are added to DI-Guy. Human/emergency subclass examples:

- ◆ fire
- ◆ medical
- ◆ police
- ◆ (more)

Appearance Fields

These fields tend to specify the "what does it look like" part of a type map. These fields tend to be set by the appearance of a character, for obvious reasons.

sex

The gender of the appearance. Humans and animals and "male" or "female", while other classes usually have sex "none".

- ◆ male
- ◆ female
- ◆ none

age

The age of the appearance. This is an age description rather than a specific age in years.

- ◆ baby
- ◆ toddler
- ◆ child
- ◆ teenager

- ◆ adult
- ◆ elderly

race

The race of the appearance. (Note that these race classifications are not meant to offend. Please excuse any unintentional offense or omission.) Generally for human class only.

Examples:

- ◆ white
- ◆ black
- ◆ hispanic
- ◆ asian
- ◆ native_american
- ◆ middle_eastern
- ◆ none
- ◆ (more in the future)

country

The country of the appearance, if any. This list will grow as more appearances are added to DI-Guy. Examples:

- ◆ usa
- ◆ uk
- ◆ france
- ◆ japan
- ◆ iraq
- ◆ afghanistan
- ◆ chile
- ◆ russia
- ◆ none
- ◆ (more in the future)

military_branch

The military branch of the appearance, if any. This list may grow as more appearances are added to DI-Guy. Examples:

- ◆ army
- ◆ navy
- ◆ air_force
- ◆ marines
- ◆ coast_guard
- ◆ irregular
- ◆ none
- ◆ (more in the future)

carried_object

The object carried by the appearance, if any. Examples:

- ◆ m16
- ◆ m203
- ◆ 2x4
- ◆ hammer
- ◆ (more)

Type Map Ids

Some functions take or return a "type map id". This is a string in which the values of the type map's fields are concatenated in the form:

`character_classcategorysubcategorysexageracecountrymilitary_branchcarried_object`

For example, the string "humanmilitarysoldiermaleadultblackusaarmym16" is a valid type map id.

All public functions in this class are callable from:

- ◆ C++
- ◆ Script

5.26.4 class diguyCharacterTypeMap

```
class BDI_DECLSPEC diguy diguyCharacterTypeMap
{
public:
    const char* get_type_map_id();
    const char* get_mapped_character_type();
    const char* get_mapped_appearance();
    const char* get_field_value(diguyCharacterTypeMapField field);
    const char* get_field_value_using_field_string(const char* field);
    int get_most_recent_match_passed();
    int get_most_recent_match_level();
    int get_most_recent_exact_value_matches();
    int get_most_recent_alternate_value_matches();
    int get_most_recent_wildcard_value_matches();
    int get_num_wildcards();
};
```

5.26.5 function diguyCharacterTypeMap::get_type_map_id

Prototype:

```
const char* get_type_map_id();
```

Description:

This function returns the type map id of this type map object, in which the current value of all fields are concatenated together, separated by slashes.

An example return value is
"humanmilitarysoldiermaleadultblackusaarmym16".

Returns:

type map id string; will never be NULL

5.26.6 function `diguyCharacterTypeMap::get_mapped_character_type`

Prototype:

```
const char* get_mapped_character_type();
```

Description:

This function returns the character type this type map maps to. This is important for match results.

The returned character type can be used, for example, in a call to `diguyScenario::create_character()`.

Returns:

character type string; will never be NULL

5.26.7 function `diguyCharacterTypeMap::get_mapped_appearance`

Prototype:

```
const char* get_mapped_appearance();
```

Description:

This function returns the appearance this type map maps to. This is important for match results.

The returned appearance can be used, for example, in a call to `diguyCharacter::set_appearance()`.

Returns:

appearance string; will never be NULL

5.26.8 function diguyCharacterTypeMap::get_field_value

Prototype:

```
const char* get_field_value(diguyCharacterTypeMapField field);
```

Returns:

the value of the specified field

5.26.9 function diguyCharacterTypeMap::get_field_value_using_field_string

Prototype:

```
const char* get_field_value_using_field_string(const char* field);
```

Returns:

the value of the specified field

Type Map Query Result Functions

Some functions query DI-Guy for the closest match of character type and appearance to specified type map field selections. The result of such a query is returned in a [diguyCharacterTypeMap](#) object.

These functions return information from the returned object about whether there was actually a match, how many exact field matches there were, etc.

5.26.10 function `diguyCharacterTypeMap::get_most_recent_match_passed`

Prototype:

```
int get_most_recent_match_passed();
```

Returns:

whether the most recent match call passed or not; 1 if so, 0 if not

5.26.11 function `diguyCharacterTypeMap::get_most_recent_match_level`

Prototype:

```
int get_most_recent_match_level();
```

Returns:

total match value from most recent match operation; higher number means better match

5.26.12 function `diguyCharacterTypeMap::get_most_recent_exact_value_matches`

Prototype:

```
int get_most_recent_exact_value_matches();
```

Returns:

number of exact value matches in most recent match operation

5.26.13 function `diguyCharacterTypeMap::get_most_recent_alternate_value_match`

Prototype:

```
int get_most_recent_alternate_value_matches();
```

Returns:

number of alternate value matches in most recent match operation

5.26.14 function `diguyCharacterTypeMap::get_most_recent_wildcard_value_match`

Prototype:

```
int get_most_recent_wildcard_value_matches();
```

Returns:

number of wildcard value matches in most recent match operation

5.26.15 function diguyCharacterTypeMap::get_num_wildcards

Prototype:

```
int get_num_wildcards();
```

Returns:

number of fields that have wildcard values

5.27 diguyCrowd

diguyCrowd.h

DI-Guy SDK Version 12.5.2

Includes: [diguyMotionSpeedType.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ Crowd Member Functions
- ◆ Behavior Functions
 - ◇ Attack Functions
 - ◇ Flee Functions
 - ◇ Mingle Functions
 - ◇ Pursue Functions
 - ◇ Travel Functions
 - ◇ Wander Functions
 - ◇ Other Behavior Functions
- ◆ Query Functions
- ◆ Companion Crowd Functions
- ◆ Avoidance Method Functions
- ◆ Callback Functions
- ◆ Event Handler Functions
- ◆ Region Functions
- ◆ Message Functions
- ◆ Unit Functions
- ◆ Experimental Functions

5.27.1 class diguyCrowd

```
class BDI_DECLSPEC_diguy diguyCrowd
{
public:
    const char* get_name();
    int set_name(const char* name);
    const char* get_ui_name();
    int set_ui_name(const char* ui_name);
    const char* get_type_name();
    int set_enabled(int enabled);
    int get_enabled();
    void set_all_members_enabled(int enabled);
    float get_awareness_radius();
    int set_awareness_radius(float radius);
    int add_member(diguyCharacter* character);
    int add_all_characters_in_group(diguyCharacterGroup* group);
    int remove_member(diguyCharacter* character);
    int remove_all_members();
    int is_member(diguyCharacter* character);
    int get_num_members();
    diguyCharacter* get_member_at_index(int index);
};
```

```

int get_index_of_member(diguyCharacter* member);
diguyCharacterGroup* get_associated_group();
int agents_attack_character(const char* character_name);
int agents_attack_group(const char* group_name);
int agents_flee_character(const char* character_name,
    float distance,
    int danger_level = 50);
int agents_flee_group(const char* group_name,
    float distance,
    int danger_level = 50);
int agents_flee_crowd(const char* crowd_name,
    float distance,
    int danger_level = 50);
int agents_flee_location(float x, float y, float z,
    float distance,
    int danger_level = 50);
void agents_remove_all_flee_objects();
int agents_mingle_in_region(const char* region_name = NULL,
    const char* subregion = NULL);
int agents_pursue_character(const char* character_name);
int agents_pursue_group(const char* group_name);
int agents_chain_pursue(const char* leader_name);
int agents_travel_behavior_path(const char* direction = NULL,
    const char* at_end = NULL,
    int waypoint_index = -1,
    float offset_y = 0.5f);
int agents_travel_path_shape(const char* path_shape_name,
    const char* direction = NULL,
    const char* at_end = NULL,
    int waypoint_index = -1,
    float offset_y = 0.5f);
int agents_travel_region_border(const char* region_name,
    const char* subregion = NULL);
int agents_wander_region(const char* region_name = NULL, // NULL -> populate region
    const char* subregion = NULL);
int agents_stop_behavior();
int set_current_behavior(const char* behavior_name);
const char* get_current_behavior(int member_index = 0);
int set_current_behavior_path_shape(const char* behavior_path_shape_name,
    float max_time_until_move_on = 0.0f);
int set_current_behavior_region(const char* behavior_region_name,
    const char* subregion = NULL,
    float max_time_until_move_on = 0.0f);
int set_current_focus_character(const char* focus_character_name);
int set_current_focus_group(const char* focus_group_name);
int set_current_params_from_profile(const char* profile_name,
    int retain_focus_objects = 1,
    int retain_navigation_values = 1);
void set_current_behavior_region_border_is_solid(int is_solid);
void set_current_postures(diguyMotionPosture posture);
void set_current_variants(diguyMotionVariant variant);
int set_initial_params_from_profile(const char* profile_name);
int is_current_behavior(const char* behavior_name,
    float threshold = 0.5f);
int is_current_behavior_and_focus_character(const char* behavior_name,
    const char* focus_character_name,
    float threshold = 0.5f);
int is_current_behavior_and_focus_group(const char* behavior_name,
    const char* group_name,
    float threshold = 0.5f);
int is_crowd_dead(float threshold = 0.5f);

```

```

int add_companion_crowd(diguyCrowd* companion_crowd);
int remove_companion_crowd(diguyCrowd* companion_crowd);
void set_dynamic_object_avoidance_method(diguyDynamicAvoidanceMethod method);
diguyDynamicAvoidanceMethod get_dynamic_object_avoidance_method();
void set_static_object_avoidance_method(diguyStaticAvoidanceMethod method);
diguyStaticAvoidanceMethod get_static_object_avoidance_method();
diguyCharacter* get_callback_character();
diguyImpact* get_callback_impact();
enum {
    CALLBACK_ID_CREATE = 1,
    CALLBACK_ID_DESTROY,
    CALLBACK_ID_CROWD_MEMBER_KILLED,
    CALLBACK_ID_CROWD_MEMBER_IMPACT,
    CALLBACK_ID_NEARBY_SCENE_OBJECT_IMPACT,
    CALLBACK_ID_NEARBY_WEAPON_FIRED,
};
#ifdef CPLUSPLUS_ONLY
int add_callback(int callback_id,
    diguyCrowdCallback* callback,
    void* callback_params = 0,
    void* callback_user_data = 0);
int remove_callback(int callback_id,
    diguyCrowdCallback* callback);
int remove_callback_with_user_data(int callback_id,
    void* callback_user_data);
#endif
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int remove_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
#ifdef CPLUSPLUS_ONLY
void all_members_add_character_callback(int callback_id,
    diguyCharacterCallback* callback,
    void* callback_params = NULL,
    void* callback_user_data = NULL);
void all_members_remove_character_callback(int callback_id,
    diguyCharacterCallback* callback);
void all_members_remove_character_callback_with_user_data(int callback_id,
    void* callback_user_data);
#endif
void all_members_add_character_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
void all_members_remove_character_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int map_event_handler_to_callback_id(int callback_id,
    const char* handler_name);
int unmap_event_handler_from_callback_id(int callback_id,
    const char* handler_name,
    int unmap_all_matches = 0);
diguyRegion* find_or_create_region();
diguyRegion* get_region();
void send_message_to_all_members(const char* message_type,
    const char* message,
    const char* message_params = NULL);
void set_unit_type(diguyTacticsUnitType tut);
diguyTacticsUnitType get_unit_type();
int set_unit_leader(diguyCharacter* leader);

```


diguyCharacter*	get_unit_leader();

5.27.2 function diguyCrowd::get_name

Prototype:

```
const char* get_name();
```

Returns:

name of this object; this value will never be NULL

Callable From:

- ◇ C++
- ◇ Script

5.27.3 function diguyCrowd::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.27.4 function diguyCrowd::get_ui_name

Prototype:

```
const char* get_ui_name();
```

Returns:

UI name of this object; this value will never be NULL

Callable From:

- ◇ C++
- ◇ Script

5.27.5 function diguyCrowd::set_ui_name

Prototype:

```
int set_ui_name(const char* ui_name);
```

Description:

This function sets the UI name of this object. The UI name is what shows up in user interface windows that show information about this crowd.

For example, the crowd name may be "crowd_squad1", the UI name may be "Squad 1".

By default the UI name matches the internal name.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.27.6 function diguyCrowd::get_type_name

Prototype:

```
const char* get_type_name();
```

Description:

Returns the type name of the object. This pointer will never be NULL.

Returns:

type name of the object

Callable From:

- ◇ C++
- ◇ Script

5.27.7 function diguyCrowd::set_enabled

Prototype:

```
int set_enabled(int enabled);
```

Description:

This function enables or disables the crowd.

Disabling a crowd does not disable or hide the members of the crowd. It does turn off any effects the crowd may have on agent actions, position, orientation, etc. To enable or disable all agents in the crowd, use [set_all_members_enabled\(\)](#) instead.

Disabled crowds also will not interact with companion crowds.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.8 function diguyCrowd::get_enabled

Prototype:

```
int get_enabled();
```

Returns:

1 if enabled, 0 if not

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.9 function diguyCrowd::set_all_members_enabled

Prototype:

```
void set_all_members_enabled(int enabled);
```

Description:

This function enables or disables all agents in the crowd. This is the equivalent to calling [diguyCharacter::set_enabled\(\)](#) for all members of the crowd.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.10 function diguyCrowd::get_awareness_radius

Prototype:

```
float get_awareness_radius();
```

Returns:

The size of the radius around the crowd's character that certain callbacks will be triggered within. Callbacks such as:
 CALLBACK_ID_NEARBY_SCENE_OBJECT_IMPACT, and
 CALLBACK_ID_NEARBY_WEAPON_FIRED, are affected by this value.

Callable From:

- ◇ C++
- ◇ Script

5.27.11 function diguyCrowd::set_awareness_radius

Prototype:

```
int set_awareness_radius(float radius);
```

Description:

Sets the crowd awareness radius. See [get_awareness_radius\(\)](#).

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Crowd Member Functions

5.27.12 function diguyCrowd::add_member

Prototype:

```
int add_member(diguyCharacter* character);
```

Description:

Adds the passed character to the crowd if not already in it.

Returns:

0 if the character is now a part of the crowd; -1 if not

Callable From:

- ◇ C++
- ◇ Script

5.27.13 function diguyCrowd::add_all_characters_in_group**Prototype:**

```
int add_all_characters_in_group(diguyCharacterGroup* group);
```

Description:

Adds all characters of the group as members of the crowd.

Returns:

Always return 0.

Callable From:

- ◇ C++
- ◇ Script

5.27.14 function diguyCrowd::remove_member**Prototype:**

```
int remove_member(diguyCharacter* character);
```

Description:

Removes the passed character from the crowd if it is a member.

Returns:

0 if the character was part of the crowd and removed; -1 if not

Callable From:

- ◇ C++
- ◇ Script

5.27.15 function diguyCrowd::remove_all_members

Prototype:

```
int remove_all_members();
```

Description:

Removes all members of the crowd.

Returns:

Always return 0.

Callable From:

- ◇ C++
- ◇ Script

5.27.16 function diguyCrowd::is_member

Prototype:

```
int is_member(diguyCharacter* character);
```

Returns:

1 if the passed character is a member of the crowd; 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.27.17 function diguyCrowd::get_num_members**Prototype:**

```
int get_num_members();
```

Returns:

number of member characters in crowd

Callable From:

- ◇ C++
- ◇ Script

5.27.18 function diguyCrowd::get_member_at_index**Prototype:**

```
diguyCharacter* get_member_at_index(int index);
```

Returns:

crowd member at specified index; NULL if no character at specified index

Callable From:

- ◇ C++
- ◇ Script

5.27.19 function diguyCrowd::get_index_of_member**Prototype:**

```
int get_index_of_member(diguyCharacter* member);
```

Returns:

index of crowd member; -1 if not a member of the crowd

Callable From:

- ◇ C++
- ◇ Script

5.27.20 function diguyCrowd::get_associated_group**Prototype:**

```
diguyCharacterGroup* get_associated_group();
```

Returns:

[diguyCharacterGroup](#) associated with the crowd

Behavior Functions

Attack Functions

5.27.21 function diguyCrowd::agents_attack_character

Prototype:

```
int agents_attack_character(const char* character_name);
```

Description:

This is a shortcut for calling `diguyCharacter::agent_attack_character()` for all members of the crowd.

This function will implicitly put agents into the attack behavior.

The current focus character will be set the specified character. Pass NULL or the empty string ("") to not change the focus character, but still change to the pursue behavior.

The current focus group will be un-set.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.22 function diguyCrowd::agents_attack_group

Prototype:

```
int agents_attack_group(const char* group_name);
```

Description:

This is a shortcut for calling `diguyCharacter::agent_attack_group( )` for all members of the crowd.

This function will implicitly put agents into the attack behavior.

The current focus group will be set the specified group. Pass NULL or the empty string ("") to not change the focus group, but still change to the pursue behavior.

The current focus character will be un-set.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

Flee Functions

5.27.23 function diguyCrowd::agents_flee_character

Prototype:

```
int agents_flee_character(const char* character_name,  
    float distance,  
    int danger_level = 50);
```

Description:

This is a shortcut for calling `diguyCharacter::agent_flee_character ( )` for all members of the crowd.

This function will implicitly put agents into the flee behavior.

Neither the current focus character nor group will be changed by this function.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.24 function diguyCrowd::agents_flee_group**Prototype:**

```
int agents_flee_group(const char* group_name,  
    float distance,  
    int danger_level = 50);
```

Description:

This is a shortcut for calling `diguyCharacter::agent_flee_group( )` for all members of the crowd.

This function will implicitly put agents into the flee behavior.

Neither the current focus character nor group will be changed by this function.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.25 function diguyCrowd::agents_flee_crowd

Prototype:

```
int agents_flee_crowd(const char* crowd_name,
    float distance,
    int danger_level = 50);
```

Description:

This is a shortcut for calling `diguyCharacter::agent_flee_crowd( )` for all members of the crowd.

This function will implicitly put agents into the flee behavior.

Neither the current focus character nor group will be changed by this function.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.26 function diguyCrowd::agents_flee_location**Prototype:**

```
int agents_flee_location(float x, float y, float z,  
    float distance,  
    int danger_level = 50);
```

Description:

This is a shortcut for calling `diguyCharacter::agent_flee_location()` for all members of the crowd.

This function will implicitly put agents into the flee behavior.

Neither the current focus character nor group will be changed by this function.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.27.27 function diguyCrowd::agents_remove_all_flee_objects

Prototype:

```
void agents_remove_all_flee_objects();
```

Description:

This is a shortcut for calling `diguyCharacter::agent_remove_all_flee_objects( )` for all members of the crowd.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

Mingle Functions

5.27.28 function diguyCrowd::agents_mingle_in_region

Prototype:

```
int agents_mingle_in_region(const char* region_name = NULL,  
    const char* subregion = NULL);
```

Description:

This is a shortcut for calling `diguyCharacter::agent_mingle_in_region()` for all members of the crowd.

This function will implicitly put agents into the mingle behavior.

Neither the current focus character nor group will be changed by this function.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

Pursue Functions

5.27.29 function diguyCrowd::agents_pursue_character

Prototype:

```
int agents_pursue_character(const char* character_name);
```

Description:

This is a shortcut for calling `diguyCharacter::agent_pursue_character()` for all members of the crowd.

This function will implicitly put agents into the pursue behavior.

The current focus character will be set the specified character. Pass NULL or the empty string ("") to not change the focus character, but still change to the pursue behavior.

The current focus group will be un-set.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.30 function diguyCrowd::agents_pursue_group**Prototype:**

```
int agents_pursue_group(const char* group_name);
```

Description:

This is a shortcut for calling `diguyCharacter::agent_pursue_group( )` for all members of the crowd.

This function will implicitly put agents into the pursue behavior.

The current focus group will be set to the specified group. Pass NULL or the empty string ("") to not change the focus group, but still change to the pursue behavior.

The current focus character will be un-set.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.31 function diguyCrowd::agents_chain_pursue

Prototype:

```
int agents_chain_pursue(const char* leader_name);
```

Description:

This causes all members of the crowd to start pursuing the individual that is closer to the leader than they are.

This function will implicitly put all agents except the leader into the pursue behavior.

The current focus character will be set to the next nearest character.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++

◇ Script
◇ Decision

Travel Functions

5.27.32 function diguyCrowd::agents_travel_behavior_path

Prototype:

```
int agents_travel_behavior_path(const char* direction = NULL,  
    const char* at_end = NULL,  
    int waypoint_index = -1,  
    float offset_y = 0.5f);
```

Description:

This is a shortcut for calling
`diguyCharacter::agent_travel_behavior_path( )` for all
members of the crowd.

This function will implicitly put agents into the travel behavior.

Neither the current focus character nor group will be changed by this
function.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.33 function diguyCrowd::agents_travel_path_shape**Prototype:**

```
int agents_travel_path_shape(const char* path_shape_name,
    const char* direction = NULL,
    const char* at_end = NULL,
    int waypoint_index = -1,
    float offset_y = 0.5f);
```

Description:

This is a shortcut for calling `diguyCharacter::agent_travel_path_shape( )` for all members of the crowd.

This function will implicitly put agents into the travel behavior.

Neither the current focus character nor group will be changed by this function.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.34 function diguyCrowd::agents_travel_region_border

Prototype:

```
int agents_travel_region_border(const char* region_name,
    const char* subregion = NULL);
```

Description:

This is a shortcut for calling `diguyCharacter::agent_travel_region_border()` for all members of the crowd.

This function will implicitly put agents into the travel behavior.

Neither the current focus character nor group will be changed by this function.

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script
◇ Decision

Wander Functions

5.27.35 function diguyCrowd::agents_wander_region

Prototype:

```
int agents_wander_region(const char* region_name = NULL, // NULL -> populate region  
                        const char* subregion = NULL);
```

Description:

This is a shortcut for calling `diguyCharacter::agent_wander_region( )` for all members of the crowd.

This function will implicitly put agents into the wander behavior.

Neither the current focus character nor group will be changed by this function.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

Other Behavior Functions

5.27.36 function diguyCrowd::agents_stop_behavior

Prototype:

```
int agents_stop_behavior();
```

Description:

This is a shortcut for calling `diguyCharacter::agent_stop_behavior ( )` for all members of the crowd.

This function will implicitly put agents into the none behavior.

Neither the current focus character nor group will be changed by this function.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.37 function diguyCrowd::set_current_behavior

Prototype:

```
int set_current_behavior(const char* behavior_name);
```

Description:

This is a shortcut for calling diguyCharacter::set_current_behavior() for all members of the crowd.

It's recommended that one of the higher level functions above that set behavior, for example agent_wander_region(), is used. These functions allow for setting the most common parameters that affect the behavior in one function call.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++

- ◇ Script
- ◇ Decision

5.27.38 function diguyCrowd::get_current_behavior

Prototype:

```
const char* get_current_behavior(int member_index = 0);
```

Description:

Returns the current behavior used by crowd, expressed as a string (e.g. "travel").

Returns:

string describing current crowd behavior, NULL on error

Callable From:

- ◇ C++
- ◇ Script

5.27.39 function diguyCrowd::set_current_behavior_path_shape**Prototype:**

```
int set_current_behavior_path_shape(const char* behavior_path_shape_name,
    float max_time_until_move_on = 0.0f);
```

Description:

Similar to [diguyAgentParams::set_behavior_path_shape\(\)](#). This is a shortcut for setting the current behavior path shape for all characters in the crowd.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.40 function diguyCrowd::set_current_behavior_region**Prototype:**

```
int set_current_behavior_region(const char* behavior_region_name,
    const char* subregion = NULL,
    float max_time_until_move_on = 0.0f);
```

Description:

Similar to [diguyAgentParams::set_behavior_region\(\)](#). This is a shortcut for setting the current behavior region for all characters in the crowd.

This function does take an additional argument, `max_time_until_move_on`. Setting this to 0 will cause all agents that occasionally pause their movement (for example, characters in the wander behavior) to move on to the new behavior region immediately. Passing a higher number will allow agents to finish their "natural" wait interval before moving on to the new region.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.41 function diguyCrowd::set_current_focus_character

Prototype:

```
int set_current_focus_character(const char* focus_character_name);
```

Description:

Similar to [diguyAgentParams::set_focus_character\(\)](#). This is a shortcut for setting the current focus character for all characters in the crowd.

The current focus group will not be affected by this function.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.42 function diguyCrowd::set_current_focus_group**Prototype:**

```
int set_current_focus_group(const char* focus_group_name);
```

Description:

Similar to [diguyAgentParams::set_focus_group\(\)](#). This is a shortcut for setting the current focus group for all characters in the crowd.

The current focus character will not be affected by this function.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.43 function diguyCrowd::set_current_params_from_profile**Prototype:**

```
int set_current_params_from_profile(const char* profile_name,
    int retain_focus_objects = 1,
    int retain_navigation_values = 1);
```

Description:

This is a shortcut for calling [diguyCharacter::agent_set_current_params_from_profile\(\)](#) for all members of the crowd.

This will affect most of the parameters of agents in this crowd, including behavior, posture and variant, etc. Focus character and group by default are changed, but can be left alone by passing 0 for retain_focus_objects.

Arguments:

profile_name	name of crowd profile whose default parameters should be used

`retain_focus_objects` pass 1 to change focus objects to those of the named profile, 0 to retain existing settings

`retain_navigation_values` Documentation Pending

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.44 function `diguyCrowd::set_current_behavior_region_border_is_solid`

Prototype:

```
void set_current_behavior_region_border_is_solid(int is_solid);
```

Description:

This is a shortcut for calling `diguyCharacter::agent_set_current_behavior_region_border_is_solid()` for all members of the crowd.

Sets whether the walls of the region the agent is in are solid. If they are, the agent will treat the walls as if they are solid walls in the environment. If not, the agent is free to walk through them.

Pass 1 to make border walls solid, 0 to make them freely passable.

Callable From:

- ◇ C++

- ◇ Script
- ◇ Decision

5.27.45 function diguyCrowd::set_current_postures

Prototype:

```
void set_current_postures(diguyMotionPosture posture);
```

Description:

Sets the posture of all members of the crowd and restarts their behaviors.

5.27.46 function diguyCrowd::set_current_variants**Prototype:**

```
void set_current_variants(diguyMotionVariant variant);
```

Description:

Sets the variant of all members of the crowd and restarts their behaviors.

5.27.47 function diguyCrowd::set_initial_params_from_profile**Prototype:**

```
int set_initial_params_from_profile(const char* profile_name);
```

Description:

This function is similar to [set_current_params_from_profile\(\)](#), but instead sets initial params.

Arguments:

`profile_name` name of crowd profile whose default parameters should be used

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Query Functions

5.27.48 function diguyCrowd::is_current_behavior**Prototype:**

```
int is_current_behavior(const char* behavior_name,
    float threshold = 0.5f);
```

Description:

This is a query function meant to allow for flexible behavior when dealing with a crowd. If more than threshold ratio are doing behavior_name then the function will return 1. The default 0.5 yields a simple majority check.

Returns:

1 if true, 0 if false

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.27.49 function diguyCrowd::is_current_behavior_and_focus_character**Prototype:**

```
int is_current_behavior_and_focus_character(const char* behavior_name,
    const char* focus_character_name,
    float threshold = 0.5f);
```

Description:

Similar to [is_current_behavior\(\)](#), but checks both behavior and focus character.

5.27.50 function `diguyCrowd::is_current_behavior_and_focus_group`

Prototype:

```
int is_current_behavior_and_focus_group(const char* behavior_name,  
    const char* group_name,  
    float threshold = 0.5f);
```

Description:

Similar to `is_current_behavior()`, but checks both behavior and focus group.

5.27.51 function `diguyCrowd::is_crowd_dead`

Prototype:

```
int is_crowd_dead(float threshold = 0.5f);
```

Description:

Similar to `is_current_behavior()`, but checks whether characters are dead.

Companion Crowd Functions

5.27.52 function diguyCrowd::add_companion_crowd

Prototype:

```
int add_companion_crowd(diguyCrowd* companion_crowd);
```

Description:

This function makes the specified crowd a "companion crowd" of this crowd. Crowd members of companion crowds know about each other and will attempt to avoid each other.

If two crowds are not companions, their respective crowd members will make no attempt to avoid each other.

Arguments:

`companion_crowd` name of crowd to become a companion

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.27.53 function diguyCrowd::remove_companion_crowd

Prototype:

```
int remove_companion_crowd(diguyCrowd* companion_crowd);
```

Description:

This function removes the specified crowd as a companion crowd.

Arguments:

`companion_crowd` name of former companion

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Avoidance Method Functions

5.27.54 function diguyCrowd::set_dynamic_object_avoidance_method

Prototype:

```
void set_dynamic_object_avoidance_method(diguyDynamicAvoidanceMethod method);
```

Description:

This function sets the dynamic avoidance method that members of this crowd will use to avoid each other. Dynamic objects are objects that are moving around; these are almost always characters.

The enumeration is defined in diguy_constants.h.

Arguments:

method	avoidance method to use

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.27.55 function `diguyCrowd::get_dynamic_object_avoidance_method`

Prototype:

```
diguyDynamicAvoidanceMethod get_dynamic_object_avoidance_method();
```

Returns:

current dynamic object avoidance method; see
[`set\_dynamic\_object\_avoidance\_method\(\)`](#)

Callable From:

- ◇ C++
- ◇ Script

5.27.56 function `diguyCrowd::set_static_object_avoidance_method`

Prototype:

```
void set_static_object_avoidance_method(diguyStaticAvoidanceMethod method);
```

Description:

This function sets the static avoidance method that members of this crowd will use to avoid walking through fixed walls and objects. Usually these are scene objects and fixed props.

The enumeration is defined in `diguy_constants.h`.

Arguments:

<code>method</code>	avoidance method to use

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.27.57 function `diguyCrowd::get_static_object_avoidance_method`

Prototype:

```
diguyStaticAvoidanceMethod get_static_object_avoidance_method();
```

Returns:

current static object avoidance method; see
[`set\_static\_object\_avoidance\_method\(\)`](#)

Callable From:

- ◇ C++
- ◇ Script

Callback Functions

5.27.58 function diguyCrowd::get_callback_character**Prototype:**

```
diguyCharacter* get_callback_character();
```

Description:

Many of the [diguyCrowd](#) callbacks set a callback character which represents if there was a particular character that the callback relates to. A few of the callbacks do not set this value.

Returns:

pointer of type [diguyCharacter](#); NULL if there was no associated character

Callable From:

- ◇ C++
- ◇ Script

5.27.59 function diguyCrowd::get_callback_impact**Prototype:**

```
diguyImpact* get_callback_impact();
```

Description:

Many of the [diguyCrowd](#) callbacks set a callback impact which represents if there was impact that the callback relates to. A few of the callbacks do not set this value.

Returns:

pointer of type [diguyImpact](#); NULL if there was no associated impact

Callable From:

- ◇ C++
- ◇ Script

5.27.60 enumeration

```
enum {
    CALLBACK_ID_CREATE = 1,
    CALLBACK_ID_DESTROY,
    CALLBACK_ID_CROWD_MEMBER_KILLED,
    CALLBACK_ID_CROWD_MEMBER_IMPACT,
    CALLBACK_ID_NEARBY_SCENE_OBJECT_IMPACT,
    CALLBACK_ID_NEARBY_WEAPON_FIRED,
};
```

Description:

This is an enumeration of the different callbacks that can be registered with [add_callback\(\)](#) and [add_callback_script\(\)](#).

CALLBACK_ID_CREATE

This callback will be called when a new crowd is created.

Note that this callback can only be added by calling `diguyScenario::add_default_crowd_callback()`; adding it with [diguyCrowd::add_callback\(\)](#) will have no effect, as by that time the crowd has already been created. There is no associated callback character.

CALLBACK_ID_DESTROY

This callback will be called when a crowd is destroyed. There is no associated callback character.

CALLBACK_ID_CROWD_MEMBER_KILLED

This callback will be called when a member of the crowd is killed.

- ◊ The callback character is the crowd member that was killed.
- ◊ The callback impact contains the impact information.

CALLBACK_ID_CROWD_MEMBER_IMPACT

Similar to `diguyCharacter::CALLBACK_ID_IMPACT`, this callback will be called when a member of the crowd is hit by a detonation. If this callback isn't present the impacted character automatically will die. Handling this callback allows for the implementation of custom damage models at a crowd level.

- ◊ The callback character is the crowd member that was hit.
- ◊ The callback impact contains the impact information.

CALLBACK_ID_NEARBY_SCENE_OBJECT_IMPACT

This callback will be called when a detonation occurs within the awareness radius (as set by `set_awareness_radius( )`) of the crowd's current bounds.

- ◇ The callback character is the character that caused the detonation.
- ◇ The callback impact contains the impact information.

CALLBACK_ID_NEARBY_WEAPON_FIRED

This callback will be called when a weapon is fired within the awareness radius (as set by `set_awareness_radius( )`) of the crowd's current bounds.

- ◇ The callback character is the character that fired the weapon.

Callbacks return a value of type `diguyCallbackReturn`, which will be `DIGUY_CALLBACK_STOP` or `DIGUY_CALLBACK_CONTINUE`. If the callback returns `DIGUY_CALLBACK_STOP`, the default handler of the function will not be called; the callback is asserting that it has done everything necessary for the function call. If the callback returns `DIGUY_CALLBACK_CONTINUE`, the default handler for the function will be called after the callback.

Usable From:

- ◇ C++
- ◇ Script

5.27.61 function diguyCrowd::add_callback

Prototype:

```
int add_callback(int callback_id,
                 diguyCrowdCallback* callback,
                 void* callback_params = 0,
                 void* callback_user_data = 0);
```

Description:

This function adds a user callback. Callbacks can be removed with [remove_callback\(\)](#) or [remove_callback_with_user_data\(\)](#).

Arguments:

callback	pointer to function with prototype diguyCrowdRegionCallback (typedefed above)
callback_id	integer id of when this callback is to be called
callback_params	not currently used; pass NULL
callback_user_data	pointer for user's own use; DI-Guy will do nothing to the contents of this pointer beyond passing it back when the callback is invoked

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.27.62 function diguyCrowd::remove_callback**Prototype:**

```
int remove_callback(int callback_id,
                    diguyCrowdCallback* callback);
```

Description:

This function removes a user callback. All callbacks matching the specified callback_id and callback function will be removed.

Arguments:

callback_id	integer id of when this callback is to be called
callback	pointer to function with prototype diguyCrowdCallback (typedefed above)

Returns:

0 on success, -1 on failure

5.27.63 function diguyCrowd::remove_callback_with_user_data**Prototype:**

```
int remove_callback_with_user_data(int callback_id,
                                   void* callback_user_data);
```

Description:

This function removes a user callback. All callbacks matching the specified callback_id and callback_user_data pointer will be removed.

Arguments:

callback_id	integer id of when this callback is to be called
callback_user_data	pointer for user's own use

Returns:

0 on success, -1 on failure

5.27.64 function diguyCrowd::add_callback_script**Prototype:**

```
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function adds a user callback script. Callback scripts can be removed with [remove_callback_script\(\)](#).

See [diguyCharacter::add_callback_script\(\)](#) for an example of use.

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback to be added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

lua specific:

When the script is called, the object for which it is being called will be in the `callback_object` global.

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.27.65 function `diguyCrowd::remove_callback_script`

Prototype:

```
int remove_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function removes a user callback script previously added with `add_callback_script()`.

See `diguyCharacter::remove_callback_script()` for an example of use.

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback previously added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script`, all callback scripts whose ids match `callback_id` and whose types match `callback_script_type` will be removed.

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

lua specific:

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.27.66 function

diguyCrowd::all_members_add_character_callback

Prototype:

```
void all_members_add_character_callback(int callback_id,
    diguyCharacterCallback* callback,
    void* callback_params = NULL,
    void* callback_user_data = NULL);
```

Description:

Equivalent to calling `diguyCharacter::add_callback()` for all members of the crowd.

Note: `callback_id` should be a **diguyCharacter** id, not a **diguyCrowd** id.

5.27.67 function

diguyCrowd::all_members_remove_character_callback

Prototype:

```
void all_members_remove_character_callback(int callback_id,
    diguyCharacterCallback* callback);
```

Description:

Equivalent to calling `diguyCharacter::remove_callback()` for all members of the crowd.

Note: `callback_id` should be a **diguyCharacter** id, not a **diguyCrowd** id.

5.27.68 function diguyCrowd::all_members_remove_character_callback_with_use

Prototype:

```
void all_members_remove_character_callback_with_user_data(int callback_id,
void* callback_user_data);
```

Description:

Equivalent to calling
[diguyCharacter::remove_callback_with_user_data\(\)](#) for
all members of the crowd.

Note: callback_id should be a **diguyCharacter** id, not a [diguyCrowd](#) id.

5.27.69 function diguyCrowd::all_members_add_character_callback_script

Prototype:

```
void all_members_add_character_callback_script(int callback_id,
const char* callback_script,
const char* callback_script_type = NULL);
```

Description:

Equivalent to calling [diguyCharacter::add_callback_script\(\)](#)
for all members of the crowd.

Note: callback_id should be a **diguyCharacter** id, not a [diguyCrowd](#) id.

5.27.70 function**diguyCrowd::all_members_remove_character_callback_script****Prototype:**

```
void all_members_remove_character_callback_script(int callback_id,  
const char* callback_script,  
const char* callback_script_type = NULL);
```

Description:

Equivalent to calling
[diguyCharacter::remove_callback_script\(\)](#) for all members
of the crowd.

Note: callback_id should be a **diguyCharacter** id, not a [diguyCrowd](#) id.

Event Handler Functions

5.27.71 function

diguyCrowd::map_event_handler_to_callback_id

Prototype:

```
int map_event_handler_to_callback_id(int callback_id,
    const char* handler_name);
```

Description:

This function maps the event handler with the given name to a callback id. This mapping will be saved in the .dss file and restored when the .dss file is loaded.

Mappings can also be made via the DI-Guy Scenario UI.

The event handler is one of the following:

- ◇ a scene object callback function registered by
diguyScenario::register_crowd_event_handler() or
diguyScenario::register_crowd_event_handler_from_library()
- ◇ a script registered by
diguyScenario::register_crowd_event_handler_script()
- ◇ a Script, Decision, or Library Function in the scenario whose "Event Type" is "Crowd"

Arguments:

`callback_id` integer id of callback
`handler_name` name of the event handler to map

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.27.72 function**diguyCrowd::unmap_event_handler_from_callback_id****Prototype:**

```
int unmap_event_handler_from_callback_id(int callback_id,
    const char* handler_name,
    int unmap_all_matches = 0);
```

Description:

This function unmaps the event handler with the given name from a callback id.

Arguments:

callback_id	integer id of callback
handler_name	name of the event handler to map
unmap_all_matches	pass 0 to unmap only the first match, pass 1 to unmap all matches

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Region Functions

5.27.73 function diguyCrowd::find_or_create_region

Prototype:

```
diguyRegion* find_or_create_region();
```

Returns:

[diguyRegion](#) associated with the crowd; will be created if it doesn't currently exist

5.27.74 function diguyCrowd::get_region

Prototype:

```
diguyRegion* get_region();
```

Returns:

[diguyRegion](#) associated with the crowd; may be NULL

Message Functions

5.27.75 function diguyCrowd::send_message_to_all_members**Prototype:**

```
void send_message_to_all_members(const char* message_type,
                                const char* message,
                                const char* message_params = NULL);
```

Description:

Equivalent to calling
[diguyCharacter::agent_accept_message\(\)](#) for all members of
 the crowd.

Arguments:

message_type	what type of message is being sent
message	the message string
message_params	optional parameter string

Callable From:

- ◇ C++
- ◇ Script

Unit Functions

5.27.76 function diguyCrowd::set_unit_type**Prototype:**

```
void set_unit_type(diguyTacticsUnitType tut);
```

5.27.77 function diguyCrowd::get_unit_type**Prototype:**

```
diguyTacticsUnitType get_unit_type();
```

5.27.78 function diguyCrowd::set_unit_leader**Prototype:**

```
int set_unit_leader(diguyCharacter* leader);
```

5.27.79 function diguyCrowd::get_unit_leader**Prototype:**

```
diguyCharacter* get_unit_leader();
```

Experimental Functions

The following functions are experimental and may change or disappear with no warning.

5.28 diguyCrowdProfile

diguyCrowdProfile.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

◆ Deprecated Functions

5.28.1 class diguyCrowdProfile

```
class BDI_DECLSPEC_diguy diguyCrowdProfile
{
public:
    const char* get_name();
    int set_name(const char* name);
    const char* get_type_name();
    int create_entry(const char* character_type = NULL,
        const char* appearance = NULL,
        float proportion = 1.0f);
    void destroy_entry(int entry_index);
    int get_num_entries();
    int get_random_entry_index();
    int set_character_type(int entry_index, const char* character_type);
    const char* get_character_type(int entry_index);
    int set_appearance(int entry_index, const char* appearance);
    const char* get_appearance(int entry_index);
    int set_mind_base_class(int entry_index, const char* mind);
    const char* get_mind_base_class(int entry_index);
    int set_proportion(int entry_index, float proportion);
    float get_proportion(int entry_index);
    int set_default_crowd_size(int default_crowd_size);
    int get_default_crowd_size();
    diguyAgentParams* get_default_agent_params();
    diguyCharacter * create_character(diguyCrowd * crowd, int is_temporary = 0);
    const char* get_mind_init_argument(int entry_index, int arg);
    int set_character_type_and_appearance(int entry_index,
        const char* character_type,
        const char* appearance);
    int add_entry(const char* character_type, const char* appearance);
```

5.28.2 function diguyCrowdProfile::get_name

Prototype:

```
const char* get_name();
```

Returns:

name of this object; this value will never be NULL

Callable From:

- ◇ C++
- ◇ Script

5.28.3 function diguyCrowdProfile::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.28.4 function diguyCrowdProfile::get_type_name

Prototype:

```
const char* get_type_name();
```

Returns:

type name of the object; this value will never be NULL

Callable From:

- ◇ C++
- ◇ Script

5.28.5 function diguyCrowdProfile::create_entry

Prototype:

```
int create_entry(const char* character_type = NULL,
                const char* appearance = NULL,
                float proportion = 1.0f);
```

Description:

This function creates a new crowd profile entry. The entry is initialized with the passed parameters.

Returns:

the index of the created entry; -1 if no entry created

Arguments:

<code>character_type</code>	see <code>set_character_type()</code>
<code>appearance</code>	see <code>set_appearance()</code>
<code>proportion</code>	see <code>set_proportion()</code>

If NULL is passed for `character_type`, the `set_character_type()` function must be called with a valid character type before this new entry is valid.

Callable From:

- ◇ C++
- ◇ Script

5.28.6 function `diguyCrowdProfile::destroy_entry`

Prototype:

```
void destroy_entry(int entry_index);
```

Description:

This function destroys the crowd profile entry at the specified index.

Callable From:

- ◇ C++
- ◇ Script

5.28.7 function diguyCrowdProfile::get_num_entries

Prototype:

```
int get_num_entries();
```

Returns:

number of entries in this crowd profile

Callable From:

- ◇ C++
- ◇ Script

5.28.8 function diguyCrowdProfile::get_random_entry_index

Prototype:

```
int get_random_entry_index();
```

Returns:

a random entry index from all entries in this profile; entries with higher proportions are more likely to be returned

Callable From:

- ◇ C++
- ◇ Script

5.28.9 function diguyCrowdProfile::set_character_type

Prototype:

```
int set_character_type(int entry_index, const char* character_type);
```

Description:

This function sets the character type of the given entry. This character type will be used for some characters in crowds created using this crowd profile.

On creation, the character type is set to "(none)".

Returns:

0 on success, -1 on failure

Arguments:

entry_index	index of entry to be modified as returned by create_entry()
character_type	string identifying character type to be associated with this entry; valid values are the same as those that can be passed to diguyScenario::create_character()

Callable From:

- ◇ C++
- ◇ Script

5.28.10 function diguyCrowdProfile::get_character_type

Prototype:

```
const char* get_character_type(int entry_index);
```

Returns:

the character type of the specified entry; "(none)" if there is no such entry, or if the specified entry has not yet had its character type set

Arguments:

`entry_index` index of entry to be queried as returned by `create_entry()`

Callable From:

- ◇ C++
- ◇ Script

5.28.11 function diguyCrowdProfile::set_appearance

Prototype:

```
int set_appearance(int entry_index, const char* appearance);
```

Description:

This function sets the appearance of the given entry. This appearance will be used for some characters in crowds created using this crowd profile.

On creation, the appearance is set to "default".

Returns:

0 on success, -1 on failure

Arguments:

`entry_index` index of entry to be modified as returned by `create_entry()`
`appearance` appearance to be associated with the new entry; passing NULL sets appearance to "default"

Callable From:

- ◇ C++
- ◇ Script

5.28.12 function diguyCrowdProfile::get_appearance

Prototype:

```
const char* get_appearance(int entry_index);
```

Returns:

the appearance of the specified entry; NULL will be returned if there is no such entry

Arguments:

`entry_index` index of entry to be queried as returned by `create_entry()`

Callable From:

- ◇ C++
- ◇ Script

5.28.13 function diguyCrowdProfile::set_mind_base_class**Prototype:**

```
int set_mind_base_class(int entry_index, const char* mind);
```

Description:

This function sets the mind of the given entry. This mind will be used for some characters in crowds created using this crowd profile.

On creation, the mind is set to "none".

Returns:

0 on success, -1 on failure

Arguments:

<code>entry_index</code>	index of entry to be modified as returned by <code>create_entry()</code>
<code>mind</code>	mind to be associated with the new entry; passing NULL sets the mind to none

Callable From:

- ◇ C++
- ◇ Script

5.28.14 function diguyCrowdProfile::get_mind_base_class**Prototype:**

```
const char* get_mind_base_class(int entry_index);
```

Returns:

the mind of the specified entry; NULL will be returned if there is no such entry

Arguments:

`entry_index` index of entry to be queried as returned by `create_entry()`

Callable From:

- ◇ C++
- ◇ Script

5.28.15 function diguyCrowdProfile::set_proportion**Prototype:**

```
int set_proportion(int entry_index, float proportion);
```

Description:

This function sets the proportion of the given entry. The proportion affects how often the specified entry will be chosen for `get_random_entry_index()` calls. Entries with higher proportions will be chosen more often than those with lower proportions.

The default proportion for a new entry is 1.0.

Returns:

0 on success, -1 on failure

Arguments:

`entry_index` index of entry to be modified as returned by `create_entry()`

proportion how often the new entry should be chosen relative to other entries

Callable From:

- ◇ C++
- ◇ Script

5.28.16 function diguyCrowdProfile::get_proportion

Prototype:

```
float get_proportion(int entry_index);
```

Returns:

the proportion of the specified entry; 0.0 will be returned if there is no such entry

Arguments:

entry_index index of entry to be queried as returned by create_entry()

Callable From:

- ◇ C++
- ◇ Script

5.28.17 function diguyCrowdProfile::set_default_crowd_size**Prototype:**

```
int set_default_crowd_size(int default_crowd_size);
```

Description:

This function sets the default crowd size (i.e., number of characters) of a crowd that is created using this profile. The default crowd size of a crowd profile is 20.

Returns:

0 on success, -1 on failure

Arguments:

`default_crowd_size` recommended number of characters that should be in a crowd based on this profile

Callable From:

- ◇ C++
- ◇ Script

5.28.18 function diguyCrowdProfile::get_default_crowd_size**Prototype:**

```
int get_default_crowd_size();
```

Returns:

the default crowd size of a crowd based on this profile

Callable From:

- ◇ C++
- ◇ Script

5.28.19 function diguyCrowdProfile::get_default_agent_params**Prototype:**

```
diguyAgentParams* get_default_agent_params();
```

Returns:

the default agent parameters that will be used for crowd members of crowds created using this profile; the parameters can be modified using function calls of the [diguyAgentParams](#) class

Callable From:

- ◇ C++
- ◇ Script

5.28.20 function diguyCrowdProfile::create_character**Prototype:**

```
diguyCharacter * create_character(diguyCrowd * crowd, int is_temporary = 0);
```

Returns:

a character that's been created from this profiles template and added to the crowd specified.

Callable From:

- ◇ C++
- ◇ Script

5.28.21 function diguyCrowdProfile::get_mind_init_argument**Prototype:**

```
const char* get_mind_init_argument(int entry_index, int arg);
```

Deprecated Functions

The following functions are deprecated.

As these functions may disappear in a future version of DI-Guy, it is recommended that you remove calls to these functions from your application.

5.28.22 function `diguyCrowdProfile::set_character_type_and_appearance`

Prototype:

```
int set_character_type_and_appearance(int entry_index,  
    const char* character_type,  
    const char* appearance);
```

Deprecated as of 8.6.3; pass desired `character_type` and `appearance` to `create_entry()` function instead, or call `set_character_type()` and `set_appearance()` separately.

5.28.23 function diguyCrowdProfile::add_entry**Prototype:**

```
int add_entry(const char* character_type, const char* appearance);
```

Deprecated as of 8.6.3; call [create_entry\(\)](#) instead.

5.29 diguyExerciseInterface

diguyExerciseInterface.h

DI-Guy SDK Version 12.5.2

Includes: [diguyExerciseInterface.h](#) | [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Exercise Control Functions](#)
- ◆ [AI Messaging Functions](#)
- ◆ [Experimental Conversion Functions](#)

Link against: **libdiguy**

5.29.1 class diguyExerciseInterface

```
class BDI_DECLSPEC_diguy diguyExerciseInterface
{
public:
    void send_pdu_xc_set_tin_tod(unsigned short site,
                                unsigned short host,
                                float ttod);
    void send_pdu_xc_load_scenario(unsigned short site,
                                   unsigned short host,
                                   const char* filename,
                                   const char* refname);
    void send_pdu_xc_save_scenario(unsigned short site,
                                   unsigned short host,
                                   const char* filename,
                                   const int use_user_name);
    void send_pdu_xc_reset_scenario(unsigned short site,
                                    unsigned short host);
    void send_pdu_xc_merge_scenario(unsigned short site,
                                    unsigned short host,
                                    float ttod,
                                    float refx, float refy, float refz,
                                    float tx, float ty, float tz,
                                    float rz, float rx, float ry,
                                    const char* filename,
                                    const char* suffix);
    void send_pdu_xc_trigger_script(unsigned short site,
                                    unsigned short host,
                                    const char* script_name);
    void send_pdu_xc_control(unsigned short site,
                             unsigned short host,
                             const char* command);
    void send_pdu_xc_time_now(float tod, float smooth_period);
    void send_pdu_xc_eval_script(unsigned short site,
                                 unsigned short host,
                                 const char* text);
    void send_pdu_xc_enter_slave_mode(unsigned short site, unsigned short host);
    void send_pdu_xc_exit_slave_mode(unsigned short site, unsigned short host);
```

```

int send_message_to_character(diguyCharacter* character,
    const char* sender_name,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);
int send_message_to_group(const char* group_name,
    const char* sender_name,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);
int broadcast_message(diguyCharacter* sender,
    float radius,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);
int broadcast_message_to_group(diguyCharacter* sender,
    const char* group_name,
    float radius,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);
const char* get_net_id(diguyCharacter* character);
int convert_mgrs_to_local(const char* MGRS_str, float* x, float* y, float* z);
const char* convert_local_to_mgrs(float x, float y, float z);
float convert_local_to_lat(float x, float y, float z);
float convert_local_to_long(float x, float y, float z);
const char* convert_local_to_lat_long_string(float x, float y, float z);
int convert_lat_long_to_local(float lat, float lon, float* x, float* y, float* z);

```

Exercise Control Functions

5.29.2 function**diguyExerciseInterface::send_pdu_xc_set_tin_tod****Prototype:**

```
void send_pdu_xc_set_tin_tod(unsigned short site,
                             unsigned short host,
                             float ttod);
```

Description:

This function sends an xc_set_tin_tod PDU. This PDU tells a remote DI-Guy application the simulated time of day at which its scenario is supposed to begin. Doing this provides a conceptual link between scenario time, which nominally starts at zero, and wall-clock time. This makes it possible for a scenario to be run under different simulated lighting conditions.

The remote DI-Guy application essentially does a `diguyScenario::set_tin_time_of_day( )` call.

Arguments:

<code>site</code>	destination host id (65535 for all sites)
<code>host</code>	destination host id (65535 for all hosts)
<code>ttod</code>	tin time of day (seconds after midnight)

Callable From:

- ◇ C++
- ◇ Script

5.29.3 function**diguyExerciseInterface::send_pdu_xc_load_scenario****Prototype:**

```
void send_pdu_xc_load_scenario(unsigned short site,
                               unsigned short host,
                               const char* filename,
                               const char* refname);
```

Description:

This function sends an xc_load_scenario PDU. This PDU tells a remote DI-Guy application to load a particular scenario file.

The remote DI-Guy application essentially does a [diguyScenario::load\(\)](#) call.

Arguments:

site	destination host id (65535 for all sites)
host	destination host id (65535 for all hosts)
filename	scenario to load
refname	scenario reference name

Callable From:

- ◇ C++
- ◇ Script

5.29.4 function**diguyExerciseInterface::send_pdu_xc_save_scenario****Prototype:**

```
void send_pdu_xc_save_scenario(unsigned short site,
                               unsigned short host,
                               const char* filename,
                               const int use_user_name);
```

Description:

This function sends an xc_save_scenario PDU. This PDU tells a remote DI-Guy application to save a particular scenario file.

The remote DI-Guy application essentially does a `diguyScenario::save_as()` call.

Arguments:

site	destination host id (65535 for all sites)
host	destination host id (65535 for all hosts)
filename	scenario to load
use_user_name	user name to uniquely identify the file

Callable From:

- ◇ C++
- ◇ Script

5.29.5 function

diguyExerciseInterface::send_pdu_xc_reset_scenario

Prototype:

```
void send_pdu_xc_reset_scenario(unsigned short site,
                                unsigned short host);
```

Description:

This function sends an xc_reset_scenario PDU. This PDU tells a remote DI-Guy application to reset the current scenario file.

The remote DI-Guy application essentially does a [diguyScenario::reset\(\)](#) call.

Arguments:

site	destination host id (65535 for all sites)
host	destination host id (65535 for all hosts)

Callable From:

- ◇ C++
- ◇ Script

5.29.6 function**diguyExerciseInterface::send_pdu_xc_merge_scenario****Prototype:**

```
void send_pdu_xc_merge_scenario(unsigned short site,
                                unsigned short host,
                                float ttod,
                                float refx, float refy, float refz,
                                float tx, float ty, float tz,
                                float rz, float rx, float ry,
                                const char* filename,
                                const char* suffix);
```

Description:

This function sends an xc_merge_scenario PDU. This PDU tells a remote DI-Guy application to merge a particular scenario file.

The remote DI-Guy application essentially does a [diguyScenario::merge\(\)](#) call.

Arguments:

site	destination host id (65535 for all sites)
host	destination host id (65535 for all hosts)
ttod	tin time of day for the merged scenario (seconds after midnight)
refx, refy, refz	transformation reference point in merged scenario (point around which translation and rotation will occur)
tx, ty, tz	translation of reference point in merged scenario
rz, rx, ry	rotation of reference point in merged scenario
filename	filename of scenario to merge
collision_suffix	suffix to be used in event of name conflicts

Callable From:

- ◇ C++
- ◇ Script

5.29.7 function

`diguyExerciseInterface::send_pdu_xc_trigger_script`

Prototype:

```
void send_pdu_xc_trigger_script(unsigned short site,
                                unsigned short host,
                                const char* script_name);
```

Description:

This function sends an `xc_trigger_script` PDU. This PDU tells a remote DI-Guy application to trigger the named script.

The remote DI-Guy application essentially does a `diguyScenario::trigger_script()` call.

Arguments:

<code>site</code>	destination host id (65535 for all sites)
<code>host</code>	destination host id (65535 for all hosts)
<code>script_name</code>	name of the script to be triggered

Callable From:

- ◇ C++
- ◇ Script

5.29.8 function diguyExerciseInterface::send_pdu_xc_control

Prototype:

```
void send_pdu_xc_control(unsigned short site,
                        unsigned short host,
                        const char* command);
```

Description:

This function sends an xc_control PDU. This PDU tells a remote DI-Guy application to PLAY, STOP, RESET, etc.

Arguments:

site	destination host id (65535 for all sites)
host	destination host id (65535 for all hosts)
command	command string. Valid values are "RESET", "RUN", "STOP", "FF", "REWIND"

Callable From:

- ◇ C++
- ◇ Script

5.29.9 function diguyExerciseInterface::send_pdu_xc_time_now

Prototype:

```
void send_pdu_xc_time_now(float tod, float smooth_period);
```

Description:

This function sends an xc_time_now PDU. This PDU tells all remote DI-Guy applications what the simulated time of day is right now. There is no way to target this message at particular DI-Guy hosts. It is received and processed by all DI-Guy applications in the DIS exercise.

Arguments:

tod	time of day: number of seconds since midnight, where $0.0 \leq \text{tod} < 86400.0$
smooth_period	how many real seconds should be taken to get into sync

Scenarios can proceed from one day to the next, up to a length of 24 hours. When midnight comes, tod should be specified AS ZERO, and should increase from there. Values less than zero or greater than or equal to 86400.0

will NOT be sent over the network. In the case nothing will be sent.

Callable From:

- ◇ C++
- ◇ Script

5.29.10 function diguyExerciseInterface::send_pdu_xc_eval_script

Prototype:

```
void send_pdu_xc_eval_script(unsigned short site,
                             unsigned short host,
                             const char* text);
```

Description:

This function sends an xc_eval_script PDU. This PDU tells a remote DI-Guy application to evaluate a literal piece of script text.

Arguments:

site	destination host id (65535 for all sites)
host	destination host id (65535 for all hosts)
text	text of the script to be executed

Callable From:

◇ C++
◇ Script

5.29.11 function diguyExerciseInterface::send_pdu_xc_enter_slave_mode

Prototype:

```
void send_pdu_xc_enter_slave_mode(unsigned short site, unsigned short host);
```

Description:

This function sends an xc_enter_slave PDU. This PDU tells a remote DI-Guy application to start slaving it's camera to incoming camera/time packets.

Arguments:

site	destination host id (65535 for all sites)
host	destination host id (65535 for all hosts)

Callable From:

◇ C++
◇ Script

5.29.12 function

diguyExerciseInterface::send_pdu_xc_exit_slave_mode

Prototype:

```
void send_pdu_xc_exit_slave_mode(unsigned short site, unsigned short host);
```

Description:

This function sends an xc_exit_slave PDU. This PDU tells a remote DI-Guy application to stop slaving it's camera to incoming camera/time packets.

Arguments:

site	destination host id (65535 for all sites)
host	destination host id (65535 for all hosts)

Callable From:

- ◇ C++
- ◇ Script

AI Messaging Functions

5.29.13 function

diguyExerciseInterface::send_message_to_character

Prototype:

```
int send_message_to_character(diguyCharacter* character,
    const char* sender_name,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);
```

Description:

This function sends a custom AI message PDU. This PDU is mapped into a [diguyCharacter::agent_accept_message\(\)](#) function call on the receiving side.

Arguments:

<code>character</code>	pointer to the remote diguyCharacter
<code>sender_name</code>	who is sending the message
<code>message_type</code>	type of message is being sent
<code>message</code>	message string
<code>message_params</code>	optional parameter string

Use [diguyExerciseInterface::get_net_id\(\)](#) to get the name of the character that the remote machine will see the sender as.

Callable From:

- ◇ C++
- ◇ Script

5.29.14 function

`diguyExerciseInterface::send_message_to_group`

Prototype:

```
int send_message_to_group(const char* group_name,
    const char* sender_name,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);
```

Description:

This function sends a custom AI message PDU. This PDU is mapped into a `diguyCharacterGroup::send_message_to_all_members()` on all clients.

Arguments:

<code>group_name</code>	name of the <code>diguyCharacterGroup</code> to target
<code>sender_name</code>	who is sending the message
<code>message_type</code>	what type of message is being sent
<code>message</code>	the message string
<code>message_params</code>	optional parameter string

Use `diguyExerciseInterface::get_net_id()` to get the name of the character that the remote machine will see the sender as.

Callable From:

- ◇ C++
- ◇ Script

5.29.15 function diguyExerciseInterface::broadcast_message

Prototype:

```
int broadcast_message(diguyCharacter* sender,
    float radius,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);
```

Description:

This function sends a custom AI message PDU. This PDU is mapped into a `diguyCharacter::agent_broadcast_message( )` on all clients.

Arguments:

sender	pointer to the <code>diguyCharacter</code> that is broadcasting the message
radius	how far to broadcast the message
message_type	what type of message is being sent
message	the message string
message_params	optional parameter string

Callable From:

- ◇ C++
- ◇ Script

5.29.16 function**diguyExerciseInterface::broadcast_message_to_group****Prototype:**

```
int broadcast_message_to_group(diguyCharacter* sender,
    const char* group_name,
    float radius,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);
```

Description:

This function sends a custom AI message PDU. This PDU is mapped into a [diguyCharacter::agent_broadcast_message_to_group\(\)](#) on all clients.

Arguments:

sender	pointer to the diguyCharacter that is broadcasting the message
group_name	the group that receive the message
radius	how far to broadcast the message
message_type	what type of message is being sent
message	the message string
message_params	optional parameter string

Callable From:

- ◇ C++
- ◇ Script

5.29.17 function diguyExerciseInterface::get_net_id**Prototype:**

```
const char* get_net_id(diguyCharacter* character);
```

Returns:

"sitenum_hostnum_idnum" as a string

Useful for knowing the name that this entity will have on remote machines.
This string should be copied.

Experimental Conversion Functions

5.29.18 function diguyExerciseInterface::convert_mgrs_to_local**Prototype:**

```
int convert_mgrs_to_local(const char* MGRS_str, float* x, float* y, float* z);
```

5.29.19 function diguyExerciseInterface::convert_local_to_mgrs**Prototype:**

```
const char* convert_local_to_mgrs(float x, float y, float z);
```


5.29.20 function diguyExerciseInterface::convert_local_to_lat**Prototype:**

```
float convert_local_to_lat(float x, float y, float z);
```

5.29.21 function diguyExerciseInterface::convert_local_to_long**Prototype:**

```
float convert_local_to_long(float x, float y, float z);
```

5.29.22 function**diguyExerciseInterface::convert_local_to_lat_long_string****Prototype:**

```
const char* convert_local_to_lat_long_string(float x, float y, float z);
```

5.29.23 function**diguyExerciseInterface::convert_lat_long_to_local****Prototype:**

```
int convert_lat_long_to_local(float lat, float lon, float* x, float* y, float* z);
```

5.30 diguyFormation

diguyFormation.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

Link against: libdiguy

5.30.1 class diguyFormation

```
class BDI_DECLSPEC_diguy diguyFormation
{
public:
    const char* get_name();
    int set_name(const char* name);
    const char* get_type_name();
    const char* get_guide_name();
    int set_num_followers(int number);
    int get_num_followers();
    int set_follower_rz_offset(int follower_index, float rz);
    const char* to_lua_string(float relative_x = 0.0f,
        float relative_y = 0.0f);
    float get_follower_x_offset(int follower_index);
    float get_follower_y_offset(int follower_index);
    float get_follower_rz_offset(int follower_index);
    int set_role_name(int follower_index, const char* role_name);
    const char* get_role_name(int follower_index);
    const char* get_follower_instance(int follower_index);
    int set_subgroup_size(int size);
    int get_subgroup_size();
    int get_num_subgroups();
    int set_first_subgroup_index(int index);
    int get_first_subgroup_index();
    int set_follower_offset(int follower_index, float x, float y);
    float get_radius();
```

5.30.2 function diguyFormation::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the object

Callable From:

- ◇ C++
- ◇ Script

5.30.3 function diguyFormation::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.30.4 function diguyFormation::get_type_name

Prototype:

```
const char* get_type_name();
```

Returns:

type name of the object; this value will never be NULL

Callable From:

- ◇ C++
- ◇ Script

5.30.5 function diguyFormation::get_guide_name

Prototype:

```
const char* get_guide_name();
```

Returns:

guide name used by this formation

Callable From:

- ◇ C++
- ◇ Script

5.30.6 function diguyFormation::set_num_followers

Prototype:

```
int set_num_followers(int number);
```

Description:

Sets number of followers in the formation.

Arguments:

number	number of followers in this formation

Returns:

0 if success

Callable From:

- ◇ C++
- ◇ Script

5.30.7 function diguyFormation::get_num_followers

Prototype:

```
int get_num_followers();
```

Returns:

number of members of formation (leader not included)

Callable From:

- ◇ C++
- ◇ Script

5.30.8 function diguyFormation::set_follower_rz_offset

Prototype:

```
int set_follower_rz_offset(int follower_index, float rz);
```

Description:

Sets rz offset of member relative to leader.

Arguments:

<code>follower_index</code>	index of follower to modify
<code>rz</code>	orientation of member relative to leader, in degrees

Returns:

0 if success

Callable From:

- ◇ C++
- ◇ Script

5.30.9 function diguyFormation::to_lua_string

Prototype:

```
const char* to_lua_string(float relative_x = 0.0f,  
float relative_y = 0.0f);
```

Description:

Creates a string describing the formation, with positions being relative to `relative_x` and `relative_y`.

Arguments:

<code>relative_x</code>	relative positions in meters
<code>relative_y</code>	

Returns:

string containing lua script

Callable From:

◇ C++
◇ Script

5.30.10 function diguyFormation::get_follower_x_offset

Prototype:

```
float get_follower_x_offset(int follower_index);
```

Description:

Gets x offset of formation member relative to leader.

Arguments:

follower_index index of follower in formation (does not include leader)

Returns:

x offset in meters

Callable From:

◇ C++

◇ Script

5.30.11 function diguyFormation::get_follower_y_offset

Prototype:

```
float get_follower_y_offset(int follower_index);
```

Description:

Gets y offset of formation member relative to leader.

Arguments:

follower_index index of follower in formation (does not include leader)

Returns:

y offset in meters

Callable From:

◇ C++

◇ Script

5.30.12 function diguyFormation::get_follower_rz_offset

Prototype:

```
float get_follower_rz_offset(int follower_index);
```

Description:

Gets rz (or yaw) of formation member relative to leader.

Arguments:

`follower_index` index of follower in formation (does not include leader)

Returns:

rz offset in degrees

Callable From:

◇ C++

◇ Script

5.30.13 function diguyFormation::set_role_name**Prototype:**

```
int set_role_name(int follower_index, const char* role_name);
```

Description:

Sets role name of member.

Arguments:

<code>follower_index</code>	index of follower to modify
<code>role_name</code>	new role

Returns:

0 if success

Callable From:

◇ C++

◇ Script

5.30.14 function diguyFormation::get_role_name**Prototype:**

```
const char* get_role_name(int follower_index);
```

Description:

Gets role. Roles generally correlate with a character's carried object.

Arguments:

follower_index index of follower in formation (does not include leader)

Returns:

string containing role, or "character" if specific role specified

Callable From:

◇ C++

◇ Script

5.30.15 function diguyFormation::get_follower_instance

Prototype:

```
const char* get_follower_instance(int follower_index);
```

Arguments:

`follower_index` index of follower in formation (does not include leader)

Returns:

string containing name of follower, if defined

Callable From:

◇ C++
◇ Script

5.30.16 function diguyFormation::set_subgroup_size**Prototype:**

```
int set_subgroup_size(int size);
```

Description:

Sets subgroup size. If > 0, this means that there are subgroups and that they will be applied.

Arguments:

size	size of subgroup

Returns:

0 if success

Callable From:

- ◇ C++
- ◇ Script

5.30.17 function diguyFormation::get_subgroup_size**Prototype:**

```
int get_subgroup_size();
```

Description:

Gets size of subgroup. If > 0, this means that there are subgroups and that they will be applied.

Returns:

number of followers in subgroup

Callable From:

- ◇ C++
- ◇ Script

5.30.18 function diguyFormation::get_num_subgroups**Prototype:**

```
int get_num_subgroups();
```

Description:

Get number of subgroups

Returns:

number of subgroups

Callable From:

- ◇ C++
- ◇ Script

5.30.19 function diguyFormation::set_first_subgroup_index**Prototype:**

```
int set_first_subgroup_index(int index);
```

Description:

Sets index of first subgroup. Any members before this member are not in a subgroup.

Arguments:

index	index of first follower in subgroup

Returns:

0 if success

Callable From:

- ◇ C++

◇ Script

5.30.20 function diguyFormation::get_first_subgroup_index

Prototype:

```
int get_first_subgroup_index();
```

Description:

Gets index of first formation member that's in a subgroup.

Returns:

index of formation member

Callable From:

◇ C++
◇ Script

5.30.21 function diguyFormation::set_follower_offset**Prototype:**

```
int set_follower_offset(int follower_index, float x, float y);
```

Description:

Sets x, y offsets of member relative to leader.

Arguments:

<code>follower_index</code>	index of follower to modify
<code>x, y</code>	offsets of member relative to leader, in meters

Returns:

0 if success

Callable From:

- ◇ C++
- ◇ Script

5.30.22 function diguyFormation::get_radius**Prototype:**

```
float get_radius();
```

Returns:

radius of formation, in meters

Callable From:

- ◇ C++
- ◇ Script

5.31 DI-Guy Graphics API Programming Overview

`diguyGraphicsAPIOverview.h`

DI-Guy SDK Version 12.5.2

Contents:

- ◆ Overview
- ◆ Types Overview
 - Links
 - Shapes
 - Files
 - Meshes
 - Graphics States
 - Materials
 - Textures
 - Shader Program
 - Shader Technique
 - Shader Instance
- ◆ Usage Overview
 - ◇ Immediate Mode Vs Scene Graph
 - Immediate Mode Renderers
 - Scene Graph Renderers
 - ◇ DI-Guy Loader Vs External Loader
 - DI-Guy Loader
 - External Loader
 - ◇ Virtual Functions and Stages
 - The Build Stage
 - The Update Stage
 - The Draw Stage
 - The Unbuild Stage
 - Accessors
 - ◇ Object Creation and Destruction
 - ◇ Accessing the DI-Guy Graphics API from the DI-Guy API
 - Initializing the DI-Guy Graphics API
 - Accessing DI-Guy Graphics API Objects
 - Accessing DI-Guy API Objects
- ◆ Programming Examples
 - ◇ Immediate Mode, DI-Guy Loader via OpenGL
 - ◇ Scene Graph, DI-Guy Loader via Open Scene Graph
 - ◇ Immediate Mode, External Loader via OpenGL
 - ◇ Scene Graph, External Loader via Open Scene Graph

This file provides an overview of the various programming examples available for the DI-Guy Graphics API. The actual code for the examples is in various subdirectories of `$DIGUYprogramming_examplesdiguy_graphics_api`.

Overview

In normal use, DI-Guy does all of its own rendering. DI-Guy ships with high-performance rendering libraries for immediate mode and scene graph rendering environments, including those tailored for OpenGL, DirectX9, and Vega Prime. It is quick and cost-effective for most users to get DI-Guy up and running in their applications using these standard options; most users get DI-Guy characters working in their simulators in a day using these libraries.

However, some users need more control over the rendering process or access to low-level rendering data to more tightly integrate DI-Guy into their system. The DI-Guy Graphics API can be used to:

- ◆ support proprietary rendering environments
- ◆ integrate DI-Guy tightly with image generators
- ◆ render DI-Guy characters with special lighting and rendering effects
- ◆ implement advanced sensor views, such as IR and radar views
- ◆ perform collision detection on DI-Guy characters
- ◆ facilitate external geometry file loaders
- ◆ etc.

The DI-Guy Graphics API provides a variety of functions to support these needs and to allow DI-Guy to be used in almost any rendering environment. The DI-Guy Graphics API supports both scene graph renderers (e.g. – Open Scene Graph, Performer and Vega Prime) and immediate mode renderers (e.g. – OpenGL).

DI-Guy Graphics API programming should be done by experienced C++ programmers comfortable with object-oriented design and familiar with their particular renderer. If you fit this description, you will find the DI-Guy Graphics API a well-architected programming solution for customizing DI-Guy graphics to your renderer.

Types Overview

The basic concept for using the DI-Guy Graphics API is quite simple. A set of classes, defined below, is available to the user. The user can make subclasses of these, and thus tailor DI-Guy rendering to the target rendering system.

Links

DI-Guy characters are composed of a hierarchy of links. These links, also known as bones, form the skeleton of the character. Links have no visible geometry of their own, but serve as attachment points for shapes. Links are not shared among characters; each character has its own collection of links.

Links are implemented with the `diguyGraphicsLink` object.

Shapes

Shapes are visible parts that are attached to links to "flesh out" the skeleton; with no shapes the link would not be visible. As links move, the shapes attached to them move. Shapes are not shared among characters; each character has its own collection of shapes.

Shapes are implemented with the `diguyGraphicsShape` object.

Files

Shapes are read from geometry files. DI-Guy most often uses the OpenFlight (.flt) format for static geometry, and the Collada (.dae) format for skinned geometry.

Geometry files are implemented with the `diguyGraphicsFile` object.

Meshes

Each shape is composed of one or more polygonal meshes. Each mesh consists of polygonal faces that have vertices, normals, and texture indices, as well as a graphics state that includes material, texture, and lighting settings. All faces in a mesh share the same graphics state. Meshes are shared among characters; if two characters with the same appearance are loaded, only one copy of each mesh is loaded.

Meshes are implemented with the `diguyGraphicsMesh` object.

Graphics States

Graphics states encode all information about how the polygons in a mesh should be drawn. This includes material information, a texture, and other lighting information. Graphics states are shared among characters; if two characters with the same appearance are loaded, only one copy of each state is defined.

Graphics states are implemented with the `diguyGraphicsState` object.

Materials

Materials specify the colors and lighting used to display the mesh. Materials include the ambient, diffuse, and specular components that are commonly used in renderer lighting models. Materials are shared among characters; if two characters with the same appearance are loaded, only one copy of each material is loaded.

Materials are implemented with the `diguyGraphicsMaterial` object.

Textures

Textures are 2D images that are painted onto mesh faces. Textures combine with materials to determine the appearance of the mesh. Textures are shared among characters; if two characters with the same appearance are loaded, only one copy of each texture is loaded.

Textures are implemented with the `diguyGraphicsTexture` object.

Shader Program

Most rendering environments use some type of shader system that offloads rendering operations from the CPU to a GPU. Terminology varies, but DI-Guy uses the term "shader" to refer to a pair of sub-programs: a vertex shader and a pixel shader (or fragment shader in OpenGL).

Shader programs are shared among characters; if two characters use the same shader, the same shader object is used for both.

Shader programs are implemented with the `diguyGraphicsShaderProgram` object

Shader Technique

A shader technique describes the various shaders that can be used to render a character. These include a number of different shaders that represent different quality levels and depending on the camera's distance from a character a different one can be picked. A technique can be subclassed and an end user can add additional shaders to it and implement their own LOD system or switch in a different shader for alternative lighting situations

Shader Technique are implemented with the `diguyGraphicsShaderTechnique` object.

Shader Instance

As mentioned above shader programs are shared among characters. Shader instances allow non-shared, character-specific shader state to be maintained. An example is having different uniform variable values for different characters, such as a highlight color that is applied during shading.

Shader instances are implemented with the `diguyGraphicsShaderInstance` object.

Usage Overview

Immediate Mode Vs Scene Graph

The DI-Guy Graphics API classifies renderers into two types: immediate mode and scene graph. The type of renderer has a large impact on which DI-Guy Graphics API classes are subclassed, and which virtual functions of those subclasses the user overrides.

Immediate Mode Renderers

Immediate mode renderers expect the user's code to perform transformations, bind materials and textures, and draw geometry at the appropriate times during the rendering of each frame. A primary example of an immediate mode renderer is OpenGL.

The most important stages in implementing DI-Guy support for an immediate mode renderer are the Build Stage and the Draw Stage. During the Build Stage renderer-specific objects may be created from the meshes, materials, and textures to facilitate later draws. During each Draw Stage calls are made to bind materials and textures, and transform and draw the objects.

The following classes are usually subclassed for immediate mode renderers:

- ◆ `diguyGraphicsLink`
- ◆ `diguyGraphicsShape`
- ◆ `diguyGraphicsMesh`
- ◆ `diguyGraphicsState`
- ◆ `diguyGraphicsMaterial`
- ◆ `diguyGraphicsTexture`
- ◆ `diguyGraphicsShaderProgram`
- ◆ `diguyGraphicsShaderInstance`
- ◆ `diguyGraphicsShaderTechnique`

Scene Graph Renderers

Scene graph renderers typically build a hierarchy of graphical objects. The scene graph renderer automatically takes care of transformations, binding materials and textures, and drawing geometry. Examples of scene graph renderers are Open Scene Graph and Vega Prime.

The most important stages in implementing DI-Guy support for a scene graph renderer are Build Stage and Update Stage. During the Build Stage all geometry, materials, and textures are turned into renderer-specific objects and inserted in the scene graph. During the Update Stage the scene graph objects are updated with new information based on DI-Guy motion data.

The following classes are usually subclassed for scene graph renderers:

- ◆ [diguyGraphicsLink](#)
- ◆ [diguyGraphicsShape](#)
- ◆ [diguyGraphicsFile](#)
- ◆ [diguyGraphicsShaderProgram](#)
- ◆ [diguyGraphicsShaderInstance](#)

DI-Guy Loader Vs External Loader

The DI-Guy Graphics API can work with the built-in DI-Guy Loader for loading and parsing geometry files or it can work with a user-supplied External Loader. The choice of using the DI-Guy Loader or an External Loader also has a large impact on which DI-Guy Graphics API classes and virtual functions the user will subclass.

Note: Use of external loaders is being deprecated as of DI-Guy version 11. The format of data used for skinned appearances is much more restricted than that for static appearances. The loading of Collada .dae files using an External Loader often does not provide the data in a format that works for DI-Guy. **Use of an External Loader for new development is not recommended!**

DI-Guy Loader

DI-Guy comes with a built-in geometry file loader that is used for some graphics environments (e.g., OpenGL, DirectX, and DI-Guy Graphics API), and makes use of native file loaders in graphics environments that provide them (e.g. Vega Prime). The DI-Guy Loader can load and parse various types of files, most notably OpenFlight (.flt) and Collada (.dae) files for object geometry, and a selection of various texture file types. The DI-Guy Loader correctly loads the human character geometry files provided in a DI-Guy Data distribution.

External Loader

Note: See above about use of external loaders being deprecated as of DI-Guy version 11.

Users can optionally make use of their own geometry file loader. This External Loader must be able to load and parse geometry and texture files and pick parts out of the resulting data for use as DI-Guy Graphics API shapes.

For best compatibility with DI-Guy motion data it is recommended that the External Loader be able to load the OpenFlight files in a DI-Guy Data distribution, but this is not a requirement. Users are free to load and display geometry from other proprietary geometry file formats, or even notify other processes or computers that geometry data should be loaded.

Virtual Functions and Stages

Unlike the main DI-Guy API, which makes extensive use of callback functions to notify the user's application of events, the DI-Guy Graphics API makes use of virtual functions in C++ classes. Each class has a set of virtual functions that can be overridden in a user-defined subclass. At appropriate times during program execution these virtual functions will be called

by the DI-Guy Graphics API.

The virtual functions are called at specific stages of program execution. These stages are the Build Stage, the Update Stage, the Draw Stage, and the Unbuild Stage.

The Build Stage

Build Stage functions are typically called during program initialization, when geometry files are being read and the information therein being turned into renderer-specific objects. Build functions may also be called during program main loop execution if new characters or appearances that haven't been preloaded are called upon.

The Build Stage is very important for both scene graph and immediate mode renderers.

The Update Stage

Update Stage functions are called during program main loop execution, when the state of objects should be modified based on motion data from DI-Guy. Most often this means updating degree-of-freedom (DOF) type objects in a scene graph with new joint angles, but may also include showing or hiding objects.

The Update Stage is very important for scene graph renderers.

The Draw Stage

Draw Stage functions are also called during program main loop execution, when it is time for objects to be rendered. Though they do not contain any visible geometry themselves, link objects "draw" themselves by applying updated transformation state to the renderer. Similarly, materials and textures have bind functions that are called during the Draw Stage.

The Draw Stage is very important for immediate mode renderers.

The Unbuild Stage

Unbuild Stage functions are typically called during program shutdown. They are responsible to freeing any renderer-specific resources that may have been allocated during the Build Stage.

The Unbuild Stage is very important for both scene graph and immediate mode renderers.

Accessors

During the function calls of the various stages listed above, the user obtains information by calling "accessor" functions.

A [diguyGraphicsLink](#) subclass object, for example, can determine which link is its parent (the link it is attached to in the link hierarchy), and which links are its children (the links that are attached to it) by calling the functions `get_parent_link( )` and `get_child_link_at_index( )`, respectively.

A `diguyGraphicsTexture` subclass object can call the function `get_texture_map_data()`, which returns a buffer containing the RGB data for the texture which can then be used to define a renderer-specific texture object.

Object Creation and Destruction

The DI-Guy Graphics API is responsible for creating and destroying all objects; the new and delete operators should not be called on DI-Graphics API classes or subclasses. In fact, the constructors and destructors of the classes are all protected, so compilers should refuse to compile code that breaks this rule.

In order to create subclass objects, object creation functions can be registered with the DI-Guy Graphics API. These create functions will be called whenever the API needs to allocate a new object, allowing the user to allocate and return a subclass of the object specialized for a specific renderer. The create functions should be registered before the initialization function discussed below is called.

Note that renderer-specific objects typically should not be created in the subclass constructor, but should instead be created in the Build Stage. Objects are typically created before their data has been fully read by the loader, with the consequence that the accessor functions cannot yet return valid data.

Accessing the DI-Guy Graphics API from the DI-Guy API

Initializing the DI-Guy Graphics API

To make use of the DI-Guy Graphics API for rendering instead of one of DI-Guy's built-in renderers, a different initialization function must be called. Instead of calling `diguy_ogl_initialize()` to make use of the built-in OpenGL renderer, for example, a program using the DI-Guy Graphics API would call `diguy_graphics_initialize()`. Similarly, instead of including the header file `diguy_graphics_ogl.h`, the DI-Guy Graphics API program would include `diguy_graphics_api.h`.

Accessing DI-Guy Graphics API Objects

The base link of a DI-Guy character's graphics hierarchy can be retrieved using the function `diguyCharacter::get_base_link()`. Once the base link has been retrieved, function calls to the base link can retrieve pointers to child links and shapes attached to links.

Accessing DI-Guy API Objects

A pointer to the character to which a link or shape belongs can be retrieved using the functions `diguyGraphicsLink::get_character()` and `diguyGraphicsShape::get_character()`, respectively.

All other DI-Guy Graphics API types are shared among multiple characters, so there is no single character to which they belong and therefore no `get_character()` function.

Programming Examples

Immediate Mode, DI-Guy Loader via OpenGL

This example shows the use of the DI-Guy Graphics API for immediate mode rendering using the DI-Guy Loader. It uses OpenGL as the renderer.

The following examples are provided:

- ◆ Playback of .dss scenario files in [simple_playback_ogl.cpp](#)

The following subclasses are implemented in this example:

- ◆ diguyOglGraphicsLink in [diguyOglGraphicsLink.cpp](#)
- ◆ diguyOglGraphicsShape in [diguyOglGraphicsShape.cpp](#)
- ◆ diguyOglGraphicsMesh in [diguyOglGraphicsMesh.cpp](#)
- ◆ diguyOglGraphicsMaterial in [diguyOglGraphicsMaterial.cpp](#)
- ◆ diguyOglGraphicsTexture in [diguyOglGraphicsTexture.cpp](#)
- ◆ diguyOglGraphicsShaderProgram in [diguyOglGraphicsShaderProgram.cpp](#)
- ◆ diguyOglGraphicsShaderInstance in [diguyOglGraphicsShaderInstance.cpp](#)

Scene Graph, DI-Guy Loader via Open Scene Graph

This example shows the use of the DI-Guy Graphics API in a scene graph renderer using the DI-Guy Loader. It uses Open Scene Graph as the renderer.

The following examples are provided:

- ◆ Playback of .dss scenario files in [simple_playback_osg_2.cpp](#)

The following subclasses are implemented in the examples:

- ◆ diguyOsgGraphicsLink in [diguyOsgGraphicsLink.cpp](#)
- ◆ diguyOsgGraphicsShape in [diguyOsgGraphicsShape.cpp](#)
- ◆ diguyOsgGraphicsMesh in [diguyOsgGraphicsMesh.cpp](#)
- ◆ diguyOsgGraphicsMaterial in [diguyOsgGraphicsMaterial.cpp](#)
- ◆ diguyOsgGraphicsTexture in [diguyOsgGraphicsTexture.cpp](#)
- ◆ diguyOsgGraphicsState in [diguyOsgGraphicsState.cpp](#)
- ◆ diguyOsgGraphicsShaderInstance in [diguyOsgGraphicsShaderInstance.cpp](#)
- ◆ diguyOsgGraphicsShaderProgram in [diguyOsgGraphicsShaderProgram.cpp](#)

Immediate Mode, External Loader via OpenGL

This example shows the use of the DI-Guy Graphics API in an immediate mode renderer using an External Loader. It uses OpenGL as the renderer.

Note that skinned appearances are not supported in this example.

The following examples are provided:

- ◆ Playback of .dss scenario files in [simple_playback_ogl_ext.cpp](#)

The following subclasses are implemented in the examples:

- ◆ `diguyOglGraphicsLink` in [diguyOglGraphicsLink.cpp](#)
- ◆ `diguyOglGraphicsShape` in [diguyOglGraphicsShape.cpp](#)
- ◆ `diguyOglGraphicsFile` in [diguyOglGraphicsFile.cpp](#)

Scene Graph, External Loader via Open Scene Graph

This example shows the use of the DI-Guy Graphics API in a scene graph renderer using an external loader. It uses Open Scene Graph as the renderer.

Note: This example exists only as a code example! It currently does not run, as there is no external loader available in OSG that can load compressed DI-Guy geometry files.

The following examples are provided:

- ◆ Playback of .dss scenario files in [simple_playback_osg_2_ext.cpp](#)

The following subclasses are implemented in this example:

- ◆ `diguyOsgExtGraphicsLink` in [diguyOsgExtGraphicsLink.cpp](#)
- ◆ `diguyOsgExtGraphicsShape` in [diguyOsgExtGraphicsShape.cpp](#)
- ◆ `diguyOsgExtGraphicsFile` in [diguyOsgExtGraphicsFile.cpp](#)

5.32 diguyGraphicsFile

diguyGraphicsFile.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Accessor Functions](#)
- ◆ [Virtual Functions](#)

Link against: **libdiguy**

5.32.1 class diguyGraphicsFile

```
class BDI_DECLSPEC_diguy diguyGraphicsFile
{
public:
    const char* get_filename();
#ifdef CPLUSPLUS_ONLY
    virtual int load(const char* filename, const char* extension);
    virtual int unload();
    virtual void* find_part(const char* part_name);
#endif
};
```

Accessor Functions

The functions in this section provide access to data in this object. They are designed to be called directly by DI-Guy programmers when needed.

5.32.2 function diguyGraphicsFile::get_filename

Prototype:

```
const char* get_filename();
```

Description:

Returns the filename of the graphics file. This is the filename that was passed to `load( )`. The filename may be NULL if `load( )` has not been called.

Returns:

filename of the graphics file

Callable From:

- ◇ C++
- ◇ Script

Virtual Functions

The functions in this section are all virtual functions that will be called by DI-Guy at various times during DI-Guy execution.

Because the functions are all virtual, DI-Guy programmers can create a subclass of this class and override them. Unless otherwise stated, the base class functions do not need to be called.

5.32.3 function diguyGraphicsFile::load

Prototype:

```
virtual int load(const char* filename, const char* extension);
```

Description:

This function is called when the contents of a geometry file need to be loaded.

DI-Guy Loader:

Implementations using the DI-Guy Loader usually do not override this function.

External Loader:

Implementations using an External Loader must override this function.

Arguments:

filename the name of the geometry file that needs to be loaded
extension the extension (usually 3 characters) of the file; can help quickly choose an appropriate external loader

The filename argument includes the full path to the file to be loaded.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during the Build Stage)

5.32.4 function diguyGraphicsFile::unload

Prototype:

```
virtual int unload();
```

Description:

This function does anything necessary to unload assets created during the [load\(\)](#) call. It will be called after all meshes, states, etc., that were extracted from the file have been unloaded.

DI-Guy Loader:

Implementations using the DI-Guy Loader usually do not override this function.

External Loader:

Implementations using an External Loader must override this function.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during the Unbuild Stage)

5.32.5 function diguyGraphicsFile::find_part

Prototype:

```
virtual void* find_part(const char* part_name);
```

Description:

This function needs to find and return an object representing a shape in the data loaded during the `load( )` call. Typically this is a group-type node above the meshes containing the geometry of the shape.

The returned object is not used directly by DI-Guy. Instead, it will be used in later calls to `diguyGraphicsShape::get_diguy_graphics_file_part( )`, during the Build Stage.

Arguments:

`part_name` the name of the part that needs to be found

DI-Guy Loader:

Implementations using the DI-Guy Loader usually do not override this function.

External Loader:

Implementations using an External Loader must override this function.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during the Build Stage)

5.33 diguyGraphicsLink

diguyGraphicsLink.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Accessor Functions](#)
- ◆ [Matrix Retrieval Functions](#)
- ◆ [Virtual Functions](#)
- ◆ [Experimental Functions](#)
- ◆ [Experimental Functions](#)
- ◆ [Deprecated Functions](#)

Link against: **libdiguy**

5.33.1 class diguyGraphicsLink

```
class BDI_DECLSPEC_diguy diguyGraphicsLink
{
public:
    const char* get_name();
    long get_uid();
    int get_index();
    diguyCharacter* get_character();
    int get_num_shapes();
    diguyGraphicsShape* get_shape_at_index(int index);
    diguyGraphicsShape* find_shape(const char* name);
    diguyGraphicsShape* find_shape_by_class_type(const char* name);
    diguyGraphicsJointType get_joint_type();
    int get_offset_translation(float* tx, float* ty, float* tz);
    int get_offset_rotation(float* rz, float* rx, float* ry);
    int get_translation(float* tx, float* ty, float* tz);
    int get_translation_double(double* tx, double* ty, double* tz);
    int get_translation_is_static();
    const char* get_var_name_tx();
    const char* get_var_name_ty();
    const char* get_var_name_tz();
    int get_rotation(float* rz, float* rx, float* ry);
    int get_rotation_is_static();
    const char* get_var_name_rz();
    const char* get_var_name_rx();
    const char* get_var_name_ry();
    int get_scale(float* sz, float* sx, float* sy);
    int get_scale_is_static();
    const char* get_var_name_sx();
    const char* get_var_name_sy();
    const char* get_var_name_sz();
    diguyGraphicsLink* get_parent_link();
    diguyGraphicsLink* get_attached_to_link();
    int get_is_position_link();
```

```

    int get_num_child_links();
    diguyGraphicsLink* get_child_link_at_index(int index);
#ifdef CPLUSPLUS_ONLY
    int get_transformation_matrix(float matrix_array[4][4], int transpose = 0);
    int get_transformation_matrix_4x4_ptr(float* matrix, int transpose = 0);
    int get_local_transformation_matrix(float matrix_array[4][4], int transpose = 0);
    int get_local_transformation_matrix_4x4_ptr(float* matrix, int transpose = 0);
    int get_link_transformation_matrix(float matrix_array[4][4], int transpose = 0);
    int get_link_transformation_matrix_4x4_ptr(float* matrix, int transpose = 0);
#endif
#ifdef CPLUSPLUS_ONLY
    const float* get_data_ptr();
    const float* get_quat_data_ptr();
#endif
    int get_translation_data_index();
    int get_rotation_data_index();
    int get_quat_data_index();
    int get_scale_data_index();
    int get_unique_color(unsigned char* r,
        unsigned char* g,
        unsigned char* b);
#ifdef CPLUSPLUS_ONLY
    virtual void build();
    virtual void post_build();
    virtual void unbuild();
    virtual void attach_to_scene();
    virtual void detach_from_scene();
    virtual void attach_to_link(diguyGraphicsLink* link);
    virtual void detach_from_link();
    virtual void update();
    virtual void push();
    virtual void pop();
    virtual void draw();
    virtual void set_shader_instance(diguyGraphicsShaderInstance* shader_instance);
    virtual void begin_character_draw();
    virtual void end_character_draw();
#endif
    static void set_use_linear_traversal_draw(int mode);
    static int get_use_linear_traversal_draw();
    int get_inverse_rotation_matrix(float matrix_array[4][4], int transpose = 0,
        int include_translation = 1);
    int get_inverse_rotation_matrix_4x4_ptr(float* matrix_array, int transpose = 0,
        int include_translation = 1);
    int get_is_base_link();
    int get_shader_matrix_index();

```

Accessor Functions

The functions in this section provide access to data in this object. They are designed to be called by DI-Guy programmers.

Unless otherwise noted all functions in this class section are callable from:

- ◆ C++
- ◆ Script

5.33.2 function diguyGraphicsLink::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the link. This pointer will never be NULL.

Returns:

name of the link

5.33.3 function diguyGraphicsLink::get_uid

Prototype:

```
long get_uid();
```

Description:

All links are assigned a unique identifier, or uid. This function returns this link's uid.

Note: unique identifiers will change between DI-Guy runs!

Returns:

unique identifier of object

5.33.4 function diguyGraphicsLink::get_index

Prototype:

```
int get_index();
```

Returns:

index of the link

5.33.5 function diguyGraphicsLink::get_character

Prototype:

```
diguyCharacter* get_character();
```

Description:

Returns a pointer to the [diguyCharacter](#) to which this link belongs.

Returns:

pointer of type [diguyCharacter](#)

5.33.6 function diguyGraphicsLink::get_num_shapes

Prototype:

```
int get_num_shapes();
```

Returns:

number of shapes on this link

5.33.7 function diguyGraphicsLink::get_shape_at_index

Prototype:

```
diguyGraphicsShape* get_shape_at_index(int index);
```

Returns:

pointer of type [diguyGraphicsShape](#); NULL if no shape at the specified index

Arguments:

index	index of the shape; indices start at 0

5.33.8 function diguyGraphicsLink::find_shape

Prototype:

```
diguyGraphicsShape* find_shape(const char* name);
```

Description:

This function returns a pointer to the specified shape.

Arguments:

name	name of link to be found

Returns:

pointer of type [diguyGraphicsShape](#); NULL if not found

5.33.9 function diguyGraphicsLink::find_shape_by_class_type**Prototype:**

```
diguyGraphicsShape* find_shape_by_class_type(const char* name);
```

Description:

This function returns a pointer to the specified shape.

Arguments:

name	name of shape with class_type to be found, currently works with heads

Returns:

pointer of type [diguyGraphicsShape](#); NULL if not found

5.33.10 function diguyGraphicsLink::get_joint_type**Prototype:**

```
diguyGraphicsJointType get_joint_type();
```

Description:

Returns the type of joint that attaches this link to its parent.

Returns:

joint type of the link

5.33.11 function diguyGraphicsLink::get_offset_translation

Prototype:

```
int get_offset_translation(float* tx, float* ty, float* tz);
```

Description:

Returns in the passed variables the translation offset of this link from its parent.

The offset will always remain fixed once DI-Guy is initialized.

Returned values are in meters.

Arguments:

tx, ty, tz pointers to floats into which result should be placed

Returns:

0 on success, -1 on failure

5.33.12 function diguyGraphicsLink::get_offset_rotation

Prototype:

```
int get_offset_rotation(float* rz, float* rx, float* ry);
```

Description:

Returns in the passed variables the rotational offset of this link from its parent.

The offset will always remain fixed once DI-Guy is initialized.

Returned values are an Euler triple in degrees, and should be applied in RZ -> RX -> RY order, after the translation offset.

Arguments:

rz, rx, ry pointers to floats into which result should be placed

Returns:

0 on success, -1 on failure

5.33.13 function diguyGraphicsLink::get_translation

Prototype:

```
int get_translation(float* tx, float* ty, float* tz);
```

Description:

Returns in the passed variables the translation amount of the link's joint.

If there is a current render camera set, the position link's values will be in the local space of the camera. This is useful for rendering far from the origin.

The values may or may not stay constant frame to frame; see [get_translation_is_static\(\)](#).

Returned values are in meters, and should be applied after the rotational offset (if any).

Arguments:

tx, ty, pointers to floats into which result should be placed

tz

Returns:

0 on success, -1 on failure

5.33.14 function diguyGraphicsLink::get_translation_double**Prototype:**

```
int get_translation_double(double* tx, double* ty, double* tz);
```

Description:

Returns in the passed variables the translation amount of the link's joint.

The values may or may not stay constant frame to frame; see [get_translation_is_static\(\)](#).

Returned values are in meters, and should be applied after the rotational offset (if any).

Arguments:

tx, ty,	pointers to doubles into which result should be placed
tz	

Returns:

0 on success, -1 on failure

5.33.15 function diguyGraphicsLink::get_translation_is_static

Prototype:

```
int get_translation_is_static();
```

Description:

Returns whether the translation returned by `get_translation()` is static (unchanging) or dynamic (can change frame to frame).

If the translation is dynamic the translation values will come from one or more DI-Guy motion variables; the names of the variables are accessible using the functions `get_var_name_tx()`, etc.

If the translation is static it will remain so; the value returned by this function should not change frame to frame.

Returns:

1 if translation is static, 0 if not, -1 on failure

5.33.16 function diguyGraphicsLink::get_var_name_tx**Prototype:**

```
const char* get_var_name_tx();
```

Description:

Returns the name of the DI-Guy motion data variable that controls translation along the X axis.

The returned pointer will be NULL if there is no associated variable.

Returns:

name of the variable; NULL if there is no variable

5.33.17 function diguyGraphicsLink::get_var_name_ty**Prototype:**

```
const char* get_var_name_ty();
```

Description:

Returns the name of the DI-Guy motion data variable that controls translation along the Y axis.

The returned pointer will be NULL if there is no associated variable.

Returns:

name of the variable; NULL if there is no variable

5.33.18 function diguyGraphicsLink::get_var_name_tz**Prototype:**

```
const char* get_var_name_tz();
```

Description:

Returns the name of the DI-Guy motion data variable that controls translation along the Z axis.

The returned pointer will be NULL if there is no associated variable.

Returns:

name of the variable; NULL if there is no variable

5.33.19 function diguyGraphicsLink::get_rotation**Prototype:**

```
int get_rotation(float* rz, float* rx, float* ry);
```

Description:

Returns in the passed variables the rotation amount of the link's joint.

The values may or may not stay constant frame to frame; see [get_rotation_is_static\(\)](#).

Returned values are an Euler triple in degrees, and should be applied in RZ -> RX -> RY order, after the translation.

Arguments:

<code>rz, rx,</code>	pointers to floats into which result should be placed
<code>ry</code>	

Returns:

0 on success, -1 on failure

5.33.20 function diguyGraphicsLink::get_rotation_is_static

Prototype:

```
int get_rotation_is_static();
```

Description:

Returns whether the rotation returned by `get_rotation()` is static (unchanging) or dynamic (can change frame to frame).

If the rotation is dynamic the rotation values will come from one or more DI-Guy motion variables; the names of the variables are accessible using the functions `get_var_name_rz()`, etc.

If the rotation is static it will remain so; the value returned by this function should not change frame to frame.

Returns:

1 if rotation is static, 0 if not, -1 on failure

5.33.21 function diguyGraphicsLink::get_var_name_rz**Prototype:**

```
const char* get_var_name_rz();
```

Description:

Returns the name of the DI-Guy motion data variable that controls rotation about the Z axis.

The returned pointer will be NULL if there is no associated variable.

5.33.22 function diguyGraphicsLink::get_var_name_rx**Prototype:**

```
const char* get_var_name_rx();
```

Description:

Returns the name of the DI-Guy motion data variable that controls rotation about the X axis.

The returned pointer will be NULL if there is no associated variable.

5.33.23 function diguyGraphicsLink::get_var_name_ry**Prototype:**

```
const char* get_var_name_ry();
```

Description:

Returns the name of the DI-Guy motion data variable that controls rotation about the Y axis.

The returned pointer will be NULL if there is no associated variable.

5.33.24 function diguyGraphicsLink::get_scale**Prototype:**

```
int get_scale(float* sz, float* sx, float* sy);
```

Description:

Returns in the passed variables the scale amount of the link's joint.

The values may or may not stay constant frame to frame; see [get_scale_is_static\(\)](#).

Returned values unitless scale factors, and should be applied after the rotation.

Arguments:

SX, SY,	pointers to floats into which result should be placed
SZ	

Returns:

0 on success, -1 on failure

5.33.25 function diguyGraphicsLink::get_scale_is_static

Prototype:

```
int get_scale_is_static();
```

Description:

Returns whether the scale returned by [get_scale\(\)](#) is static (unchanging) or dynamic (can change frame to frame).

If the scale is dynamic the scale values will come from one or more DI-Guy motion variables; the names of the variables are accessible using the functions [get_var_name_sx\(\)](#), etc.

If the scale is static it will remain so; the value returned by this function should not change frame to frame.

Returns:

1 if scale is static, 0 if not, -1 on failure

5.33.26 function diguyGraphicsLink::get_var_name_sx**Prototype:**

```
const char* get_var_name_sx();
```

Description:

Returns the name of the DI-Guy motion data variable that controls scale in the Z dimension.

The returned pointer will be NULL if there is no associated variable.

5.33.27 function diguyGraphicsLink::get_var_name_sy**Prototype:**

```
const char* get_var_name_sy();
```

Description:

Returns the name of the DI-Guy motion data variable that controls scale in the Y dimension.

The returned pointer will be NULL if there is no associated variable.

5.33.28 function diguyGraphicsLink::get_var_name_sz**Prototype:**

```
const char* get_var_name_sz();
```

Description:

Returns the name of the DI-Guy motion data variable that controls scale in the X dimension.

The returned pointer will be NULL if there is no associated variable.

5.33.29 function diguyGraphicsLink::get_parent_link**Prototype:**

```
diguyGraphicsLink* get_parent_link();
```

Description:

Returns a pointer to this link's parent link. Parent links are always links within the same character. If a link is attached to a link in another character (i.e., it is "parented to" another link), that link pointer can be obtained by calling [get_attached_to_link\(\)](#).

The returned pointer will be NULL if the link is the position link and has no parent.

Returns:

object of type [diguyGraphicsLink](#)

5.33.30 function diguyGraphicsLink::get_attached_to_link**Prototype:**

```
diguyGraphicsLink* get_attached_to_link();
```

Description:

Returns a pointer to this link's attached to link. The attached to link is different than the parent link in that a parent link is always a link in the same character, while an attached to link is always a link in a different character.

Only position links can be attached to links of another character.

The returned pointer will be NULL if the link is not attached to the link of another character.

Returns:

object of type [diguyGraphicsLink](#)

5.33.31 function diguyGraphicsLink::get_is_position_link**Prototype:**

```
int get_is_position_link();
```

Description:

Returns 1 if this link is the position link, or 0 if not. The position link is the root link of the DI-Guy character's link and shape hierarchy, and has no parent link.

Returns:

1 if this link is the position link, 0 if not, -1 on error

5.33.32 function diguyGraphicsLink::get_num_child_links**Prototype:**

```
int get_num_child_links();
```

Description:

Returns how many child links this link has. Child links can be accessed using the [get_child_link_at_index\(\)](#) function.

Returns:

number of child links

5.33.33 function diguyGraphicsLink::get_child_link_at_index**Prototype:**

```
diguyGraphicsLink* get_child_link_at_index(int index);
```

Description:

Returns a pointer to this link's child link at the specified index.

The returned pointer will be NULL if there is no link at that index.

Arguments:

index	index of the child link; indices start at 0

Returns:

object of type [diguyGraphicsLink](#)

Matrix Retrieval Functions

All functions in this class section are restricted to C++.

5.33.34 function diguyGraphicsLink::get_transformation_matrix

Prototype:

```
int get_transformation_matrix(float matrix_array[4][4], int transpose = 0);
```

Description:

This function fills out the passed float array with the world-coordinate system transformation matrix of this link.

NOTE:

If the character's performance level (CPL) is 3 or greater, link matrices are not computed or updated, and this call should not be used. See [diguyCharacter::set_minimum_cpl\(\)](#) for more information.

Arguments:

matrix_array two-dimensional array of floats into which the matrix should be copied

transpose

defaults to zero, sets if the matrix should be treated as row-major or column-major

Returns:

0 on success, -1 on failure

5.33.35 function `diguyGraphicsLink::get_transformation_matrix_4x4_ptr`

Prototype:

```
int get_transformation_matrix_4x4_ptr(float* matrix, int transpose = 0);
```

Description:

Pointer version of `get_transformation_matrix()`, see for docs.

5.33.36 function `diguyGraphicsLink::get_local_transformation_matrix`

Prototype:

```
int get_local_transformation_matrix(float matrix_array[4][4], int transpose = 0);
```

Description:

This function fills out the passed float array with the character-local-coordinate system transformation matrix of this link.

NOTE:

If the character's performance level (CPL) is 3 or greater, link matrices are not computed or updated, and this call should not be used. See [`diguyCharacter::set\_minimum\_cpl\(\)`](#) for more information.

Arguments:

<code>matrix_array</code>	two-dimensional array of floats into which the matrix should be copied
<code>transpose</code>	defaults to 0, sets if the matrix should be treated as row-major or column-major

Returns:

0 on success, -1 on failure

5.33.37 function**diguyGraphicsLink::get_local_transformation_matrix_4x4_ptr****Prototype:**

```
int get_local_transformation_matrix_4x4_ptr(float* matrix, int transpose = 0);
```

Description:

Pointer version of [get_local_transformation_matrix\(\)](#), see for docs.

5.33.38 function**diguyGraphicsLink::get_link_transformation_matrix****Prototype:**

```
int get_link_transformation_matrix(float matrix_array[4][4], int transpose = 0);
```

Description:

This function fills out the passed float array with the link-local-coordinate system transformation matrix of this link. This is the same value as the matrix built from [get_rotation\(\)](#) and [get_translation\(\)](#)

NOTE:

If the character's performance level (CPL) is 3 or greater, link matrices are not computed or updated, and this call should not be used. See [diguyCharacter::set_minimum_cpl\(\)](#) for more information.

Arguments:

matrix_array

two-dimensional array of floats into which the matrix should be copied

transpose defaults to 0, sets if the matrix should be treated as row-major or column-major

Returns:

0 on success, -1 on failure

5.33.39 function diguyGraphicsLink::get_link_transformation_matrix_4x4_ptr

Prototype:

```
int get_link_transformation_matrix_4x4_ptr(float* matrix, int transpose = 0);
```

Description:

Pointer version of [get_link_transformation_matrix\(\)](#), see for docs.

5.33.40 function diguyGraphicsLink::get_data_ptr**Prototype:**

```
const float* get_data_ptr();
```

Description:

Returns a pointer to the current motion data array. Along with data indices returned below, motion data can be directly read from this state vector instead of making multiple calls for getting transform data.

The rotations in this array will be in Euler angles. If quaternions are being used, [get_quat_data_ptr\(\)](#) should be called instead.

This function may return NULL, in which case the functions [get_translation\(\)](#), [get_rotation\(\)](#), and [get_scale\(\)](#) must be used.

The pointer returned by this function may change from frame to frame, and therefore should not be saved.

This function should not be used for the position link (when [get_is_position_link\(\)](#) returns 1).

This function should not be called in a derived class's constructor; the pointer will not yet be valid.

Returns:

pointer to floats; may return NULL if array isn't available

Callable From:

◇ C++

5.33.41 function diguyGraphicsLink::get_quat_data_ptr**Prototype:**

```
const float* get_quat_data_ptr();
```

Description:

This function is similar to [get_data_ptr\(\)](#), except rotations in the returned array will be in quaternions instead of Euler angles.

Returns:

pointer to floats; may return NULL if array isn't available

Callable From:

◇ C++

5.33.42 function diguyGraphicsLink::get_translation_data_index**Prototype:**

```
int get_translation_data_index();
```

Description:

Returns the index of this link's translation data in the float array returned by [get_data_ptr\(\)](#) or [get_quat_data_ptr\(\)](#). The translation values are in (tx, ty, tz) order in the array.

This function may return -1, in which case this link has no translation data.

The index returned by this function will not change from frame to frame, and can therefore be saved once it has been obtained.

This function should not be used for the position link (when [get_is_position_link\(\)](#) returns 1). Use [get_translation\(\)](#) instead.

This function should not be called in a derived class's constructor; the index will not yet be valid.

Returns:

index into float array; -1 if no translation data for link

5.33.43 function diguyGraphicsLink::get_rotation_data_index

Prototype:

```
int get_rotation_data_index();
```

Description:

Returns the index of this link's rotation data in the float array returned by [get_data_ptr\(\)](#). If quaternion data is being used, call [get_quat_data_ptr\(\)](#) instead.

The rotation values in the data array will be Euler angles in (rz, rx, ry) order.

This function may return -1, in which case this link has no rotation data.

This function should not be used for the position link (when [get_is_position_link\(\)](#) returns 1). Use [get_rotation\(\)](#) instead.

This function should not be called in a derived class's constructor; the index will not yet be valid.

Returns:

index into float array; -1 if no rotation data for link

5.33.44 function diguyGraphicsLink::get_quat_data_index

Prototype:

```
int get_quat_data_index();
```

Description:

Returns the index of this link's rotation data in the float array returned by [get_quat_data_ptr\(\)](#). If Euler angle data is being used, call [get_data_ptr\(\)](#) instead.

The rotation values in the data array will be a quaternion in (qx, qy, qz, qw) order.

This function may return -1, in which case this link has no rotation data.

The index returned by this function will not change from frame to frame, and can therefore be saved once it has been obtained.

This function should not be used for the position link (when [get_is_position_link\(\)](#) returns 1). Use [get_rotation\(\)](#) instead.

This function should not be called in a derived class's constructor; the index will not yet be valid.

Returns:

index into float array; -1 if no rotation data for link

5.33.45 function diguyGraphicsLink::get_scale_data_index

Prototype:

```
int get_scale_data_index();
```

Description:

Returns the index of this link's scale data in the float array returned by [get_data_ptr\(\)](#) or [get_quat_data_ptr\(\)](#). The scale values are in (sx, sy, sz) order in the array.

This function may return -1, in which case this link has no scale data.

The index returned by this function will not change from frame to frame, and can therefore be saved once it has been obtained.

This function should not be used for the position link (when [get_is_position_link\(\)](#) returns 1). Use [get_scale\(\)](#) instead.

This function should not be called in a derived class's constructor; the index will not yet be valid.

Returns:

index into float array; -1 if no scale data for link

5.33.46 function `diguyGraphicsLink::get_unique_color`

Prototype:

```
int get_unique_color(unsigned char* r,
                    unsigned char* g,
                    unsigned char* b);
```

Description:

Returns in the passed pointers the unique color of this link. All links in a DI-Guy appearance are assigned a unique RGB color that can be used in pick/selection operations.

Also see `diguyScenario::map_color_to_impact()`.

See
[programming_examples/diguy_graphics_apiogl_examples/intersection_test](#)
 for an example of its usage.

Returns:

0 on success, -1 on failure

@TODO Test with lua probably non-functional

Virtual Functions

The functions in this section are all virtual functions that will be called by DI-Guy at various times during DI-Guy execution.

Because the functions are all virtual, DI-Guy programmers can create a subclass of this class and override them. Unless otherwise stated, the base class functions do not need to be called.

5.33.47 function diguyGraphicsLink::build

Prototype:

```
virtual void build();
```

Description:

This function will be called by DI-Guy when it is time for renderer-specific link objects to be created. All information necessary to build the link should be available via the Accessor Functions above when this function is called.

Builds happen in depth-first traversal order, so the parent's `build()` function will always be called before this link's `build()` function.

Immediate Mode:

Immediate mode renderers usually do not override this function.

Scene Graph:

Scene graph renderers usually do override this function, to create scene graph nodes that allow for the various transformations to happen.

The following nodes are commonly created:

- ◊ a static transform node to hold the offsets
- ◊ a dynamic transform node (or DOF) to allow motion data to be applied
- ◊ either an LOD or switch node to allow graphics LODs to be applied

LOD nodes, that automatically handle LOD switching, should be created if automatic LOD switching is desired.

A switch node should be created if manual setting of the graphics LOD is desired.

If this link is a position link (it has no parent link), the scene graph nodes should not yet be attached to the scene graph; that should be done during the call to `attach()`.

Otherwise the scene graph nodes should start out attached to their attach points (a scene graph node of the parent link).

Callable From:

- ◊ N/A (automatically called by DI-Guy Graphics API during the Build Stage)

5.33.48 function diguyGraphicsLink::post_build**Prototype:**

```
virtual void post_build();
```

Description:

This function will be called by DI-Guy when all links and shapes of a DI-Guy character have been built, providing an opportunity for final touch-ups to be made to the built hierarchy.

This function will be called only for the top-level position link.

Immediate Mode:

Immediate mode renderers usually do not override this function.

Scene Graph:

Scene graph renderers sometimes override this function. They might do so to do renderer-specific optimizations to the created hierarchy.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during the Build Stage)

5.33.49 function diguyGraphicsLink::unbuild

Prototype:

```
virtual void unbuild();
```

Description:

This function will be called by DI-Guy when it is time for the renderer-specific link objects created during the `build()` call to be destroyed.

Unbuilds happen in reverse depth-first traversal order, so this link's `unbuild()` function will always be called before the parent's `unbuild()` function.

Immediate Mode:

Immediate mode renderers usually do not override this function.

Scene Graph:

Scene graph renderers usually do override this function, to destroy the scene graph nodes created by `build()`.

Callable From:

◊ N/A (automatically called by DI-Guy Graphics API during the Unbuild Stage)

5.33.50 function diguyGraphicsLink::attach_to_scene**Prototype:**

```
virtual void attach_to_scene();
```

Description:

This function will be called by DI-Guy when it is time for renderer-specific link objects created during the `build()` call to be attached to their attach points.

This function will only be called for the position link.

Immediate Mode:

Immediate mode renderers usually do not override this function.

Scene Graph:

Scene graph renderers usually do override this function, to attach the scene graph nodes created during `build()` to somewhere on the scene graph.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during the Build Stage, and sometimes during the Update Stage)

5.33.51 function diguyGraphicsLink::detach_from_scene

Prototype:

```
virtual void detach_from_scene();
```

Description:

This function will be called by DI-Guy when it is time for renderer-specific link objects created during the `build()` call to be detached from their attach points.

Immediate Mode:

Immediate mode renderers usually do not override this function.

Scene Graph:

Scene graph renderers usually do override this function, to detach the scene graph nodes created during `build()` from the scene graph.

Callable From:

◊ N/A (automatically called by DI-Guy Graphics API during the Build Stage, and sometimes during the Update Stage)

5.33.52 function diguyGraphicsLink::attach_to_link

Prototype:

```
virtual void attach_to_link(diguyGraphicsLink* link);
```

Description:

This function will be called by DI-Guy when it is time for renderer-specific link objects created during the `build()` call to be attached to a link of another character.

This function will only be called for the position link.

Immediate Mode:

Immediate mode renderers usually do not override this function.

Scene Graph:

Scene graph renderers usually do override this function, to attach the scene graph nodes created during `build()` to somewhere on the scene graph.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during the Build Stage, and sometimes during the Update Stage)

5.33.53 function diguyGraphicsLink::detach_from_link

Prototype:

```
virtual void detach_from_link();
```

Description:

This function will be called by DI-Guy when it is time for renderer-specific link objects created during the `build()` call to be detached from a link of another character.

This function will only be called for the position link.

Immediate Mode:

Immediate mode renderers usually do not override this function.

Scene Graph:

Scene graph renderers usually do override this function, to detach the scene graph nodes created during `build()` from somewhere on the scene graph.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during the Build Stage, and sometimes during the Update Stage)

5.33.54 function diguyGraphicsLink::update

Prototype:

```
virtual void update();
```

Description:

This function will be called by DI-Guy when it is time for renderer-specific link objects created during the `build()` call to be updated with new transformation data.

Updates happen in depth-first traversal order, so the parent's `update()` function will always be called before this link's `update()` function.

Immediate Mode:

Immediate mode renderers usually do not override this function.

Scene Graph:

Scene graph renderers usually do override this function, to update the scene graph nodes created during `build()` with new values.

These values should be obtained by calling `get_translation()`, `get_rotation()`, and `get_scale()`.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during Update Stage)

5.33.55 function `diguyGraphicsLink::push`

Prototype:

```
virtual void push();
```

Description:

This function will be called by DI-Guy when it is traversing the link hierarchy to draw.

Immediate Mode:

Immediate mode renderers can override this function. Typically some type of transformation matrix stack push operation is performed.

Scene Graph:

Scene graph renderers usually do not override this function; transformation states are handled automatically by scene graphs.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during Draw Stage)

5.33.56 function diguyGraphicsLink::pop

Prototype:

```
virtual void pop();
```

Description:

This function will be called by DI-Guy when it is traversing the link hierarchy to draw.

Immediate Mode:

Immediate mode renderers can override this function. Typically some type of transformation matrix stack pop operation is performed.

Scene Graph:

Scene graph renderers usually do not override this function; transformation states are handled automatically by scene graphs.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during Draw Stage)

5.33.57 function diguyGraphicsLink::draw

Prototype:

```
virtual void draw();
```

Description:

This function will be called by DI-Guy when it is time for the transformations of this link to be applied.

Immediate Mode:

Immediate mode renderers usually do override this function, to apply transformations to the transformation stack.

Scene Graph:

Scene graph renderers usually do not override this function; transformation states are handled automatically by scene graphs.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during Draw Stage)

5.33.58 function diguyGraphicsLink::set_shader_instance

Prototype:

```
virtual void set_shader_instance(diguyGraphicsShaderInstance* shader_instance);
```

Description:

This function will be called by DI-Guy when it a shader should be bound to this link. This is important in some scene graph environments, when the scene graph is responsible for binding the shader.

Immediate Mode:

Immediate mode renderers usually do not override this function.

Scene Graph:

Scene graph renderers can override this function but some don't need to.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during Build Stage)

Experimental Functions

5.33.59 function diguyGraphicsLink::begin_character_draw**Prototype:**

```
virtual void begin_character_draw();
```

Description:

This function will be called by DI-Guy when the drawing of a new character is beginning. This happens when the position link is drawn.

Immediate Mode:

Immediate mode renderers can override this function. Typically some type of transformation matrix stack push operation is performed. This is also an appropriate place to do any per-character modifications, such as:

- ◊ applying scale
- ◊ applying the base translation of the character
- ◊ picking a current LOD
- ◊ etc.

Scene Graph:

Scene graph renderers usually do not override this function; transformation states are handled automatically by scene graphs.

Callable From:

- ◊ N/A (automatically called by DI-Guy Graphics API during Draw Stage)

5.33.60 function diguyGraphicsLink::end_character_draw**Prototype:**

```
virtual void end_character_draw();
```

Description:

This function will be called by DI-Guy when the drawing traversal of the character's link hierarchy has completed.

Immediate Mode:

Immediate mode renderers can override this function. Typically any matrix stack push operation done in [begin_character_draw\(\)](#) is undone here by an appropriate pop operation.

Scene Graph:

Scene graph renderers usually do not override this function; transformation states are handled automatically by scene graphs.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during Draw Stage)

5.33.61 function

diguyGraphicsLink::set_use_linear_traversal_draw

Prototype:

```
static void set_use_linear_traversal_draw(int mode);
```

Description:

This value corresponds to `bdiGraphicsInitGraphicsAPI.use_linear_traversal_draw`. This value can be changed at runtime, typically for debugging or performance comparison purposes.

Overview of how the modes differ:

```
if (!get_use_linear_traversal_draw())
{
    // push the base transform of the character in the world
    // the get_transformation_matrix() of the position link
    m_position_link->begin_character_draw();

    // each link should multiply by the get_link_transformation_matrix concatenation
    // them as the link array is traversed
    m_position_link->push();
    m_position_link->recursive_draw();
    m_position_link->pop();
    m_position_link->end_character_draw();
}

//linear draw behavior
else
{
    // push the base transform of the character in the world
    // the get_transformation_matrix() of the position link
    m_position_link->begin_character_draw();
    int i;
    for each link in link array
    {
        // note the ability to skip links with nothing attached to them
        if (link->get_num_shapes() > 0 || link->get_is_position_link())
        {
            // each link should multiply by the get_local_transformation_matrix
            // which contains the precalculated concatenated matrix
            link->push();
            link->draw();
            link->pop();
        }

        if (link->has_attached_creatures())
        {
            for each attached creature
            {
                attached_creature->draw_as_child();
            }
        }
    }
}
```

```
m_position_link->end_character_draw();
```

5.33.62 function diguyGraphicsLink::get_use_linear_traversal_draw

Prototype:

```
static int get_use_linear_traversal_draw();
```

Description:

Returns whether recursive draw mode is enabled. See [set_use_linear_traversal_draw\(\)](#).

Experimental Functions

The following functions are experimental and may change or disappear with no warning.

5.33.63 function diguyGraphicsLink::get_inverse_rotation_matrix

Prototype:

```
int get_inverse_rotation_matrix(float matrix_array[4][4], int transpose = 0,  
    int include_translation = 1);
```

5.33.64 function**diguyGraphicsLink::get_inverse_rotation_matrix_4x4_ptr****Prototype:**

```
int get_inverse_rotation_matrix_4x4_ptr(float* matrix_array, int transpose = 0,  
int include_translation = 1);
```

Deprecated Functions

The following functions are deprecated.

As these functions may disappear in a future version of DI-Guy, it is recommended that you remove calls to these functions from your application.

5.33.65 function diguyGraphicsLink::get_is_base_link**Prototype:**

```
int get_is_base_link();
```

Description:

Deprecated as of 9.1.4; use
`diguyGraphicsLink::get_is_position_link()` instead.

5.33.66 function diguyGraphicsLink::get_shader_matrix_index**Prototype:**

```
int get_shader_matrix_index();
```

Description:

Deprecated as of 12.0.0 due to architecture change

5.34 diguyGraphicsMaterial

diguyGraphicsMaterial.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Accessor Functions](#)
- ◆ [Virtual Functions](#)

Link against: `libdiguy_graphics_api`

5.34.1 class diguyGraphicsMaterial

```
class BDI_DECLSPEC_diguy diguyGraphicsMaterial
{
public:
    const char* get_name();
#ifdef CPLUSPLUS_ONLY
    void get_ambient(float* rgba);
    void get_diffuse(float* rgba);
    void get_specular(float* rgba);
    void get_emission(float* rgba);
    void get_shininess(float* s);
    int get_contains_alpha_component();
    void free_loader_memory();
    static diguyGraphicsMaterial* get_last_bound_material();
    static void clear_last_bound_material();
#endif
#ifdef CPLUSPLUS_ONLY
    virtual void build();
    virtual void unbuild();
    virtual void bind_now();
#endif
}
```

Accessor Functions

The functions in this section provide access to data in this object. They are designed to be called by DI-Guy programmers.

5.34.2 function diguyGraphicsMaterial::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the texture. This pointer will never be NULL.

Returns:

name of the texture

Callable From:

- ◇ C++
- ◇ Script

5.34.3 function diguyGraphicsMaterial::get_ambient

Prototype:

```
void get_ambient(float* rgba);
```

Description:

This function returns the red, green, blue, and alpha components of the material's ambient setting. All returned values will be between 0 (least intensity) and 1 (highest intensity).

Arguments:

rgba	pointer to an array of four floats into which results should be copied

Callable From:

◇ C++

5.34.4 function diguyGraphicsMaterial::get_diffuse

Prototype:

```
void get_diffuse(float* rgba);
```

Description:

This function returns the red, green, blue, and alpha components of the material's diffuse setting. All returned values will be between 0 (least intensity) and 1 (highest intensity).

Arguments:

rgba	pointer to an array of four floats into which results should be copied

Callable From:

◇ C++

5.34.5 function diguyGraphicsMaterial::get_specular

Prototype:

```
void get_specular(float* rgba);
```

Description:

This function returns the red, green, blue, and alpha components of the material's specular setting. All returned values will be between 0 (least intensity) and 1 (highest intensity).

Arguments:

rgba	pointer to an array of four floats into which results should be copied

Callable From:

◇ C++

5.34.6 function diguyGraphicsMaterial::get_emission

Prototype:

```
void get_emission(float* rgba);
```

Description:

This function returns the red, green, blue, and alpha components of the material's emission setting. All returned values will be between 0 (least intensity) and 1 (highest intensity).

Arguments:

rgba	pointer to an array of four floats into which results should be copied

Callable From:

◇ C++

5.34.7 function diguyGraphicsMaterial::get_shininess

Prototype:

```
void get_shininess(float* s);
```

Description:

This function returns the material's shininess setting.

Arguments:

s pointer to float into which result should be copied

Callable From:

◇ C++

5.34.8 function diguyGraphicsMaterial::get_contains_alpha_component

Prototype:

```
int get_contains_alpha_component();
```

Description:

This function returns whether the material contains an alpha component. Materials that have an alpha component can make objects semi-transparent, which can affect when said objects should be drawn.

Returns:

1 if texture contains an alpha component, 0 if not; -1 on failure

Callable From:

◇ C++

5.34.9 function `diguyGraphicsMaterial::free_loader_memory`

Prototype:

```
void free_loader_memory();
```

Description:

This function frees most memory allocated by the DI-Guy geometry loader for this object.

This function should only be called during or after the object's `build()` function has been called. After it has been called, calls to many accessor functions will fail.

Callable From:

◇ C++

5.34.10 function `diguyGraphicsMaterial::get_last_bound_material`

Prototype:

```
static diguyGraphicsMaterial* get_last_bound_material();
```

Description:

This function returns the last material that was bound.

In `bind_now()` the current material can be checked against the last bound material; if the values are the same the `bind_now()` function can most often immediately return. This is desirable because rendering state changes can have significant performance overhead.

Callable From:

◇ C++

5.34.11 function `diguyGraphicsMaterial::clear_last_bound_material`

Prototype:

```
static void clear_last_bound_material();
```

Description:

This function clears out the last bound material pointer.

`diguyScenario::draw()` calls it automatically, but it's recommended that this function also be called when the drawing of a new character is beginning. This can be done in `diguyGraphicsLink::begin_character_draw()`.

Callable From:

◇ C++

Virtual Functions

The functions in this section are all virtual functions that will be called by DI-Guy at various times during DI-Guy execution.

Because the functions are all virtual, DI-Guy programmers can create a subclass of this class and override them. Unless otherwise stated, the base class functions do not need to be called.

5.34.12 function diguyGraphicsMaterial::build

Prototype:

```
virtual void build();
```

Description:

This function will be called by DI-Guy when it is time for a renderer-specific material object to be created. At that point all material information has been read and is available via the Accessor Functions above.

Immediate Mode:

Immediate mode renderers may override this function, to create an object that may be invoked by `bind_now( )` to set material attributes.

Scene Graph:

Scene graph renderers usually do override this function, to create a material object that will be associated with `diguyGraphicsMesh` objects.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during the Build Stage)

5.34.13 function diguyGraphicsMaterial::unbuild

Prototype:

```
virtual void unbuild();
```

Description:

This function will be called by DI-Guy when it is time for a renderer-specific material object to be destroyed.

Immediate Mode:

Immediate mode renderers may override this function, to destroy any object that may have been created by `build()`.

Scene Graph:

Scene graph renderers usually do override this function, to destroy any object that may have been created by `build()`.

Callable From:

◊ N/A (automatically called by DI-Guy Graphics API during the Unbuild Stage)

5.34.14 function diguyGraphicsMaterial::bind_now

Prototype:

```
virtual void bind_now();
```

Description:

This function will be called by DI-Guy when the material should be made active, or bound.

Note that this material should be checked against the material returned by `get_last_bound_material()`. If they're the same this function should return early.

Immediate Mode:

Immediate mode renderers usually do override this function, to invoke the object created by `build()`, or make calls to the renderer to set appropriate material state.

Scene Graph:

Scene graph renderers usually do not override this function.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during Draw Stage)

5.35 diguyGraphicsMesh

diguyGraphicsMesh.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [diguyGraphicsVertexFormats.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Accessor Functions](#)
- ◆ [Geometry Data Accessor Functions](#)
 - ◇ [Shader Accessor Functions](#)
 - ◇ [Vertex Buffer Data Accessor Functions](#)
 - ◇ [Index Buffer Data Accessor Functions](#)
- ◆ [Virtual Functions](#)
- ◆ [Deprecated Functions](#)

Link against: **libdiguy_graphics_api**

5.35.1 class diguyGraphicsMesh

```
class BDI_DECLSPEC_diguy diguyGraphicsMesh
{
public:
    const char* get_name();
    long get_uid();
    diguyGraphicsState* get_state(int graphics_state_switch_index = 0);
    diguyGraphicsState* get_current_graphics_state();
    int get_num_switched_graphics_states();
    int get_is_skinned_mesh();
    int get_is_deformable();
    void free_loader_memory();
    diguyGraphicsShaderProgram* get_shader_program();
    diguyGraphicsShaderInstance* get_shader_instance();
    diguyGraphicsVertexFormat get_vertex_format();
    int get_vertex_count();
    int get_vertex_buffer_size();
#ifdef CPLUSPLUS_ONLY
    void* get_vertex_buffer_data();
    float* get_first_vertex();
    int get_vertex_offset();
    int get_vertex_stride();
    float* get_first_normal();
    int get_normal_offset();
    int get_normal_stride();
    float* get_first_texture_indices();
    int get_texture_indices_offset();
    int get_texture_indices_stride();
    float* get_first_link_index();
    int get_link_index_offset();
    int get_link_index_stride();
    float* get_first_link_weight();
#endif
};
```



```

    int get_link_weight_offset();
    int get_link_weight_stride();
    float* get_first_tangent();
    int get_tangent_offset();
    int get_tangent_stride();
#endif
    diguyGraphicsIndexType get_index_type();
    int get_index_count();
#ifdef CPLUSPLUS_ONLY
    int get_index_buffer_size();
    void* get_index_buffer_data();
#endif
#ifdef CPLUSPLUS_ONLY
    virtual void build();
    virtual void unbuild();
    virtual void draw();
#endif
#ifdef CPLUSPLUS_ONLY
    int get_vertex_info(int* num_vertices,
        float** vertex_array,
        float** normal_array,
        float** texture_array,
        int* index_array_size,
        unsigned short** index_array);
#endif

```

Accessor Functions

The functions in this section provide access to data in this object. They are designed to be called by DI-Guy programmers.

5.35.2 function diguyGraphicsMesh::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the mesh. This pointer will never be NULL.

Returns:

name of the mesh

5.35.3 function diguyGraphicsMesh::get_uid

Prototype:

```
long get_uid();
```

Description:

All meshes are assigned a unique identifier (uid). This function returns the uid of the mesh.

Note: uids will change between DI-Guy runs.

Returns:

unique identifier of the object

5.35.4 function diguyGraphicsMesh::get_state

Prototype:

```
diguyGraphicsState* get_state(int graphics_state_switch_index = 0);
```

Description:

This function returns the graphics state with which faces in this mesh should be drawn. Specific information about material and texture can then be accessed from this object.

Some meshes may have multiple possible graphics states associated with them. In effect this means that depending on a character's appearance, different textures or material settings may be used.

Call [get_num_switched_graphics_states\(\)](#) to get the number of graphics states that are associated with this mesh.

This function is typically called during the Build stage for scene graph renderers, when geoset/mesh objects are being created and their graphics states set.

Returns:

object of type [diguyGraphicsState](#); NULL on failure

5.35.5 function diguyGraphicsMesh::get_current_graphics_state**Prototype:**

```
diguyGraphicsState* get_current_graphics_state();
```

Description:

This function returns the last bound graphic state object this mesh used. There should always be at least one. This can be used by immediate mode renderers during draw calls to find out more about the properties of the mesh.

See [get_state\(\)](#) for more information on switched graphics states.

Returns:

object of type [diguyGraphicsState](#); NULL on failure

5.35.6 function diguyGraphicsMesh::get_num_switched_graphics_states**Prototype:**

```
int get_num_switched_graphics_states();
```

Description:

This function returns the number of switched graphics states this mesh has. There should always be at least one.

See [get_state\(\)](#) for more information on switched graphics states.

Returns:

number of switched graphics states this mesh has

5.35.7 function diguyGraphicsMesh::get_is_skinned_mesh

Prototype:

```
int get_is_skinned_mesh();
```

Description:

This function returns whether or not the mesh is "skinned".

The position of vertices in a skinned mesh are determined at run-time by transforming them with multiple link transforms.

It is likely that some type of shader is required to render the mesh correctly. The shader recommended for this mesh can be obtained by calling [get_shader_program\(\)](#).

Though they sound similar, skinned meshes and deformable meshes are different. The vertices in a deformable mesh have already been transformed when the [draw\(\)](#) function is called; nothing more needs to be done. The vertices in a skinned are not ready to be drawn, as they must first be transformed by a shader.

Returns:

1 if mesh is skinned, 0 if not, -1 on failure

5.35.8 function diguyGraphicsMesh::get_is_deformable

Prototype:

```
int get_is_deformable();
```

Description:

This function returns whether or not the mesh is deformable.

The position of vertices in a deformable mesh may shift from frame to frame and therefore should not be put into a renderer-specific object that cannot be updated. (For example, vertex buffer objects in OpenGL.)

The position of vertices in a non-deformable, non-skinned mesh remain fixed from frame to frame.

Though they sound similar, skinned meshes and deformable meshes are different. The vertices in a deformable mesh have already been transformed when the `draw()` function is called; nothing more needs to be done. The vertices in a skinned mesh are not ready to be drawn, as they must first be transformed by a shader.

Returns:

1 if mesh is deformable, 0 if not, -1 on failure

5.35.9 function `diguyGraphicsMesh::free_loader_memory`

Prototype:

```
void free_loader_memory();
```

Description:

This function frees most memory allocated by the DI-Guy geometry loader for this object.

This function should only be called during or after the object's `build()` function has been called. After it has been called, calls to many accessor functions will fail.

Geometry Data Accessor Functions

The functions in this section provide access to geometry data of this mesh, including vertex positions, normals, texture indices, etc.

They are designed to be called by DI-Guy programmers during the Build Stage. Calling them before the Build Stage will return invalid results.

Functions that return pointers to data (`get_vertex_buffer_data()`, etc.) cannot be called after `free_loader_memory()` is called.

For in-depth examples of how these functions work, look at the following files in `programming_examples/diguy_graphics_api`:

- ◆ `ogl_examplescommonoglGraphicsMesh.cpp`
- ◆ `osg_2_examplescommondiguyOsgGraphicsMesh.cpp`

For more information on how the returned data is structured, refer to the documentation for the types `diguyGraphicsVertexFormat` and `DIGUY_GRAPHICS_INDEX_TYPE_UINT`.

Shader Accessor Functions

5.35.10 function `diguyGraphicsMesh::get_shader_program`

Prototype:

```
diguyGraphicsShaderProgram*  get_shader_program();
```

Description:

This function returns a pointer to the shader program designated for this mesh, if any. Shader programs are shared objects usable by many meshes.

Returns:

pointer to `diguyGraphicsShaderProgram`, or NULL if this mesh does not have a shader specified

5.35.11 function diguyGraphicsMesh::get_shader_instance

Prototype:

```
diguyGraphicsShaderInstance* get_shader_instance();
```

Description:

If this mesh has a shader program specified (see [get_shader_program\(\)](#)), this function returns a pointer to the shader instance object specific to this mesh.

Returns:

pointer to [diguyGraphicsShaderInstance](#), or NULL if this mesh does not have a shader specified

Vertex Buffer Data Accessor Functions

5.35.12 function diguyGraphicsMesh::get_vertex_format

Prototype:

```
diguyGraphicsVertexFormat get_vertex_format();
```

Returns:

vertex format that describes the format of the data returned by the [get_vertex_buffer_data\(\)](#) function

5.35.13 function diguyGraphicsMesh::get_vertex_count**Prototype:**

```
int get_vertex_count();
```

Returns:

how many vertices this mesh has

5.35.14 function diguyGraphicsMesh::get_vertex_buffer_size**Prototype:**

```
int get_vertex_buffer_size();
```

Returns:

size, in bytes, of the data buffer returned by
[get_vertex_buffer_data\(\)](#)

5.35.15 function diguyGraphicsMesh::get_vertex_buffer_data**Prototype:**

```
void* get_vertex_buffer_data();
```

Description:

This function returns a pointer to the raw vertex buffer data of this mesh. The format of the data can be obtained by calling `get_vertex_format()`.

Type-correct pointers into the data can be obtained by calling `get_first_vertex()`, `get_first_normal()`, etc.

Returns:

pointer to raw vertex buffer data

5.35.16 function diguyGraphicsMesh::get_first_vertex**Prototype:**

```
float* get_first_vertex();
```

Description:

This function returns a pointer to vertex position data of the first vertex in the raw vertex buffer data returned by `get_vertex_buffer_data()`.

The XYZ position of the first vertex is stored in the first three values of the returned pointer. The XYZ positions of subsequent vertices need to be obtained by consecutively adding the vertex stride to this returned pointer, as returned by `get_vertex_stride()`.

NOTE: `get_vertex_stride()` returns a value in **bytes**, not `sizeof(float)`, so be careful about pointer arithmetic.

This approach is also used for the first normal, texture indices, link index, and link weight of the vertex data.

An alternative approach to getting a pointer to the first vertex position is to use `get_vertex_offset()`. See that function for more information.

Returns:

pointer to vertex position of first vertex in vertex buffer data

5.35.17 function diguyGraphicsMesh::get_vertex_offset

Prototype:

```
int get_vertex_offset();
```

Description:

This function returns how many bytes the first vertex position is into the raw vertex buffer data returned by `get_vertex_buffer_data()`.

The XYZ positions of subsequent vertices need to be obtained by consecutively adding the vertex stride to this offset, as returned by `get_vertex_stride()`.

NOTE: `get_vertex_stride()` returns a value in **bytes**, not `sizeof(float)`, so be careful about pointer arithmetic.

This approach is also used for the first normal, texture indices, link index, and link weight offsets into the vertex data.

An alternative approach to getting a pointer to the first vertex position is to use `get_first_vertex()` to get a type-correct pointer.

Returns:

offset in bytes of the first vertex position in raw vertex buffer data

5.35.18 function diguyGraphicsMesh::get_vertex_stride**Prototype:**

```
int get_vertex_stride();
```

Description:

This function returns how many bytes there are between each vertex position in the raw data returned by [get_vertex_buffer_data\(\)](#).

See [get_first_vertex\(\)](#) and [get_vertex_offset\(\)](#) for more information.

NOTE: this function returns a value in **bytes**, not `sizeof(float)`, so be careful about pointer arithmetic.

Returns:

stride in bytes between each vertex position in raw vertex buffer data

5.35.19 function diguyGraphicsMesh::get_first_normal**Prototype:**

```
float* get_first_normal();
```

Returns:

pointer to normal of first vertex in vertex buffer data; see [get_first_vertex\(\)](#) for more information

5.35.20 function diguyGraphicsMesh::get_normal_offset**Prototype:**

```
int get_normal_offset();
```

Returns:

offset in bytes of the first normal in raw vertex buffer data; see [get_vertex_offset\(\)](#) for more information

5.35.21 function diguyGraphicsMesh::get_normal_stride**Prototype:**

```
int get_normal_stride();
```

Returns:

stride in bytes between each normal in raw vertex buffer data; see [get_vertex_stride\(\)](#) for more information

5.35.22 function diguyGraphicsMesh::get_first_texture_indices**Prototype:**

```
float* get_first_texture_indices();
```

Returns:

pointer to texture indices of first vertex in vertex buffer data

5.35.23 function diguyGraphicsMesh::get_texture_indices_offset**Prototype:**

```
int get_texture_indices_offset();
```

Returns:

offset in bytes of the first texture indices in raw vertex buffer data; see [get_vertex_offset\(\)](#) for more information

5.35.24 function diguyGraphicsMesh::get_texture_indices_stride**Prototype:**

```
int get_texture_indices_stride();
```

Returns:

stride in bytes between each texture indices pair in raw vertex buffer data; see [get_vertex_stride\(\)](#) for more information

5.35.25 function diguyGraphicsMesh::get_first_link_index**Prototype:**

```
float* get_first_link_index();
```

Returns:

pointer to link index of first vertex in vertex buffer data, in float form

5.35.26 function diguyGraphicsMesh::get_link_index_offset**Prototype:**

```
int get_link_index_offset();
```

Returns:

offset in bytes of the first link index in raw vertex buffer data; see [get_vertex_offset\(\)](#) for more information

5.35.27 function diguyGraphicsMesh::get_link_index_stride**Prototype:**

```
int get_link_index_stride();
```

Returns:

stride in bytes between each link index in raw vertex buffer data; see [get_vertex_stride\(\)](#) for more information

5.35.28 function diguyGraphicsMesh::get_first_link_weight**Prototype:**

```
float* get_first_link_weight();
```

Returns:

pointer to link weight of first vertex in vertex buffer data

5.35.29 function diguyGraphicsMesh::get_link_weight_offset**Prototype:**

```
int get_link_weight_offset();
```

Returns:

offset in bytes of the first link offset in raw vertex buffer data; see [get_vertex_offset\(\)](#) for more information

5.35.30 function diguyGraphicsMesh::get_link_weight_stride**Prototype:**

```
int get_link_weight_stride();
```

Returns:

stride in bytes between each link weight in raw vertex buffer data; see [get_vertex_stride\(\)](#) for more information

5.35.31 function diguyGraphicsMesh::get_first_tangent**Prototype:**

```
float* get_first_tangent();
```

Description:

Returns a pointer to the UV tangent and flip bit of first vertex in the vertex buffer data.

There are 3 floats representing the tangent, and one for UV mirroring. See <http://www.dhpoware.com/demos/gls/NormalMapping.html> for an outline of how tangents are calculated. Binormals are calculated in the vertex shader making their storage unnecessary. Tangents should exist for all skinned/dae models.

See [get_first_vertex\(\)](#) for more information about how to interpret tangent pointers and strides.

5.35.32 function diguyGraphicsMesh::get_tangent_offset**Prototype:**

```
int get_tangent_offset();
```

Returns:

offset in bytes of the first tangent in raw vertex buffer data; see [get_vertex_offset\(\)](#) for more information

5.35.33 function diguyGraphicsMesh::get_tangent_stride

Prototype:

```
int get_tangent_stride();
```

Returns:

stride in bytes between each tangent in raw vertex buffer data; see [get_vertex_stride\(\)](#) for more information

Index Buffer Data Accessor Functions

5.35.34 function diguyGraphicsMesh::get_index_type

Prototype:

```
diguyGraphicsIndexType get_index_type();
```

Returns:

index type used in data returned by the [get_index_buffer_data\(\)](#) function

5.35.35 function diguyGraphicsMesh::get_index_count**Prototype:**

```
int get_index_count();
```

Description:

This function returns how many indices there are in this mesh. The number of triangles in this mesh can be derived by dividing this value by 3.

Note that indices in index buffer data are different than the indices contained in vertex buffer data. Index buffer indices specify which **vertices** in vertex buffer data are used to create polygons in a mesh (e.g., for the first triangle, use vertices 0, 1, and 2), while indices stored in vertex buffer data specify which **links** affect the final position of the vertex.

Returns:

number of indices in index buffer data returned by
[get_index_buffer_data\(\)](#)

5.35.36 function diguyGraphicsMesh::get_index_buffer_size**Prototype:**

```
int get_index_buffer_size();
```

Returns:

size, in bytes, of the data buffer returned by
[get_index_buffer_data\(\)](#)

5.35.37 function diguyGraphicsMesh::get_index_buffer_data**Prototype:**

```
void* get_index_buffer_data();
```

Description:

This function returns a pointer to the raw index buffer data of this mesh. The format of the data can be obtained by calling [get_index_type\(\)](#).

Returns:

pointer to raw index buffer data

Virtual Functions

The functions in this section are all virtual functions that will be called by DI-Guy at various times during DI-Guy execution.

Because the functions are all virtual, DI-Guy programmers can create a subclass of this class and override them. Unless otherwise stated, the base class functions do not need to be called.

5.35.38 function diguyGraphicsMesh::build

Prototype:

```
virtual void build();
```

Description:

This function will be called by DI-Guy when it is time for a renderer-specific mesh object to be created. All information necessary to build the mesh should be available via the Accessor Functions above when this function is called.

Immediate Mode:

Immediate mode renderers may override this function, to create an object that may be invoked by `draw()`.

Scene Graph:

Scene graph renderers usually do override this function, to create a geometry object that gets associated with a `diguyGraphicsState` object obtained by calling `get_state()`.

Callable From:

◊ N/A (automatically called by DI-Guy Graphics API during the Build Stage)

5.35.39 function diguyGraphicsMesh::unbuild**Prototype:**

```
virtual void unbuild();
```

Description:

This function will be called by DI-Guy when it is time for a renderer-specific mesh object to be destroyed.

Immediate Mode:

Immediate mode renderers may override this function, to destroy any object that may have been created by `build()`.

Scene Graph:

Scene graph renderers usually do override this function, to destroy any object that may have been created by `build()`.

Callable From:

◊ N/A (automatically called by DI-Guy Graphics API during the Unbuild Stage)

5.35.40 function diguyGraphicsMesh::draw**Prototype:**

```
virtual void draw();
```

Description:

This function will be called by DI-Guy when the mesh should be drawn. For scene graph renderers, most likely nothing more needs to be done by this function.

Callable From:

◊ N/A (automatically called by DI-Guy Graphics API during Draw Stage)

Deprecated Functions

The following functions are deprecated.

As these functions may disappear in a future version of DI-Guy, it is recommended that you remove calls to these functions from your application.

5.35.41 function diguyGraphicsMesh::get_vertex_info

Prototype:

```
int get_vertex_info(int* num_vertices,  
    float** vertex_array,  
    float** normal_array,  
    float** texture_array,  
    int* index_array_size,  
    unsigned short** index_array);
```

Description:

Deprecated as of 10.0.0; use Geometry Data Accessor Functions instead. This function can't provide data for all vertex formats or index formats that DI-Guy now supports. This function assumes that vertex buffer data is in array format, and that indices in index buffer data are unsigned shorts. Data for skinned meshes can't be represented with this function.

5.36 diguyGraphicsShaderInstance

diguyGraphicsShaderInstance.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Accessor Functions](#)
- ◆ [Virtual Functions](#)
- ◆ [Shader Variables](#)
- ◆ [Deprecated Functions](#)

Link against: `libdiguy_graphics_api`

This class implements shader instances for use with shader programs for a character.

Shader **programs** are shared among all characters that use the shader program. Any variables used in a shader program should therefore apply to all characters using the shader.

Shader **instances** are not shared. Each character that has a shader program assigned to it will have a private shader instance in which variables specific to the character or its instance can be stored. A common case for this is current values of GLSL shader uniform variables.

Also see [diguyGraphicsShaderProgram](#).

5.36.1 class diguyGraphicsShaderInstance

```
class BDI_DECLSPEC_diguy diguyGraphicsShaderInstance
{
public:
    diguyGraphicsShaderTechnique* get_shader_technique();
#ifdef CPLUSPLUS_ONLY
    const float* get_shader_matrix_data_ptr();
#endif
    diguyCharacter* get_character();
#ifdef CPLUSPLUS_ONLY
    virtual int bind();
    virtual int unbind();
#endif
    int get_num_variables();
    const char* get_variable_name_at_index(int index);
    int has_variable(const char* name);
    const char* get_variable_type(const char* name);
    void reset_variables_to_initial_values();
    void reset_variables_to_program_defaults();
    int set_variable_float(const char* name, float value, int set_current = 1, int set_in = 1);
    int set_variable_vec2(const char* name, float x, float y, int set_current = 1, int set_in = 1);
    int set_variable_vec3(const char* name, float x, float y, float z, int set_current = 1, int set_in = 1);
};
```

```

int set_variable_vec4(const char* name, float x, float y, float z, float w, int set_c
int set_variable_int(const char* name, int value, int set_current = 1, int set_initia
int get_variable_float(const char* name, float* value, int get_current = 1);
int get_variable_vec2(const char* name, float* x, float* y, int get_current = 1);
int get_variable_vec3(const char* name, float* x, float* y, float* z, int get_current
int get_variable_vec4(const char* name, float* r, float* g, float* b, float* a, int g
int get_variable_int(const char* name, int* value, int get_current = 1);
diguyGraphicsShaderProgram* get_shader_program();
int get_num_link_matrices();
virtual int update_link_matrices(int num_matrices,
    const float* mat_data);

```

Accessor Functions

The functions in this section provide access to data in this object. They are designed to be called by DI-Guy programmers.

5.36.2 function diguyGraphicsShaderInstance::get_shader_technique

Prototype:

```
diguyGraphicsShaderTechnique* get_shader_technique();
```

Description:

This function returns the shader technique this shader instance is associated with.

Returns:

pointer to type [diguyGraphicsShaderTechnique](#); should never be NULL

Callable From:

◇ C++

5.36.3 function diguyGraphicsShaderInstance::get_shader_matrix_data_ptr

Prototype:

```
const float* get_shader_matrix_data_ptr();
```

Description:

This function returns a pointer to shader matrix data. This is the same pointer that is passed into `bind_link_matrices( )`.

The returned pointer should not be cached for later used, as it may change multiple times during the lifetime of this shader instance.

Returns:

pointer to first value in array of floats

Callable From:

◇ C++

5.36.4 function diguyGraphicsShaderInstance::get_character

Prototype:

```
diguyCharacter* get_character();
```

Description:

This function returns a pointer to the character that this shader instance is associated with.

Returns:

pointer to type [diguyCharacter](#); may be NULL

Callable From:

◇ C++

Virtual Functions

The functions in this section are all virtual functions that will be called by DI-Guy at various times during DI-Guy execution.

Because the functions are all virtual, DI-Guy programmers can create a subclass of this class and override them. Unless otherwise stated, the base class functions do not need to be called.

5.36.5 function diguyGraphicsShaderInstance::bind

Prototype:

```
virtual int bind();
```

Description:

This function will be called by DI-Guy when it is time for the object that this shader instance is attached to to be rendered. This happens during the Draw Stage, just before the geometry of a character or scene object is to be rendered.

Any instance level shader control variables should be set in this function. Instance level variables will affect only shader operations in which this instance is active. Variables that should affect the overall program of a shader should be set in `diguyGraphicsShaderProgram::bind()`.

Immediate Mode:

Immediate mode renderers usually do override this function.

Scene Graph:

Scene graph renderers usually do not override this function; the renderer will typically do this internally. If shader variables need to be bound, however, this function should be overridden.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during the Draw Stage)

5.36.6 function diguyGraphicsShaderInstance::unbind

Prototype:

```
virtual int unbind();
```

Description:

This function will be called by DI-Guy when the object this shader is attached to is done being rendered.

This happens during the Draw Stage, just after the geometry of a character has been rendered.

Any shader variables bound in `bind( )` should be unbound here.

Immediate Mode:

Immediate mode renderers usually do override this function.

Scene Graph:

Scene graph renderers usually do not override this function; the renderer will typically do this internally. If the `bind( )` function was overridden, however, this function probably should be, too.

Callable From:

◊ N/A (automatically called by DI-Guy Graphics API during the Draw Stage)

Shader Variables

For GLSL shaders, these variables are represented internally as uniform variables. DI-Guy reference shaders commonly use the prefix "ufrn_" for their uniform variables.

Instance shader variables are not shared between instances. Variables that should affect all instances of the program this instance is associated with should be set at the shader program level, using equivalent functions in [diguyGraphicsShaderProgram](#).

5.36.7 function `diguyGraphicsShaderInstance::get_num_variables`

Prototype:

```
int get_num_variables();
```

5.36.8 function `diguyGraphicsShaderInstance::get_variable_name_at_index`

Prototype:

```
const char* get_variable_name_at_index(int index);
```

5.36.9 function `diguyGraphicsShaderInstance::has_variable`

Prototype:

```
int has_variable(const char* name);
```

5.36.10 function `diguyGraphicsShaderInstance::get_variable_type`

Prototype:

```
const char* get_variable_type(const char* name);
```

5.36.11 function `diguyGraphicsShaderInstance::reset_variables_to_initial_values`

Prototype:

```
void reset_variables_to_initial_values();
```

5.36.12 function `diguyGraphicsShaderInstance::reset_variables_to_program_defa`

Prototype:

```
void reset_variables_to_program_defaults();
```

5.36.13 function `diguyGraphicsShaderInstance::set_variable_float`

Prototype:

```
int set_variable_float(const char* name, float value, int set_current = 1, int set_i
```

5.36.14 function `diguyGraphicsShaderInstance::set_variable_vec2`

Prototype:

```
int set_variable_vec2(const char* name, float x, float y, int set_current = 1, int s
```

5.36.15 function `diguyGraphicsShaderInstance::set_variable_vec3`

Prototype:

```
int set_variable_vec3(const char* name, float x, float y, float z, int set_current =
```

5.36.16 function diguyGraphicsShaderInstance::set_variable_vec4

Prototype:

```
int set_variable_vec4(const char* name, float x, float y, float z, float w, int set_
```

5.36.17 function diguyGraphicsShaderInstance::set_variable_int

Prototype:

```
int set_variable_int(const char* name, int value, int set_current = 1, int set_initi
```

5.36.18 function diguyGraphicsShaderInstance::get_variable_float

Prototype:

```
int get_variable_float(const char* name, float* value, int get_current = 1);
```

5.36.19 function diguyGraphicsShaderInstance::get_variable_vec2

Prototype:

```
int get_variable_vec2(const char* name, float* x, float* y, int get_current = 1);
```

5.36.20 function `diguyGraphicsShaderInstance::get_variable_vec3`

Prototype:

```
int get_variable_vec3(const char* name, float* x, float* y, float* z, int get_current)
```

5.36.21 function `diguyGraphicsShaderInstance::get_variable_vec4`

Prototype:

```
int get_variable_vec4(const char* name, float* r, float* g, float* b, float* a, int get_current)
```


5.36.22 function diguyGraphicsShaderInstance::get_variable_int**Prototype:**

```
int get_variable_int(const char* name, int* value, int get_current = 1);
```

Deprecated Functions

The following functions are deprecated.

As these functions may disappear in a future version of DI-Guy, it is recommended that you remove calls to these functions from your application.

5.36.23 function diguyGraphicsShaderInstance::get_shader_program

Prototype:

```
diguyGraphicsShaderProgram* get_shader_program();
```

Deprecated Functions: changed due to API restructuring

This function returns the shader program this shader instance is associated with.

Returns:

pointer to type [diguyGraphicsShaderProgram](#); should never be NULL

Callable From:

◇ C++

5.36.24 function diguyGraphicsShaderInstance::get_num_link_matrices

Prototype:

```
int get_num_link_matrices();
```

Description:

This function has been deprecated. Skinned matrices are now bound at a per shape level.

This function returns the number of link matrices this instance needs to support. This is useful to know when requesting resources during shader initialization (e.g., requesting uniform variable space).

This same number will be passed to [update_link_matrices\(\)](#) and [bind_link_matrices\(\)](#).

Returns:

number of link matrices needed by this shader instance

Callable From:

◇ C++

5.36.25 function

diguyGraphicsShaderInstance::update_link_matrices

Prototype:

```
virtual int update_link_matrices(int num_matrices,
                                const float* mat_data);
```

Description:

This function has been deprecated as of DI-Guy 12. Skinned matrices are now updated at a per shape level.

This function will be called by DI-Guy when the matrix data of a character's links needs to be updated. Scene graph renderers will queue up this data to be sent to the graphics card at the appropriate time.

Note that the format of the data in `mat_data` will be affected by the [bdiGraphicsInitGraphicsAPI](#) variable `transpose_shader_matrices` value.

Immediate Mode:

Immediate mode renderers usually do not override this function, but instead use `bind_link_matrices( )` during the Draw Stage.

Scene Graph:

Scene graph renderers usually do override this function.

Callable From:

◊ N/A (automatically called by DI-Guy Graphics API during the Update Stage)

5.37 diguyGraphicsShaderProgram

diguyGraphicsShaderProgram.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Accessor Functions](#)
- ◆ [Texture Management](#)
- ◆ [Virtual Functions](#)
- ◆ [Shader Features](#)
- ◆ [Matrix Format Functions](#)
- ◆ [Shader Variables](#)
- ◆ [Private Functionality](#)

Link against: `libdiguy_graphics_api`

5.37.1 enumeration diguyGraphicsShaderQualityLevel

```
typedef enum
{
    DIGUY_GRAPHICS_SHADER_QUALITY_UNKNOWN = -1,
    DIGUY_GRAPHICS_SHADER_QUALITY_UNLIT = 0,
    DIGUY_GRAPHICS_SHADER_QUALITY_PER_VERTEX = 1,
    DIGUY_GRAPHICS_SHADER_QUALITY_PER_PIXEL = 2,
    DIGUY_GRAPHICS_SHADER_QUALITY_PER_PIXEL_BUMP_MAPPED = 3,
    DIGUY_GRAPHICS_SHADER_QUALITY_MAX
} diguyGraphicsShaderQualityLevel;
```

Description:

This enumeration lists the quality level of various shader programs that DI-Guy might create.

```
int
diguyApp::set_max_character_shader_quality_level(int
quality_level); int
diguyApp::set_max_scene_object_shader_quality_level(int
quality_level);
```

5.37.2 enumeration `diguyGraphicsShaderProgramType`

```
typedef enum
{
    DIGUY_GRAPHICS_SHADER_PROGRAM_TYPE_UNKNOWN = -1,
    DIGUY_GRAPHICS_SHADER_PROGRAM_TYPE_GENERAL = 0,
    DIGUY_GRAPHICS_SHADER_PROGRAM_TYPE_SKIN,
    DIGUY_GRAPHICS_SHADER_PROGRAM_TYPE_REFLECTOR
} diguyGraphicsShaderProgramType;
```

Description:

This enumeration lists the types of shader programs that DI-Guy might request.

DIGUY_GRAPHICS_SHADER_PROGRAM_TYPE_GENERAL

DIGUY_GRAPHICS_SHADER_PROGRAM_TYPE_SKIN

DIGUY_GRAPHICS_SHADER_PROGRAM_TYPE_REFLECTOR

This class implements shader programs for use with DI-Guy character graphics.

Shader **programs** are shared among all characters that use the shader program. Any variables used in a shader program should therefore apply to all characters using the shader.

Shader **instances** are not shared. Each character that has a shader program assigned to it will have a private shader instance in which variables specific to the character or its instance can be stored. A common case for this is current values of shader uniform variables.

Also see [diguyGraphicsShaderInstance](#).

5.37.3 class diguyGraphicsShaderProgram

```

class BDI_DECLSPEC diguy diguyGraphicsShaderProgram
{
public:
    const char* get_name();
    const char* get_vertex_shader_filename();
    const char* get_pixel_shader_filename();
    diguyGraphicsShaderProgramType get_shader_type();
    diguyGraphicsShaderQualityLevel get_quality_level();
    void derive_diguy_shader_source_filename(char* result,
        int result_max_len,
        const char* shader_base_name,
        const char* shader_language,
        const char* shader_type,
        int include_diguy_default_shader_directory);
    void print_shader_info_to_log();
    int load();
    int set_textures_used_bitmask(int texture_bitmask);
    int get_textures_used_bitmask();

#ifdef CPLUSPLUS_ONLY
    virtual int load_vertex_shader_source(const char* filename_base);
    virtual int load_pixel_shader_source(const char* filename_base);
    virtual int reload();
    virtual int link();
    virtual int get_loaded();
    virtual int get_linked();
    virtual int get_valid();
    virtual int bind();
    virtual int unbind();
    virtual int bind_link_matrices(int num_matrices,
        const float* mat_data);
    virtual int bind_color_array(int num_colors,
        const float* color_data);
    virtual int bind_uniform_with_location(int location,
        const char* name,
        int value_size,
        float* values);
    virtual int bind_uniform_with_location(int location,
        const char* name,
        int value_size,
        int* values);
    virtual int get_attrib_bind_location(const char* name);
    virtual int get_uniform_bind_location(const char* name);
#endif

    int get_num_additional_feature_names();
    const char * get_additional_feature_name(int index);
    int has_additional_feature(const char * name);
    static int get_transposed_matrix_data();

    static int get_use_4x3_shader_matrix_data();

    int get_num_variables();
    const char* get_variable_name_at_index(int index);
    int has_variable(const char* name);
    const char* get_variable_type(const char* name);
    int set_variable_float(const char* name, float value, int set_current = 1, int set_in);
    int set_variable_vec2(const char* name, float x, float y, int set_current = 1, int set_in);

```

```

int set_variable_vec3(const char* name, float x, float y, float z, int set_current = 1);
int set_variable_vec4(const char* name, float x, float y, float z, float w, int set_current = 1);
int set_variable_int(const char* name, int value, int set_current = 1, int set_initial = 0);
int get_variable_float(const char* name, float* value, int get_current = 1);
int get_variable_vec2(const char* name, float* x, float* y, int get_current = 1);
int get_variable_vec3(const char* name, float* x, float* y, float* z, int get_current = 1);
int get_variable_vec4(const char* name, float* r, float* g, float* b, float* a, int get_current = 1);
int get_variable_int(const char* name, int* value, int get_current = 1);

```

Accessor Functions

The functions in this section provide access to data in this object. They are designed to be called by DI-Guy programmers.

All are Callable From:

- ◆ C++
- ◆ Script

5.37.4 function diguyGraphicsShaderProgram::get_name

Prototype:

```
const char* get_name();
```

Returns:

name of the shader This pointer will never be NULL.

5.37.5 function diguyGraphicsShaderProgram::get_vertex_shader_filename

Prototype:

```
const char* get_vertex_shader_filename();
```

Returns:

the vertex filename of the shader This pointer will never be NULL.

5.37.6 function `diguyGraphicsShaderProgram::get_pixel_shader_filename`

Prototype:

```
const char* get_pixel_shader_filename();
```

Returns:

the pixel shader filename of the shader This pointer will never be NULL.

5.37.7 function `diguyGraphicsShaderProgram::get_shader_type`

Prototype:

```
diguyGraphicsShaderProgramType get_shader_type();
```

Returns:

program type of the shader; see [diguyGraphicsShaderProgramType](#)

5.37.8 function diguyGraphicsShaderProgram::get_quality_level**Prototype:**

```
diguyGraphicsShaderQualityLevel get_quality_level();
```

Returns:

the quality level type of the shader; see
[diguyGraphicsShaderQualityLevel](#)

5.37.9 function diguyGraphicsShaderProgram::derive_diguy_shader_source_filename**Prototype:**

```
void derive_diguy_shader_source_filename(char* result,
    int result_max_len,
    const char* shader_base_name,
    const char* shader_language,
    const char* shader_type,
    int include_diguy_default_shader_directory);
```

Description:

This function derives filenames for shader programs from the provided information.

Arguments:

<code>result</code>	string buffer in which result should be placed
<code>result_max_len</code>	

	size of supplied string buffer; too small will result in a truncated filename
<code>shader_base_name</code>	base name of shader program; e.g. "diguy_character"
<code>shader_language</code>	language of shader program; e.g. "glsl" or "hlsl"
<code>shader_type</code>	shader program type; e.g. "vert" or "frag"
<code>include_diguy_default_shader_directory</code>	pass 1 to include the default DI-Guy shader directory in the filename; pass 0 to return the filename only with no path

The constructed filename has the format
`"(shader_base_name)_(shader_language).(shader_type)"`; e.g.,
`"diguy_character_glsl.vert"`.

The DI-Guy shader directory defaults to `$DIGUYgeometryshaders`.

Returns:

filename result in the result argument

5.37.10 function diguyGraphicsShaderProgram::print_shader_info_to_log

Prototype:

```
void print_shader_info_to_log();
```

Description:

Prints information about the shader to the log.

5.37.11 function diguyGraphicsShaderProgram::load

Prototype:

```
int load();
```

Description:

Forces the underlying shader to load. Returns 0 on success.

Texture Management

5.37.12 function `diguyGraphicsShaderProgram::set_textures_used_bitmask`

Prototype:

```
int set_textures_used_bitmask(int texture_bitmask);
```

Description:

Sets a bit mask consisting of [diguyTextureMapMasks](#) or'd together to specify what textures should bind when this shader is active.

See

%DIGUY%\programming_examples\diguy_graphics_api\ogl_examples\common\diguyOglGraphicsS for an example of usage. By default in the graphics API we attempt to bind all textures.

5.37.13 function `diguyGraphicsShaderProgram::get_textures_used_bitmask`

Prototype:

```
int get_textures_used_bitmask();
```

Returns:

the current or'd together list of `diguyTextureMapMasks` that this shader attempts to bind. Defaults to `DIGUY_TEXTURES_MAP_MASK_ALL` (0xffffffff)

Virtual Functions

The functions in this section are all virtual functions that will be called by DI-Guy at various times during DI-Guy execution.

Because the functions are all virtual, DI-Guy programmers can create a subclass of this class and override them. Unless otherwise stated, the base class functions do not need to be called.

5.37.14 function diguyGraphicsShaderProgram::load_vertex_shader_source

Prototype:

```
virtual int load_vertex_shader_source(const char* filename_base);
```

Description:

This function will be called by DI-Guy when it is time for the vertex part of this shader to be loaded and compiled. The utility function [derive_diguy_shader_source_filename\(\)](#) can be used to convert the passed filename_base to an actual filename.

For example, assume that m_vertex_shader_filename is a character array of 512 bytes. The following function call would derive an appropriate vertex shader filename:

```
derive_diguy_shader_source_filename(m_vertex_shader_filename,
    511,
    filename_base,
    "glsl",
    "vert",
    1);
```

This function is called during the Build Stage.

Returns:

0 to indicate load was successful, -1 otherwise

Immediate Mode:

Immediate mode renderers usually do override this function.

Scene Graph:

Scene graph renderers usually do override this function.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during the Build Stage)

5.37.15 function diguyGraphicsShaderProgram::load_pixel_shader_source

Prototype:

```
virtual int load_pixel_shader_source(const char* filename_base);
```

Description:

This function is the pixel shader (AKA fragment shader in OpenGL) equivalent of [load_vertex_shader_source\(\)](#).

5.37.16 function diguyGraphicsShaderProgram::reload

Prototype:

```
virtual int reload();
```

Description:

This function will be called by DI-Guy if this shader program should be reloaded. The same filenames derived in the [load_vertex_shader_source\(\)](#) and [load_pixel_shader_source\(\)](#) functions should be used.

This function can be called during the Update Stage.

Returns:

0 to indicate reload was successful, -1 otherwise

Immediate Mode:

Immediate mode renderers usually do override this function.

Scene Graph:

Scene graph renderers usually do override this function.

Callable From:

◊ N/A (automatically called by DI-Guy Graphics API during the Build Stage)

5.37.17 function diguyGraphicsShaderProgram::link

Prototype:

```
virtual int link();
```

Description:

This function will be called by DI-Guy when it is time for the shader program to be linked.

This function is called during the Build Stage after the `load_vertex_shader_source()` and `load_pixel_shader_source()` functions.

Once the link has successfully completed shader variable bind indices can be looked up.

Returns:

0 to indicate link was successful, -1 otherwise

Immediate Mode:

Immediate mode renderers usually do override this function.

Scene Graph:

Scene graph renderers can override this function.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during the Build Stage)

5.37.18 function diguyGraphicsShaderProgram::get_loaded

Prototype:

```
virtual int get_loaded();
```

Description:

Returns whether the shader has been loaded. Note that because a shader has been loaded doesn't mean that it is valid. See [get_valid\(\)](#).

Returns:

1 if shader has been loaded; 0 if not

5.37.19 function diguyGraphicsShaderProgram::get_linked**Prototype:**

```
virtual int get_linked();
```

Description:

Returns whether the shader has been linked. Note that this only means that it is been attempted to link the shader. The link may have failed, and the shader not be valid. See [get_valid\(\)](#).

Returns:

1 if shader has been linked; 0 if not

5.37.20 function diguyGraphicsShaderProgram::get_valid**Prototype:**

```
virtual int get_valid();
```

Description:

Returns whether shader is valid. An invalid shader will not work correctly, leaving graphics that use this shader unshaded. What this will look like visually is undetermined.

Returns:

1 if shader is valid; 0 if not

5.37.21 function diguyGraphicsShaderProgram::bind

Prototype:

```
virtual int bind();
```

Description:

This function will be called by DI-Guy when this program is being bound for rendering.

It is during this function call that program level shader variables should be set. Program level variables will affect **all** shader operations in which this program is active. Variables that should affect specific instances of a shader should be set in [diguyGraphicsShaderInstance::bind\(\)](#).

One example in the DI-Guy reference shaders is the `ufrn_textures_enabled` variable. Setting it to 0 will prevent textures from being applied during the shader program run.

This happens during the Draw Stage.

Immediate Mode:

Immediate mode renderers can override this function.

Scene Graph:

Scene graph renderers usually do not override this function; the renderer will typically do this internally. If shader variables need to be bound, however, this function should be overridden.

Callable From:

◊ N/A (automatically called by DI-Guy Graphics API during the Draw Stage)

5.37.22 function diguyGraphicsShaderProgram::unbind

Prototype:

```
virtual int unbind();
```

Description:

In general variables that were set in `bind()` should be unset or unbound here.

This happens during the Draw Stage.

Immediate Mode:

Immediate mode renderers can override this function.

Scene Graph:

Scene graph renderers usually do not override this function; the renderer will typically do this internally. If the `bind()` function was overridden, however, this function probably should be, too.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during the Draw Stage)

5.37.23 function `diguyGraphicsShaderProgram::bind_link_matrices`

Prototype:

```
virtual int bind_link_matrices(int num_matrices,
                              const float* mat_data);
```

Description:

This function will be called by DI-Guy when the matrix data of a character's links needs to be sent to the graphics card.

For GLSL-based shaders, these values are typically passed into the vertex shader via uniform variables.

Note that the format of the data in `mat_data` will be affected by the `bdiGraphicsInitGraphicsAPI` variable `transpose_shader_matrices` value.

Immediate Mode:

Immediate mode renderers usually do override this function.

Scene Graph:

Scene graph renderers usually do not override this function, but instead use `diguyGraphicsShape::get_shader_matrix_data( )` during the Update Stage.

Callable From:

◊ N/A (automatically called by DI-Guy Graphics API during the Draw Stage)

5.37.24 function diguyGraphicsShaderProgram::bind_color_array

Prototype:

```
virtual int bind_color_array(int num_colors,
    const float* color_data);
```

Description:

This function will be called by DI-Guy when the color data of a character's links needs to be sent to the graphics card.

For GLSL-based shaders, these values are typically passed into the vertex shader via uniform variables.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during the Draw Stage)

5.37.25 function diguyGraphicsShaderProgram::bind_uniform_with_location

Prototype:

```
virtual int bind_uniform_with_location(int location,
    const char* name,
    int value_size,
    float* values);
```

Description:

These functions will be called by DI-Guy when there is a shader uniform that needs to be updated and sent to the graphics card.

For GLSL-based shaders, these values are typically passed into the shader via uniform variables. This function will get repeatedly invoked if a character is using per shape materials. See [diguyGraphicsShape::set_material_parameter_4f\(\)](#) for more details.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during the Draw Stage)

Arguments:

<code>location</code>	the shader uniform to feed this data to, this is the value that was retrieved via a call to <code>get_ufrm_bind_location()</code>
<code>name</code>	string representation of the uniform, useful for debugging
<code>value_size</code>	number of ints or floats to send to the video card
<code>values</code>	pointer to ints or floats to send to the video card

5.37.26 function

`diguyGraphicsShaderProgram::bind_uniform_with_location`

Prototype:

```
virtual int bind_uniform_with_location(int location,
    const char* name,
    int value_size,
    int* values);
```

5.37.27 function `diguyGraphicsShaderProgram::get_attrib_bind_location`

Prototype:

```
virtual int get_attrib_bind_location(const char* name);
```

Description:

This function will be called by DI-Guy when core code is attempting to identify the capabilities of the end users shader. Currently attribute locations are unused. Return -1 to disable any functionality.

5.37.28 function `diguyGraphicsShaderProgram::get_uniform_bind_location`

Prototype:

```
virtual int get_uniform_bind_location(const char* name);
```

Description:

This function will be called by DI-Guy when core code is attempting to identify the capabilities of the end users shader. This needs to be implemented to enable per shape advanced materials. Return -1 to disable any functionality.

Shader Features

Shader config files have an optional list of additional features that can be used to data drive program loading.

5.37.29 function diguyGraphicsShaderProgram::get_num_additional_feature_names

Prototype:

```
int get_num_additional_feature_names();
```

5.37.30 function `diguyGraphicsShaderProgram::get_additional_feature_name`

Prototype:

```
const char * get_additional_feature_name(int index);
```

5.37.31 function `diguyGraphicsShaderProgram::has_additional_feature`

Prototype:

```
int has_additional_feature(const char * name);
```

Matrix Format Functions

5.37.32 function

`diguyGraphicsShaderProgram::get_transposed_matrix_data`

Prototype:

```
static int get_transposed_matrix_data();
```

Returns:

The value specified in `bdiGraphicsInitGraphicsAPI.transpose_shader_matrices`. Indicates if DI-Guy will attempt to transpose 4x4 matrices before sending to video card.

5.37.33 function `diguyGraphicsShaderProgram::get_use_4x3_shader_matrix_data`

Prototype:

```
static int get_use_4x3_shader_matrix_data();
```

Returns:

The value specified in `bdiGraphicsInitGraphicsAPI.use_4x3_shader_matrices`. Indicates if DI-Guy is sending 4x3 matrix data instead of 4x4 data.

Shader Variables

For GLSL shaders, these variables are represented internally as uniform variables. DI-Guy reference shaders commonly use the prefix "ufm_" for their uniform variables.

Program level shader variables are shared between all shader instances that use this program. Variables that should affect specific instances associated with individual objects (typically characters and scene objects) should be set at the shader instance level, using equivalent functions in `diguyGraphicsShaderInstance`.

5.37.34 function `diguyGraphicsShaderProgram::get_num_variables`

Prototype:

```
int get_num_variables();
```

5.37.35 function `diguyGraphicsShaderProgram::get_variable_name_at_index`

Prototype:

```
const char* get_variable_name_at_index(int index);
```

5.37.36 function diguyGraphicsShaderProgram::has_variable**Prototype:**

```
int has_variable(const char* name);
```

**5.37.37 function
diguyGraphicsShaderProgram::get_variable_type****Prototype:**

```
const char* get_variable_type(const char* name);
```


5.37.38 function diguyGraphicsShaderProgram::set_variable_float

Prototype:

```
int set_variable_float(const char* name, float value, int set_current = 1, int set_i
```

5.37.39 function diguyGraphicsShaderProgram::set_variable_vec2

Prototype:

```
int set_variable_vec2(const char* name, float x, float y, int set_current = 1, int s
```

5.37.40 function diguyGraphicsShaderProgram::set_variable_vec3

Prototype:

```
int set_variable_vec3(const char* name, float x, float y, float z, int set_current = 0)
```

5.37.41 function diguyGraphicsShaderProgram::set_variable_vec4

Prototype:

```
int set_variable_vec4(const char* name, float x, float y, float z, float w, int set_current = 0)
```

5.37.42 function diguyGraphicsShaderProgram::set_variable_int**Prototype:**

```
int set_variable_int(const char* name, int value, int set_current = 1, int set_initi
```

**5.37.43 function
diguyGraphicsShaderProgram::get_variable_float****Prototype:**

```
int get_variable_float(const char* name, float* value, int get_current = 1);
```

5.37.44 function diguyGraphicsShaderProgram::get_variable_vec2

Prototype:

```
int get_variable_vec2(const char* name, float* x, float* y, int get_current = 1);
```

5.37.45 function diguyGraphicsShaderProgram::get_variable_vec3

Prototype:

```
int get_variable_vec3(const char* name, float* x, float* y, float* z, int get_current = 1);
```

5.37.46 function diguyGraphicsShaderProgram::get_variable_vec4

Prototype:

```
int get_variable_vec4(const char* name, float* r, float* g, float* b, float* a, int
```

5.37.47 function diguyGraphicsShaderProgram::get_variable_int

Prototype:

```
int get_variable_int(const char* name, int* value, int get_current = 1);
```

Private Functionality

5.38 diguyGraphicsShaderTechnique

diguyGraphicsShaderTechnique.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Accessor Functions](#)
- ◆ [Shader Creation Functions](#)
- ◆ [Shader Uniform Functions](#)
- ◆ [Virtual Functions](#)
- ◆ [Private Functionality](#)

Link against: libdiguy_graphics_api

This struct is passed into [diguyGraphicsShaderTechnique::pick_shader_program\(\)](#) to provide the end user information to base the resulting shader choice.

This class implements shader lod system for use with DI-Guy character graphics.

Shader **Techniques** are shared among all characters that use the shader Technique. Any variables used in a shader Technique should therefore apply to all characters using the shader.

Also see [diguyGraphicsShaderInstance](#).

5.38.1 class diguyGraphicsShaderTechnique

```
class BDI_DECLSPEC_diguy diguyGraphicsShaderTechnique
{
public:
    const char* get_name();
    int get_num_shader_lod_programs();
    diguyGraphicsShaderProgram* get_shader_lod_program_at_index(int index);
    float get_shader_lod_range_at_index(int index);
    diguyGraphicsShaderProgram * get_shader_by_distance(float distance, int max_quality_level);

    int get_program_index_by_distance(float distance, int max_quality_level);
    diguyGraphicsShaderProgram* get_intersection_shader();
    diguyGraphicsShaderProgram* get_shadow_pass_shader();
    diguyGraphicsShaderProgram * find_or_create_shader_variation(diguyGraphicsShaderProgram* program);
    void add_additional_shader(diguyGraphicsShaderProgram* program);
    int get_num_additional_shaders();
    diguyGraphicsShaderProgram* get_additional_shader_at_index(int index);
    int set_variable_float(const char* name, float value, int set_current = 1, int set_index = 0);
    int set_variable_vec2(const char* name, float x, float y, int set_current = 1, int set_index = 0);
    int set_variable_vec3(const char* name, float x, float y, float z, int set_current = 1, int set_index = 0);
    int set_variable_vec4(const char* name, float x, float y, float z, float w, int set_current = 1, int set_index = 0);
```

```
int set_variable_int(const char* name, int value, int set_current = 1, int set_initial = 0);  
virtual diguyGraphicsShaderProgram * pick_shader_program(const diguyShaderPickDescription * pick_desc);  
  
virtual void build();
```

Accessor Functions

The functions in this section provide access to data in this object. They are designed to be called by DI-Guy programmers.

All functions are Callable From:

- ◆ C++
- ◆ Script

5.38.2 function diguyGraphicsShaderTechnique::get_name**Prototype:**

```
const char* get_name();
```

Returns:

the name of the shader technique. This pointer will never be NULL.

5.38.3 function diguyGraphicsShaderTechnique::get_num_shader_lod_programs**Prototype:**

```
int get_num_shader_lod_programs();
```

Description:

Returns the number of lod shaders the technique has.

5.38.4 function**diguyGraphicsShaderTechnique::get_shader_lod_program_at_index****Prototype:**

```
diguyGraphicsShaderProgram* get_shader_lod_program_at_index(int index);
```

Description:

Returns the shader program at a lod index.

5.38.5 function**diguyGraphicsShaderTechnique::get_shader_lod_range_at_index****Prototype:**

```
float get_shader_lod_range_at_index(int index);
```

Description:

Returns the distance that a shader lod switches out at.

5.38.6 function `diguyGraphicsShaderTechnique::get_shader_by_distance`

Prototype:

```
diguyGraphicsShaderProgram * get_shader_by_distance(float distance, int max_quality_
```

Description:

Returns a shader based on the rendering distance.

5.38.7 function `diguyGraphicsShaderTechnique::get_program_index_by_distanc`

Prototype:

```
int get_program_index_by_distance(float distance, int max_quality_level);
```

Description:

Returns the shader index based on the rendering distance.

5.38.8 function `diguyGraphicsShaderTechnique::get_intersection_shader`

Prototype:

```
diguyGraphicsShaderProgram* get_intersection_shader();
```

Description:

Returns the shader specified for intersections.

5.38.9 function `diguyGraphicsShaderTechnique::get_shadow_pass_shader`

Prototype:

```
diguyGraphicsShaderProgram* get_shadow_pass_shader();
```

Description:

Returns the shader specified for shadow map generation.

Shader Creation Functions

5.38.10 function

`diguyGraphicsShaderTechnique::find_or_create_shader_variation`

Prototype:

```
diguyGraphicsShaderProgram * find_or_create_shader_variation(diguyGraphicsShaderProg
```

Description:

This function allows you create a new shader that uses the same vertex and pixel files as the one passed in. The user is then able modify the subclassed shader before calling `load()`. This function should allow end users to create an arbitrary number of shader permutations and then return them via overriding `pick_shader_program()`. See `diguyOglGraphicsShaderTechnique::build()` for an example of it's usage. Override `pick_shader_program()` to return a procedurally created shader.

5.38.11 function `diguyGraphicsShaderTechnique::add_additional_shader`

Prototype:

```
void add_additional_shader(diguyGraphicsShaderProgram* program);
```

Description:

Adds an additional shader program to the technique these will get automatically updated if any of the shader uniform functions are used. These are mostly for convenience. Users will probably want their own data structures to store shader lookup tables.

5.38.12 function `diguyGraphicsShaderTechnique::get_num_additional_shaders`

Prototype:

```
int get_num_additional_shaders();
```

Returns:

The number of shaders that have been procedurally added to the technique.

5.38.13 function `diguyGraphicsShaderTechnique::get_additional_shader_at_index`

Prototype:

```
diguyGraphicsShaderProgram* get_additional_shader_at_index(int index);
```

Returns:

A pointer to the procedural shaders that has been procedurally added to the technique.

Shader Uniform Functions

5.38.14 function `diguyGraphicsShaderTechnique::set_variable_float`

Prototype:

```
int set_variable_float(const char* name, float value, int set_current = 1, int set_i
```

Description:

Convenience functions. Loops through all shaders in this technique and sets uniform variables if they exist.

Callable From:

- ◇ C++
- ◇ Script

5.38.15 function `diguyGraphicsShaderTechnique::set_variable_vec2`

Prototype:

```
int set_variable_vec2(const char* name, float x, float y, int set_current = 1, int s
```


5.38.16 function diguyGraphicsShaderTechnique::set_variable_vec3

Prototype:

```
int set_variable_vec3(const char* name, float x, float y, float z, int set_current = 0)
```

5.38.17 function diguyGraphicsShaderTechnique::set_variable_vec4

Prototype:

```
int set_variable_vec4(const char* name, float x, float y, float z, float w, int set_current = 0)
```

5.38.18 function `diguyGraphicsShaderTechnique::set_variable_int`

Prototype:

```
int set_variable_int(const char* name, int value, int set_current = 1, int set_initi
```

Virtual Functions

The functions in this section are all virtual functions that will be called by DI-Guy at various times during DI-Guy execution.

Because the functions are all virtual, DI-Guy programmers can create a subclass of this class and override them. Unless otherwise stated, the base class functions do not need to be called.

5.38.19 function diguyGraphicsShaderTechnique::pick_shader_program

Prototype:

```
virtual diguyGraphicsShaderProgram * pick_shader_program(const diguyShaderPickDescr
```

Description:

Allows users to use a subclass to override the default shader LOD system see diguyOglGraphicsShaderTechnique::pick_shader_program() for an example of it's usage.

5.38.20 function diguyGraphicsShaderTechnique::build

Prototype:

```
virtual void build();
```

Description:

Allows users to do their own post shader load behavior.

Private Functionality

5.39 diguyGraphicsShape

diguyGraphicsShape.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Accessor Functions](#)
- ◆ [Virtual Functions](#)

Link against: **libdiguy**

5.39.1 class diguyGraphicsShape

```
class BDI_DECLSPEC_diguy diguyGraphicsShape
{
public:
    const char* get_name();
    long get_uid();
    diguyCharacter* get_character();
    int get_has_offset_translation();
    int get_offset_translation(float* tx, float* ty, float* tz);
    int get_has_offset_rotation();
    int get_offset_rotation(float* rz, float* rx, float* ry);
    int get_has_offset_scale();
    int get_offset_scale(float* sx, float* sy, float* sz);
    const char* get_shape_set_name();
    const char* get_shape_set_shape_name();
    const char* get_recommended_attachment_link_name();
    int get_num_lods();
#ifdef CPLUSPLUS_ONLY
    float* get_lod_ranges();
#endif
    int get_graphics_lod();
    const char* get_shape_set_filename_for_lod(int lod);
    int get_num_meshes(int lod);
    diguyGraphicsMesh* get_mesh_at_index_for_lod(int index, int lod);
    diguyGraphicsLink* get_link();
    const char* get_switch_var_name();
    int get_switch_value();
    int get_is_hidden();
    diguyGraphicsFile* get_diguy_graphics_file(int lod);
#ifdef CPLUSPLUS_ONLY
    void* get_diguy_graphics_file_part(int lod);
#endif
    int get_graphics_state_switch_index();
    int get_num_switched_graphics_states();
    diguyGraphicsState* get_switched_graphics_state(int switch_value = -1);
    int get_unique_color(unsigned char* r,
        unsigned char* g,
        unsigned char* b);
};
```

```

int get_contains_skinned_file();
int set_material_parameter_4f(const char* parameter_name,
    float x, float y, float z, float w,
    float blend_time = 0.0);
int get_material_parameter_4f(const char* parameter_name,
    float* x, float* y, float* z, float* w);
unsigned char * get_variation_texture();
int get_variation_texture_size();
#ifdef CPLUSPLUS_ONLY
    int get_shader_matrix_data(int* num_matrixes, const float** matrix_data);
#endif
#ifdef CPLUSPLUS_ONLY
    virtual void build();
    virtual void unbuild();
    virtual void update();
    virtual void show();
    virtual void hide();
    virtual void draw(int lod);
    virtual void set_graphics_lod_ranges(float* ranges);
    virtual void set_graphics_lod(int lod);
    virtual void update_variation_texture(int num_pixels, unsigned char * pixel_data);
    int get_bind_color_array(int* num_colors, const float** color_data);

#endif

```

Accessor Functions

The functions in this section provide access to data in this object. They are designed to be called by DI-Guy programmers.

Unless otherwise noted all functions in this class section are callable from:

- ◆ C++
- ◆ Script

5.39.2 function diguyGraphicsShape::get_name

Prototype:

```
const char* get_name();
```

Description:

This function returns the name of the shape. The returned pointer will never be NULL.

Returns:

name of the shape as C string

5.39.3 function diguyGraphicsShape::get_uid

Prototype:

```
long get_uid();
```

Description:

All shapes are assigned a unique identifier, or uid. This function returns this shape's uid.

Note: unique identifiers can change between DI-Guy runs.

Returns:

unique identifier of object

5.39.4 function diguyGraphicsShape::get_character

Prototype:

```
diguyCharacter* get_character();
```

Description:

This function returns a pointer to the [diguyCharacter](#) to which this shape belongs.

Returns:

pointer of type [diguyCharacter](#)

5.39.5 function diguyGraphicsShape::get_has_offset_translation

Prototype:

```
int get_has_offset_translation();
```

Description:

This function returns a 1 if the shape has a translation offset from its link, 0 otherwise.

Returns:

1 if shape has an offset, 0 if not, -1 on error

See Also:

[get_offset_translation\(\)](#)

5.39.6 function diguyGraphicsShape::get_offset_translation

Prototype:

```
int get_offset_translation(float* tx, float* ty, float* tz);
```

Description:

This function returns in the passed variables the translation offset of this shape from its link.

The offset will always remain fixed once DI-Guy is initialized.

Returned values are in meters.

Arguments:

tx, ty, tz pointers to floats into which result should be placed

Returns:

0 on success, -1 on failure

See Also:

[get_has_offset_translation\(\)](#), [get_offset_rotation\(\)](#),
[get_offset_scale\(\)](#)

5.39.7 function diguyGraphicsShape::get_has_offset_rotation**Prototype:**

```
int get_has_offset_rotation();
```

Description:

This function returns a 1 if the shape has a rotation offset from its link, 0 otherwise.

Returns:

1 if shape has an offset, 0 if not, -1 on error

See Also:

[get_offset_rotation\(\)](#)

5.39.8 function diguyGraphicsShape::get_offset_rotation**Prototype:**

```
int get_offset_rotation(float* rz, float* rx, float* ry);
```

Description:

This function returns in the passed variables the rotational offset of this shape from its link.

The offset will always remain fixed once DI-Guy is initialized.

Returned values are an Euler triple in degrees, and should be applied in RZ -> RX -> RY order, after the translation offset.

Arguments:

`rz, rx, ry` pointers to floats into which result should be placed

Returns:

0 on success, -1 on failure

See Also:

[get_has_offset_rotation\(\)](#), [get_offset_translation\(\)](#),
[get_offset_scale\(\)](#)

5.39.9 function diguyGraphicsShape::get_has_offset_scale

Prototype:

```
int get_has_offset_scale();
```

Description:

This function returns a 1 if the shape has a scale offset from its link, 0 otherwise.

Returns:

1 if shape has an offset, 0 if not, -1 on error

See Also:

[get_offset_scale\(\)](#)

5.39.10 function diguyGraphicsShape::get_offset_scale**Prototype:**

```
int get_offset_scale(float* sx, float* sy, float* sz);
```

Description:

This function returns in the passed variables the scale "offset" of this shape from its link.

The offset will always remain fixed once DI-Guy is initialized.

Returned values are unitless scale factors.

Arguments:

`sz`, `sx`, `sy` pointers to floats into which result should be placed

See Also:

[get_has_offset_scale\(\)](#), [get_offset_translation\(\)](#),
[get_offset_rotation\(\)](#)

Returns:

0 on success, -1 on failure

5.39.11 function diguyGraphicsShape::get_shape_set_name**Prototype:**

```
const char* get_shape_set_name();
```

Description:

This function returns the name of the shape set of which this shape is a part.

A shape set is a collection of shapes that are related, coming from a set of related geometry files, sometimes one file per LOD.

A shape is a collection of parts each representing the same thing at varying levels of detail.

For example, in its config files DI-Guy has a shape set named "firefighter", which references five OpenFlight files named firefighter_LOD1.flt through firefighter_LOD5.flt. The firefighter shape set contains separate shapes for the head, arms, legs, etc., at five different levels of detail.

Use the function [get_shape_set_filename_for_lod\(\)](#) to get the actual geometry filenames that should be used for each LOD.

Returns:

shape set of shape as C string

5.39.12 function diguyGraphicsShape::get_shape_set_shape_name

Prototype:

```
const char* get_shape_set_shape_name();
```

Description:

This function returns the name of the shape. This name identifies which pieces of geometry out of the shape set should comprise the shape.

Returns:

shape name of shape as C string

5.39.13 function diguyGraphicsShape::get_recommended_attachment_link_name

Prototype:

```
const char* get_recommended_attachment_link_name();
```

Description:

This function returns the name of the link to which this shape is typically attached.

Use the function [get_link\(\)](#) to get a pointer to the actual link to which this shape is attached.

Returns:

name of recommended attachment link as C string

5.39.14 function diguyGraphicsShape::get_num_lods**Prototype:**

```
int get_num_lods();
```

Description:

This function returns how many LODs this shape has. This will be a number between 1 and 7.

Returns:

number of LODs

5.39.15 function diguyGraphicsShape::get_lod_ranges**Prototype:**

```
float* get_lod_ranges();
```

Description:

This function returns the LOD ranges of this shape. The returned pointer points to an array of eight floats, which represent ranges in meters at which LODs should be shown.

Returns:

LOD ranges

Only Callable From C++

5.39.16 function diguyGraphicsShape::get_graphics_lod**Prototype:**

```
int get_graphics_lod();
```

Description:

This function returns the most recent lod setting as set by `set_graphics_lod( )`, or as set by `diguyCharacter::set_graphics_lod( )`.

Returns:

LOD index; integer from 1 to `get_num_lods( )`

5.39.17 function diguyGraphicsShape::get_shape_set_filename_for_lod**Prototype:**

```
const char* get_shape_set_filename_for_lod(int lod);
```

Description:

This function returns the filename of the geometry file from which the shape is read for a specific LOD.

Arguments:

lod	level of detail; a number between 1 and <code>get_num_lods()</code>

Returns:

geometry filename as C string

5.39.18 function diguyGraphicsShape::get_num_meshes**Prototype:**

```
int get_num_meshes(int lod);
```

Description:

This function returns the number of low-level geometry meshes that comprise this shape.

Arguments:

lod level of detail; a number between 1 and get_num_lods()

Returns:

number of meshes in this shape

5.39.19 function diguyGraphicsShape::get_mesh_at_index_for_lod**Prototype:**

```
diguyGraphicsMesh* get_mesh_at_index_for_lod(int index, int lod);
```

Description:

This function returns a pointer to one of the low-level geometry mesh objects that comprise this shape.

Arguments:

index - index of 1
mesh; a number
between 0 and
get_num_meshes()
lod level of detail; a number between 1 and
get_num_lods()

Returns:

pointer to object of type [diguyGraphicsMesh](#); NULL on error

5.39.20 function diguyGraphicsShape::get_link

Prototype:

```
diguyGraphicsLink* get_link();
```

Description:

This function returns a pointer to the link to which this shape is actually attached.

Arguments:

index	- index of 1 mesh; a number between 0 and get_num_meshes()
lod	level of detail; a number between 1 and get_num_lods()

Returns:

pointer to object of type [diguyGraphicsMesh](#); NULL on error

5.39.21 function `diguyGraphicsShape::get_switch_var_name`

Prototype:

```
const char* get_switch_var_name();
```

Description:

This function returns the name of the DI-Guy motion data variable that controls whether this shape should be visible.

The returned pointer will be NULL if there is no associated variable.

Returns:

name of the variable; NULL if there is no variable

5.39.22 function diguyGraphicsShape::get_switch_value**Prototype:**

```
int get_switch_value();
```

Description:

This function returns the value of the DI-Guy motion data variable for which the shape should be visible.

Note that the variable in the motion data is a float, but gets cast to an integer and compared against this number during shape updates.

Returns:

number representing value for which shape should be visible

5.39.23 function diguyGraphicsShape::get_is_hidden**Prototype:**

```
int get_is_hidden();
```

Description:

This function returns whether this shape should currently be hidden or shown, based on the value of the shape's switch variable (if any).

Returns:

1 if shape should be hidden, 0 if not

5.39.24 function diguyGraphicsShape::get_diguy_graphics_file**Prototype:**

```
diguyGraphicsFile* get_diguy_graphics_file(int lod);
```

Description:

This function returns the geometry file from which geometry for the shape was read. Each LOD of the shape may potentially come from a different file.

Note that if a [diguyGraphicsFile](#) create function has not been registered with a call to [diguy_graphics_set_file_create_func\(\)](#), this function will return NULL.

Arguments:

lod level of detail; a number between 1 and get_num_lods()

Returns:

object of type [diguyGraphicsFile](#)

5.39.25 function diguyGraphicsShape::get_diguy_graphics_file_part**Prototype:**

```
void* get_diguy_graphics_file_part(int lod);
```

Description:

This function returns a pointer to an object that represents the geometry for the shape. The returned object is an object that has been returned by a previous call to [diguyGraphicsFile::find_part\(\)](#), and can be safely cast to the type of object that has been so returned.

Note that if a [diguyGraphicsFile](#) create function has not been registered with a call to [diguy_graphics_set_file_create_func\(\)](#), this function will return NULL.

Arguments:

lod level of detail; a number between 1 and get_num_lods()

Returns:

generic object which must be cast appropriately

Only Callable From C++

5.39.26 function `diguyGraphicsShape::get_graphics_state_switch_index`

Prototype:

```
int get_graphics_state_switch_index();
```

Description:

Some shapes have meshes that have multiple selectable graphics states, including materials and textures. This function returns the value that specifies which graphics state to use for meshes of this shape.

This is different than `get_switch_value( )`, which returns the value of the switch that controls whether or not meshes of this state are shown.

Returns:

value of state switch; -1 if there is no state switch

5.39.27 function `diguyGraphicsShape::get_num_switched_graphics_states`

Prototype:

```
int get_num_switched_graphics_states();
```

Description:

Returns the number of switched graphics states available to this shape. See [get_graphics_state_switch_index\(\)](#).

Returns:

number of switched graphics states

5.39.28 function `diguyGraphicsShape::get_switched_graphics_state`

Prototype:

```
diguyGraphicsState* get_switched_graphics_state(int switch_value = -1);
```

Description:

Returns the graphics state associated with the passed switch value.

Returns:

pointer to type [diguyGraphicsState](#)

5.39.29 function diguyGraphicsShape::get_unique_color**Prototype:**

```
int get_unique_color(unsigned char* r,
                    unsigned char* g,
                    unsigned char* b);
```

Description:

Returns in the passed pointers the unique color of this shape. All shapes in a DI-Guy appearance are assigned a unique RGB color that can be used in pick/selection operations.

Also see [diguyScenario::map_color_to_impact\(\)](#).

See [programming_examplesdiguy_graphics_apiogl_examples/intersection_test](#) for an example of its usage.

Returns:

0 on success, -1 on failure

5.39.30 function diguyGraphicsShape::get_contains_skinned_file**Prototype:**

```
int get_contains_skinned_file();
```

Description:

This function identifies if this shape has skinned data. Scene graph renderers will need to know this in the build phase to help set up the matrix data shader functionality.

Returns:

1 if contains at lease one file with skinned data, 0 if not

5.39.31 function

diguyGraphicsShape::set_material_parameter_4f

Prototype:

```
int set_material_parameter_4f(const char* parameter_name,
                             float x, float y, float z, float w,
                             float blend_time = 0.0);
```

Description:

By default DI-Guy provides three 4 float uniforms at the shape level that can be used by end users to customize the appearance of characters via shaders and misc textures. You can see a demonstration of this in the Exface_sales_demo.dss file where an additional texture is used. The uniforms are named:

Example:

```
local character = this_scenario:get_character_at_index(0);
local link = character:get_link_at_index(0);
local shape = link:find_shape_by_class_type("head");
shape:set_material_parameter_4f("ufrn_blood_color", 1, .1, .1, .5, 1);
```

5.39.32 function

diguyGraphicsShape::get_material_parameter_4f

Prototype:

```
int get_material_parameter_4f(const char* parameter_name,
                             float* x, float* y, float* z, float* w);
```

Description:

Allow access to the current values of a material parameter, accounting for value blending. Returns -1 if the parameter can't be found.

5.39.33 function `diguyGraphicsShape::get_variation_texture_size`

Prototype:

```
int get_variation_texture_size();
```

Description:

Returns a char * of length `get_variation_texture_size()` that represents the pixels in a unique per shape texture. Typically a Retained mode renderer would need to use this to load a state node with the texture see `diguyOsgGraphicsShape.cpp` for an example. For an immediate mode renderer we push the texture to the object via `update_variation_texture()`

5.39.34 function `diguyGraphicsShape::get_shader_matrix_data`

Prototype:

```
int get_shader_matrix_data(int* num_matrixes, const float** matrix_data);
```

Description:

This function is typically called by scene graph renderers as part of `diguyGraphicsShape::update()`. Scene graph renderers should queue up this data to be sent to the graphics card at the appropriate time.

Note that the format of the data in `mat_data` will be affected by the `bdiGraphicsInitGraphicsAPI` variable `transpose_shader_matrices` value.

Callable From:

Virtual Functions

The functions in this section are all virtual functions that will be called by DI-Guy at various times during DI-Guy execution.

Because the functions are all virtual, DI-Guy programmers can create a subclass of this class and override them. Unless otherwise stated, the base class functions do not need to be called.

5.39.35 function diguyGraphicsShape::build

Prototype:

```
virtual void build();
```

Description:

This function will be called by DI-Guy when it is time for renderer-specific shape objects to be created. All information necessary to build the shape should be available via the Accessor Functions above when this function is called.

The `build()` function for the link to which a shape is attached will be called before the shape's `build()` function.

Immediate Mode:

Immediate mode renderers can override this function, to create a pre-compiled object that expedites the `draw()`. Display lists in OpenGL are an example of this.

Scene Graph:

Scene graph renderers usually do override this function, to create scene graph nodes that will hold geometry nodes of meshes attached to the shape.

The following nodes are typically created:

- ◇ a switch node to control showing and hiding of the shape
- ◇ an LOD node to control which meshes are shown due to LOD calculations
- ◇ a geometry parent or group-type object to which mesh nodes can be attached

The scene graph nodes created by this function should be attached to the nodes of the link to which the shape is attached. Use the `get_link()` function to get a pointer to the link.

In addition, the scene graph objects of meshes attached to the shape should be retrieved and attached to the shape's nodes. Use the `get_mesh_at_index_for_lod()` function to get pointers to the meshes.

Callable From:

- ◇ N/A (automatically called by DI-Guy Graphics API during the Build Stage)

5.39.36 function diguyGraphicsShape::unbuild

Prototype:

```
virtual void unbuild();
```

Description:

This function will be called by DI-Guy when it is time for the renderer-specific shape objects created during the `build()` call to be destroyed.

The `unbuild()` function for the shape will be called before the `unbuild()` function of the link to which the shape is attached.

Immediate Mode:

Scene graph renderers can override this function, to destroy the scene graph nodes created by `build()`.

Scene Graph:

Scene graph renderers usually do override this function, to destroy the scene graph nodes created by `build()`.

Callable From:

◊ N/A (automatically called by DI-Guy Graphics API during the Unbuild Stage)

5.39.37 function diguyGraphicsShape::update

Prototype:

```
virtual void update();
```

Description:

This function will be called by DI-Guy when it is time for renderer-specific shape objects created during the `build()` call to be updated with new data.

Shape updates control whether the shape is visible based on motion data. The DI-Guy Graphics software looks at the switch value and calls `show()` or `hide()` as appropriate, so it is likely that this function does not need to be overridden in immediate mode.

The `update()` function for the link to which a shape is attached will be called before the shape's `update()` function.

Immediate Mode:

Immediate mode renderers usually do not override this function.

Scene Graph:

Scene graph renderers should override this function if they are supporting skinned characters. Each skinned shape can potentially have different uniforms that need to be sent to the video card; see [get_shader_matrix_data\(\)](#).

Callable From:

◊ N/A (automatically called by DI-Guy Graphics API during Update Stage)

5.39.38 function diguyGraphicsShape::show

Prototype:

```
virtual void show();
```

Description:

This function will be called by DI-Guy when the shape should be shown, typically due to a change in the motion data that controls the switch. See [get_switch_var_name\(\)](#) and [get_switch_value\(\)](#).

Immediate Mode:

Immediate mode renderers usually do not override this function. DI-Guy does not call the [draw\(\)](#) function of hidden shapes.

Scene Graph:

Scene graph renderers usually do override this function, to update the state of the switch node created during the [build\(\)](#) call.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during Update Stage)

5.39.39 function diguyGraphicsShape::hide

Prototype:

```
virtual void hide();
```

Description:

This function will be called by DI-Guy when the shape should be hidden, typically due to a change in the motion data that controls the switch. See [get_switch_var_name\(\)](#) and [get_switch_value\(\)](#).

Immediate Mode:

Immediate mode renderers usually do not override this function. DI-Guy does not call the [draw\(\)](#) function of hidden shapes.

Scene Graph:

Scene graph renderers usually do override this function, to update the state of the switch node created during the [build\(\)](#) call.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during Update Stage)

5.39.40 function diguyGraphicsShape::draw

Prototype:

```
virtual void draw(int lod);
```

Description:

This function will be called by DI-Guy when it is time for the geometry of this shape to be drawn.

Immediate Mode:

Immediate mode renderers usually do not override this function. The actual draw calls happen in `diguyGraphicsMesh::draw()` calls.

Scene Graph:

Scene graph renderers usually do not override this function; drawing is handled automatically by scene graphs.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during Draw Stage)

5.39.41 function diguyGraphicsShape::set_graphics_lod_ranges

Prototype:

```
virtual void set_graphics_lod_ranges(float* ranges);
```

Description:

This function will be called by DI-Guy when the LOD ranges of this shape need to be updated after the shape has initially been built in the Build Stage.

Arguments:

`lod_ranges` new lod ranges for this shape

See [diguyScenario::set_default_lod_ranges\(\)](#) for a description of what will be in the `lod_ranges` argument.

Immediate Mode:

Immediate mode renderers usually do not override this function. Instead, they use the lod passed into the [draw\(\)](#) function.

Scene Graph:

Scene graph renderers can override this function, to update the ranges of LOD nodes created in the Build Stage, if any.

If there were LOD-type nodes created in the Build Stage, their ranges should be updated based on the passed values.

If no LOD-type nodes were created in the Build Stage, this function should have no effect.

Typically only one of `set_graphics_lod_ranges()` or `set_graphics_lod()` is implemented.

Callable From:

◊ N/A (automatically called by DI-Guy Graphics API when the DI-Guy API function `diguyCharacter::set_lod_ranges()` is called)

5.39.42 function diguyGraphicsShape::set_graphics_lod

Prototype:

```
virtual void set_graphics_lod(int lod);
```

Description:

This function will be called by DI-Guy when the LOD of this link is "manually" set by a `diguyCharacter::set_graphics_lod()` function call.

Immediate Mode:

Immediate mode renderers usually do not override this function. Instead, they use the lod passed into the `draw()` function.

Scene Graph:

Scene graph renderers sometimes override this function, to update the state of switch nodes created in the Build Stage.

Typically only one of `set_graphics_lod_ranges()` or `set_graphics_lod()` is implemented.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API when the DI-Guy API function `diguyCharacter::set_graphics_lod()` is called)

5.39.43 function `diguyGraphicsShape::update_variation_texture`

Prototype:

```
virtual void update_variation_texture(int num_pixels, unsigned char * pixel_data);
```

Description:

Pushes a `char *` pixel buffer of length `num_pixels` that represents the unique per shape texture. Typically a Immediate mode renderer would need to use this to create or update a texture see `diguyOglGraphicsShape.cpp` for an example. For an retained mode renderer we have `get_variation_texture()` as an accessor.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during Build Stage)

5.39.44 function diguyGraphicsShape::get_bind_color_array**Prototype:**

```
int get_bind_color_array(int* num_colors, const float** color_data);
```

Description:

Returns the unique color of the character links. All links in a DI-Guy appearance are assigned a unique RGB color that can be used in pick/selection operations. The data returned in `color_data` is an array of length `num_colors` with 4 floats per entry (r, g, b, a).

See

`programming_examplesdiguy_graphics_apiosg_2_examples/intersection_test` for an example of its usage.

Returns:

0 on success, -1 on failure

5.40 diguyGraphicsState

diguyGraphicsState.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#) | [diguyGraphicsTexture.h](#)

Contents:

- ◆ [Accessor Functions](#)
- ◆ [Virtual Functions](#)

Link against: `libdiguy_graphics_api`

5.40.1 class diguyGraphicsState

```
class BDI_DECLSPEC_diguy diguyGraphicsState
{
public:
    diguyGraphicsMaterial* get_material();
    int get_num_textures();
    diguyGraphicsTexture* get_texture(int texture_index = 0);
    diguyTextureMapType get_texture_type(int index);
    int get_textures_available();
    int get_contains_alpha_component();
    int get_is_unlit();
    void free_loader_memory();
#ifdef CPLUSPLUS_ONLY
    virtual void build();
    virtual void unbuild();
    virtual void bind_now();
#endif
#endif
```

Accessor Functions

The functions in this section provide access to data in this object. They are designed to be called by DI-Guy programmers.

5.40.2 function `diguyGraphicsState::get_material`

Prototype:

```
diguyGraphicsMaterial* get_material();
```

Description:

This function returns the material of this state. Specific information about the material can then be accessed from the returned object.

Returns:

object of type `diguyGraphicsMaterial`; NULL on failure

Callable From:

◇ C++

5.40.3 function `diguyGraphicsState::get_num_textures`

Prototype:

```
int get_num_textures();
```

Description:

This function returns the number of textures this state has.

Returns:

number of textures associated with this state

Callable From:

◇ C++

5.40.4 function `diguyGraphicsState::get_texture`

Prototype:

```
diguyGraphicsTexture* get_texture(int texture_index = 0);
```

Description:

This function returns the nth texture of this state. Specific information about the texture can then be accessed from the returned object.

Returns:

object of type `diguyGraphicsTexture`; NULL on failure

Callable From:

◇ C++

5.40.5 function `diguyGraphicsState::get_texture_type`

Prototype:

```
diguyTextureMapType get_texture_type(int index);
```

Description:

This function returns the way that a texture is used in the state. Possibilities include diffuse, specular, normal.

Returns:

`diguyTextureMapType` enum

Callable From:

◇ C++

5.40.6 function `diguyGraphicsState::get_textures_available`

Prototype:

```
int get_textures_available();
```

Description:

This function returns a bit-wise or'ed mask of available textures in this state. Possibilities include `DIGUY_TEXTURE_MAP_MASK_DIFFUSE`, `DIGUY_TEXTURE_MAP_MASK_SPECULAR`, `DIGUY_TEXTURE_MAP_MASK_BUMP`, etc. See `diguyTextureMapMasks` enum for a complete list.

5.40.7 function diguyGraphicsState::get_contains_alpha_component

Prototype:

```
int get_contains_alpha_component();
```

Description:

This function returns whether either the material or texture of this state contains an alpha component. States that have an alpha component can make objects semi-transparent, which can affect when said objects should be drawn.

Returns:

1 if state contains an alpha component, 0 if not; -1 on failure

Callable From:

◇ C++

5.40.8 function diguyGraphicsState::get_is_unlit

Prototype:

```
int get_is_unlit();
```

Description:

This function returns whether faces drawn with this state should be lit or unlit.

Returns:

1 if faces unlit, 0 if lit, -1 on failure

Callable From:

◇ C++

5.40.9 function diguyGraphicsState::free_loader_memory

Prototype:

```
void free_loader_memory();
```

Description:

This function frees most memory allocated by the DI-Guy geometry loader for this object.

This function should only be called during or after the object's `build()` function has been called. After it has been called, calls to many accessor functions will fail.

Callable From:

◇ C++

Virtual Functions

The functions in this section are all virtual functions that will be called by DI-Guy at various times during DI-Guy execution.

Because the functions are all virtual, DI-Guy programmers can create a subclass of this class and override them. Unless otherwise stated, the base class functions do not need to be called.

5.40.10 function diguyGraphicsState::build

Prototype:

```
virtual void build();
```

Description:

This function will be called by DI-Guy when it is time for a renderer-specific state object to be created. A subclass of this class isn't always necessary, particularly for immediate mode renderers in which everything might be done in a subclass of [diguyGraphicsMaterial](#) and a subclass of [diguyGraphicsTexture](#).

Immediate Mode:

Immediate mode renderers usually do not override this function.

Scene Graph:

Scene graph renderers may override this function, to create a state object that will be associated with [diguyGraphicsMesh](#) objects.

The subclass version of this function may need to refer to material and texture objects created by the [build\(\)](#) calls of subclasses of [diguyGraphicsMaterial](#) and [diguyGraphicsTexture](#) objects.

Callable From:

◊ N/A (automatically called by DI-Guy Graphics API during the Build Stage)

5.40.11 function diguyGraphicsState::unbuild

Prototype:

```
virtual void unbuild();
```

Description:

This function will be called by DI-Guy when it is time for a renderer-specific state object to be destroyed.

Immediate Mode:

Immediate mode renderers may override this function, to destroy any object that may have been created by `build()`.

Scene Graph:

Scene graph renderers may override this function, to destroy any object that may have been created by `build()`.

Callable From:

◊ N/A (automatically called by DI-Guy Graphics API during the Unbuild Stage)

5.40.12 function diguyGraphicsState::bind_now

Prototype:

```
virtual void bind_now();
```

Description:

This function will be called by DI-Guy when the state should be made active, or bound.

Immediate Mode:

Immediate mode renderers can override this function, but may not need to. The material and texture of the state will receive their own `bind_now()` calls.

Scene Graph:

Scene graph renderers usually do not override this function.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during Draw Stage)

5.41 diguyGraphicsTexture

diguyGraphicsTexture.h

DI-Guy SDK Version 12.5.2

Includes: `declspec_diguy.h` | [diguy_constants.h](#)

Contents:

- ◆ [Accessor Functions](#)
- ◆ [Virtual Functions](#)

Link against: `libdiguy_graphics_api`

5.41.1 enumeration diguyGraphicsTextureCompressionType

```
typedef enum
{
    DIGUY_GRAPHICS_TEXTURE_COMPRESSION_TYPE_NONE = 0,
    DIGUY_GRAPHICS_TEXTURE_COMPRESSION_TYPE_DXT1,
    DIGUY_GRAPHICS_TEXTURE_COMPRESSION_TYPE_DXT2,
    DIGUY_GRAPHICS_TEXTURE_COMPRESSION_TYPE_DXT3,
    DIGUY_GRAPHICS_TEXTURE_COMPRESSION_TYPE_DXT4,
    DIGUY_GRAPHICS_TEXTURE_COMPRESSION_TYPE_DXT5,
    DIGUY_GRAPHICS_TEXTURE_COMPRESSION_TYPE_DXT5_NM
} diguyGraphicsTextureCompressionType;
```

Description:

This enumeration lists the types of compression that texture map data may have.

5.41.2 enumeration diguyTextureMapType

```
typedef enum
{
    DIGUY_TEXTURE_MAP_TYPE_UNKNOWN = -1,
    DIGUY_TEXTURE_MAP_TYPE_DIFFUSE = 0,
    DIGUY_TEXTURE_MAP_TYPE_SHADOW,
    DIGUY_TEXTURE_MAP_TYPE_SPECULAR,
    DIGUY_TEXTURE_MAP_TYPE_BUMP,
    DIGUY_TEXTURE_MAP_TYPE_COMPONENTS,
    DIGUY_TEXTURE_MAP_TYPE_COMPONENT_RAMP,
    DIGUY_TEXTURE_MAP_TYPE_MISC_1,
    DIGUY_TEXTURE_MAP_TYPE_MISC_2
} diguyTextureMapType;
```

Description:

This enumeration lists the type of texture map a given image file is being used as.

5.41.3 enumeration diguyTextureMapMasks

```
typedef enum
{
    DIGUY_TEXTURE_MAP_MASK_DIFFUSE      = 0x00000001,
    DIGUY_TEXTURE_MAP_MASK_SHADOW       = 0x00000002,
    DIGUY_TEXTURE_MAP_MASK_SPECULAR     = 0x00000004,
    DIGUY_TEXTURE_MAP_MASK_BUMP         = 0x00000008,
    DIGUY_TEXTURE_MAP_MASK_COMPONENTS   = 0x00000010,
    DIGUY_TEXTURE_MAP_MASK_MISC_1       = 0x00000020,
    DIGUY_TEXTURE_MAP_MASK_MISC_2       = 0x00000040,
    DIGUY_TEXTURES_MAP_MASK_ALL          = 0xffffffff
} diguyTextureMapMasks;
```

Description:

This enumeration lists how the state stores the available textures in the textures_available bitfield. Also used by the shader programs.

5.41.4 class diguyGraphicsTexture

Description:

This function converts the passed C-style string into an enumerated value.

Returns:

enumeration value of string

Callable From:

- ◆ C++
- ◆ Script

```
class BDI_DECLSPEC diguy diguyGraphicsTexture
{
public:
    const char* get_name();
    const char* get_filename();
    int get_width();
    int get_height();
    int get_num_components();
    int get_contains_alpha_component();
#ifdef CPLUSPLUS_ONLY
    unsigned char* get_texture_map_data();
#endif
    int get_texture_map_data_size();
    diguyGraphicsTextureCompressionType get_texture_compression_type();
    int get_num_included_mipmaps();
    int get_mipmap_data_size(int mipmap);
    int get_mipmap_data_offset(int mipmap);
    int get_gl_format();
    const char* get_gl_format_as_string();
    int get_dds_block_size();
    int get_clamp_s();
    int get_clamp_t();
    void free_loader_memory();
#ifdef CPLUSPLUS_ONLY
    virtual void build();
    virtual void unbuild();
    virtual void bind_now(diguyTextureMapType texture_type);
    static diguyGraphicsTexture* get_last_bound_texture(int texture_unit = 0);
    static void clear_last_bound_texture();
#endif
};
```


Accessor Functions

The functions in this section provide access to data in this object. They are designed to be called by DI-Guy programmers.

5.41.5 function diguyGraphicsTexture::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the texture. This pointer will never be NULL.

Returns:

name of the texture

Callable From:

◇ C++

5.41.6 function diguyGraphicsTexture::get_filename

Prototype:

```
const char* get_filename();
```

Description:

This function returns the filename of the texture. This pointer may be NULL if the texture does not have an associated filename.

Returns:

filename of the texture

Callable From:

◇ C++

5.41.7 function diguyGraphicsTexture::get_width

Prototype:

```
int get_width();
```

Returns:

the width of the texture, in pixels; -1 on failure

Callable From:

◇ C++

5.41.8 function diguyGraphicsTexture::get_height**Prototype:**

```
int get_height();
```

Returns:

the height of the texture, in pixels; -1 on failure

Callable From:

◇ C++

5.41.9 function diguyGraphicsTexture::get_num_components**Prototype:**

```
int get_num_components();
```

Description:

This function returns how many components the texture has. The number of components will be an integer between 1 and 4.

Most textures have three components, red (R), green (G), and blue (B), or RGB. These textures commonly have filenames with the extension .rgb.

Four component textures add an alpha (A) channel, resulting in RGBA. These textures commonly have filenames with the extension .rgba.

One component textures have only an intensity (I). These textures commonly have filenames with the extension .int.

Two component textures have only an intensity (I), and an alpha channel. These textures commonly have filenames with the extension .inta.

Returns:

number of components in texture; -1 on failure

Callable From:

◇ C++

5.41.10 function diguyGraphicsTexture::get_contains_alpha_component

Prototype:

```
int get_contains_alpha_component();
```

Description:

This function returns whether the texture contains an alpha component. Textures that have an alpha component can make objects semi-transparent, which can affect when said objects should be drawn.

Returns:

1 if texture contains an alpha component, 0 if not; -1 on failure

Callable From:

◇ C++

5.41.11 function diguyGraphicsTexture::get_texture_map_data_size

Prototype:

```
int get_texture_map_data_size();
```

Description:

This function returns how many bytes are used by the texture map data buffer.

Returns:

buffer size of texture map data, in bytes; may be 0 in the case of a NULL texture

Callable From:

◇ C++

5.41.12 function diguyGraphicsTexture::get_texture_compression_type

Prototype:

```
diguyGraphicsTextureCompressionType get_texture_compression_type();
```

Description:

This function returns which compression method, if any, is applied to the data returned by `get_texture_map_data()`.

Most often compressed textures come from .dds (DirectDraw Surface) files.

Returns:

compression type enumeration

Callable From:

◇ C++

5.41.13 function `diguyGraphicsTexture::get_num_included_mipmaps`

Prototype:

```
int get_num_included_mipmaps();
```

Description:

This function returns how many mipmaps are already included in the texture map data.

Typically only .dds (DirectDraw Surface) textures have precomputed mipmaps in them.

Note that values 0 and 1 are somewhat the same: there is only one image in the texture map data: the base, unscaled image.

Returns:

num of included mipmaps

Callable From:

◇ C++

5.41.14 function diguyGraphicsTexture::get_mipmap_data_size**Prototype:**

```
int get_mipmap_data_size(int mipmap);
```

Description:

This function returns the size of the data, in bytes, of the specified mipmap.

"mipmap" 0 is the base, unscaled image, mipmap 1 is the first scaled image, etc.

Returns:

data size, in bytes, of mipmap; -1 on error

Callable From:

◇ C++

5.41.15 function diguyGraphicsTexture::get_mipmap_data_offset**Prototype:**

```
int get_mipmap_data_offset(int mipmap);
```

Description:

This function returns the data offset of the specified mipmap into the data pointer returned by `get_texture_map_data()`.

"mipmap" 0 is the base, unscaled image, mipmap 1 is the first scaled image, etc.

Returns:

data offset, in bytes, of mipmap; -1 on error

Callable From:

◇ C++

5.41.16 function diguyGraphicsTexture::get_gl_format**Prototype:**

```
int get_gl_format();
```

Description:

This function returns the OpenGL pixel format enumeration of this texture.

This enumeration will be one of the following:

- ◇ GL_LUMINANCE
- ◇ GL_LUMINANCE_ALPHA
- ◇ GL_RGB
- ◇ GL_RGBA
- ◇ GL_COMPRESSED_RGB_S3TC_DXT1_EXT
- ◇ GL_COMPRESSED_RGBA_S3TC_DXT1_EXT
- ◇ GL_COMPRESSED_RGBA_S3TC_DXT5_EXT

Returns:

OpenGL pixel format enum; -1 on error

In OpenGL and OSG environments, the returned value should be cast into a GLenum.

Callable From:

- ◇ C++

5.41.17 function diguyGraphicsTexture::get_gl_format_as_string**Prototype:**

```
const char* get_gl_format_as_string();
```

Description:

This function returns the OpenGL pixel format string of this texture. There are a large number of potential results depending if compressed textures and/or sRGB textures are enabled.

Example of possible results:

```

◇ GL_RGB GL_RGBA
◇ sRGB: GL_SRGB GL_SRGB_ALPHA
◇ Compressed: GL_COMPRESSED_RGB_S3TC_DXT1
               GL_COMPRESSED_RGBA_S3TC_DXT5
◇ sRGB + Compressed:
               GL_COMPRESSED_RGB_ALPHA_S3TC_DXT1
               GL_COMPRESSED_SRGB_ALPHA_S3TC_DXT5
```

5.41.18 function diguyGraphicsTexture::get_dds_block_size**Prototype:**

```
int get_dds_block_size();
```

Description:

This function returns the "block size" of the dds compression types. This value can be useful in generating mipmap data offsets.

Alternatively, the function [get_mipmap_data_offset\(\)](#) can be called to get the precomputed mipmap data offsets.

Currently this value will be either 8 (for DXT1 compression) or 16 (for DXT3 and DXT5 compression).

Returns:

block size, in bytes

Callable From:

◇ C++

5.41.19 function diguyGraphicsTexture::get_clamp_s**Prototype:**

```
int get_clamp_s();
```

Description:

For texture indices less than zero and greater than one, a decision must be made as to whether the texture is repeated or whether the edge pixels are replicated to fill the space.

This function returns whether the texture should be repeated horizontally (don't clamp) or whether the edge column of pixels should be used (do clamp).

Note that in MultiGen Creator attribute (.attr) files, this corresponds to the "Wrap Method u" attribute.

Returns:

0 is texture should be tiled, 1 if not; -1 on failure

Callable From:

◇ C++

5.41.20 function diguyGraphicsTexture::get_clamp_t**Prototype:**

```
int get_clamp_t();
```

Description:

For texture indices less than zero and greater than one, a decision must be made as to whether the texture is repeated or whether the edge pixels are replicated to fill the space.

This function returns whether the texture should be repeated vertically (don't clamp) or whether the edge row of pixels should be used (do clamp).

Note that in MultiGen Creator attribute (.attr) files, this corresponds to the "Wrap Method v" attribute.

Returns:

0 is texture should be repeated, 1 if not; -1 on failure

Callable From:

◇ C++

5.41.21 function diguyGraphicsTexture::free_loader_memory

Prototype:

```
void free_loader_memory();
```

Description:

This function frees most memory allocated by the DI-Guy geometry loader for this object.

This function should only be called during or after the object's `build()` function has been called. After it has been called, calls to many accessor functions will fail.

Callable From:

◇ C++

Virtual Functions

The functions in this section are all virtual functions that will be called by DI-Guy at various times during DI-Guy execution.

Because the functions are all virtual, DI-Guy programmers can create a subclass of this class and override them. Unless otherwise stated, the base class functions do not need to be called.

5.41.22 function diguyGraphicsTexture::build

Prototype:

```
virtual void build();
```

Description:

This function will be called by DI-Guy when it is time for a renderer-specific texture object to be created. At that point the texture information has been read from the texture file, and is available via the Accessor Functions above.

Immediate Mode:

Immediate mode renderers may override this function, to create an object that may be invoked by `bind_now()`.

Scene Graph:

Scene graph renderers usually do override this function, to create an object that gets associated with `diguyGraphicsMesh` objects.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during the Build Stage)

5.41.23 function diguyGraphicsTexture::unbuild

Prototype:

```
virtual void unbuild();
```

Description:

This function will be called by DI-Guy when it is time for a renderer-specific texture object to be destroyed.

Immediate Mode:

Immediate mode renderers may override this function, to destroy any object that may have been created by `build( )`.

Scene Graph:

Scene graph renderers usually do override this function, to destroy any object that may have been created by `build( )`.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during the Unbuild Stage)

5.41.24 function diguyGraphicsTexture::bind_now

Prototype:

```
virtual void bind_now(diguyTextureMapType texture_type);
```

Description:

This function will be called by DI-Guy when the texture should be made active, or bound.

Immediate Mode:

Immediate mode renderers usually do override this function, to invoke the object created by `build( )`, or make calls to the renderer to set appropriate texture state.

Scene Graph:

Scene graph renderers usually do not override this function.

Callable From:

◇ N/A (automatically called by DI-Guy Graphics API during Draw Stage)

5.41.25 function diguyGraphicsTexture::get_last_bound_texture

Prototype:

```
static diguyGraphicsTexture* get_last_bound_texture(int texture_unit = 0);
```

Description:

This function returns the last bound texture used by DI-Guy. This value can be used in `diguyGraphicsTexture::bind( )` to check to see if a texture rebind is necessary; avoiding a texture rebind can avoid a significant performance hit in some rendering systems.

Callable From:

◇ C++

5.41.26 function `diguyGraphicsTexture::clear_last_bound_texture`

Prototype:

```
static void clear_last_bound_texture();
```

Description:

This function clears out the last bound texture variable.
`diguyScenario::draw()` automatically calls it, but it is also recommended that it be called before drawing individual `diguyCharacters` in case something outside of DI-Guy has changed the texture.

Callable From:

◇ C++

5.42 diguyGraphicsVertexFormats

diguyGraphicsVertexFormats.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

Link against: [libdiguy_graphics_api](#)

5.42.1 enumeration diguyGraphicsVertexFormat

```
typedef enum
{
    DIGUY_GRAPHICS_VERTEX_FORMAT_UNKNOWN           = -1,
    DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2         = 0,
    DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_ARRAYS  = 1,
    DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_I4F_W4  = 2,
    DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_I4F_W4_ARRAYS = 3,
    DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_T4_I4F_W4 = 4,
    DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_T4      = 5
} diguyGraphicsVertexFormat;
```

Description:

This enumeration lists types of vertex data formats DI-Guy may use. In general the same vertex format meshes will use cannot be the same for all meshes, as static meshes don't use indices and weights, but dynamic meshes (e.g., skinned meshes) do.

The data member `use_array_format_for_vertex_buffer_data` of the [bdiGraphicsInitGraphicsAPI](#) structure can be used to specify whether interleaved or array format is to be used.

See [diguyGraphicsMesh::get_vertex_format\(\)](#).

DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2

The most basic vertex format DI-Guy supports: One vertex position, one normal, and one UV texture coordinate. These values are interleaved like so:

vnt vnt vnt vnt ...

Where 'v' represents three floats specifying vertex position, 'n' represents three floats specifying normal direction, and 't' represents two floats specifying texture indices.

This format is commonly used by static meshes. It cannot be used by skinned meshes.

DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_ARRAYS

This vertex format puts all vertices first, followed by all normals, followed by all texture indices:

This format is commonly used by static meshes. It cannot be used by skinned meshes.

DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_I4F_W4

Vertices for skinned meshes require more info; specifically, they need the indices of the links/bones that influence them and the weights of said influences. DI-Guy supports at most 4 link influences per vertex.

Indices are stored as floats instead of an int or unsigned int because data passed to shaders in attribute variables (at least for glsl shaders) needs to be in float form, not int.

These values are interleaved like so:

Where 'i' and 'w' each represent four floats specifying indices and weights, respectively.

This format is commonly used by skinned meshes. It could be used by static meshes, but doing so would waste memory.

DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_I4F_W4_ARRAYS

Similar to [DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_I4F_W4](#), but values are not interleaved:

This format is commonly used by skinned meshes. It could be used by static meshes, but doing so would waste memory.

5.42.2 enumeration `diguyGraphicsIndexType`

```
typedef enum
{
    DIGUY_GRAPHICS_INDEX_TYPE_UNKNOWN    = -1,
    DIGUY_GRAPHICS_INDEX_TYPE_UINT      = 0,
    DIGUY_GRAPHICS_INDEX_TYPE_USHORT    = 1,
    DIGUY_GRAPHICS_INDEX_TYPE_D3D_WORD  = 2
} diguyGraphicsIndexType;
```

Description:

This enumeration lists the types that indices stored in index buffer data might have.

The data member `default_vertex_type` of the `bdiGraphicsInitGraphicsAPI` structure can be used to specify which index type is needed.

Also see `diguyGraphicsMesh::get_index_type()`.

Note that indices in index buffer data are different than the indices contained in vertex buffer data. Index buffer indices specify which **vertices** in vertex buffer data are used to create polygons in a mesh (e.g., for the first triangle, use vertices 0, 1, and 2), while indices stored in vertex buffer data specify which **links** affect the final position of the vertex.

DIGUY_GRAPHICS_INDEX_TYPE_UINT

Index buffer data is stored as unsigned integers (4 bytes).

DIGUY_GRAPHICS_INDEX_TYPE_USHORT

Index buffer data is stored as unsigned short integers (2 bytes).

DIGUY_GRAPHICS_INDEX_TYPE_D3D_WORD

Index buffer data is stored as Windows WORD type. This is a 32 bit unsigned integer, so it's really the same as `DIGUY_GRAPHICS_INDEX_TYPE_UINT`.

5.43 diguyIGuyController

diguyIGuyController.h

DI-Guy SDK Version 12.5.2

Includes: [declspec_diguy.h](#) | [diguyMotionPosture.h](#) | [diguyMotionVariant.h](#)

Contents:

- ◆ [Character Functions](#)
- ◆ [Interaction Functions](#)
- ◆ [Camera Functions](#)
- ◆ [Mouse Functions](#)
- ◆ [Keyboard Functions](#)
- ◆ [Joystick Functions](#)
- ◆ [Posture and Variant Functions](#)
- ◆ [Misc Enable and Disable Functions](#)
- ◆ [Performance Functions](#)
- ◆ [Misc Functions](#)
- ◆ [Facing Functions](#)
- ◆ [Deprecated Functions](#)

5.43.1 class diguyIGuyController

```
class BDI_DECLSPEC_diguy diguyIGuyController
{
public:
    void set_iguy_character(const char* name,
        diguyMotionPosture initial_posture = DIGUY_MOTION_POSTURE_ANY,
        diguyMotionVariant initial_variant = DIGUY_MOTION_VARIANT_ANY);
    diguyCharacter* get_iguy_character();
    diguyCharacter* get_interaction_subject();
    void set_camera_change_enabled(int enabled);
    int get_camera_change_enabled();
    void set_camera_use_1st_person_flag(int flag);
    int get_camera_use_1st_person_flag();
    void set_camera_attachment_link(const char* attachment_link);
    const char* get_camera_attachment_link();
    void set_camera_3rd_person_look_from_offset(float x, float y, float z);
    void get_camera_3rd_person_look_from_offset(float* x, float* y, float* z);
    void set_camera_3rd_person_look_at_offset(float x, float y, float z);
    void get_camera_3rd_person_look_at_offset(float* x, float* y, float* z);
    void set_mouse_input_enabled(int enabled);
    int get_mouse_input_enabled();
    void set_input_capture_enabled(int enabled);
    int get_input_capture_enabled();
    void set_keyboard_input_enabled(int enabled);
    int get_keyboard_input_enabled();
    void set_joystick_input_enabled(int enabled);
    int get_joystick_input_enabled();
    void set_posture_change_enabled(int enabled);
    int get_posture_change_enabled();
```

```

void set_posture(diguyMotionPosture posture);
diguyMotionPosture get_posture();
void set_variant_change_enabled(int enabled);
int get_variant_change_enabled();
void set_variant(diguyMotionVariant variant);
diguyMotionVariant get_variant();
void set_gaze_change_enabled(int enabled);
int get_gaze_change_enabled();
void set_orientation_change_enabled(int enabled);
int get_orientation_change_enabled();
void set_position_change_enabled(int enabled);
int get_position_change_enabled();
void set_use_force_action_flag(int use_force_action_flag);
int get_use_force_action_flag();
void set_skip_transitions_flag(int skip_transitions_flag);
int get_skip_transitions_flag();
void set_always_run_flag(int always_run_flag);
int get_always_run_flag();
float get_face_rz();
float get_face_ry();
void set_face_rz_desired(float face_rz_desired);
float get_face_rz_desired();
void set_face_ry_desired(float face_ry_desired);
float get_face_ry_desired();
void face_character(const char* character_name,
    const char* link,
    float offset_rz = 0.0f,
    float offset_ry = 0.0f);
void set_mouse_look_changes_orientation(int val);
int get_mouse_look_changes_orientation();

```

Character Functions

5.43.2 function diguyIGuyController::set_iguy_character

Prototype:

```
void set_iguy_character(const char* name,
    diguyMotionPosture initial_posture = DIGUY_MOTION_POSTURE_ANY,
    diguyMotionVariant initial_variant = DIGUY_MOTION_VARIANT_ANY);
```

Description:

This function sets the interactive character (I-Guy).

The I-Guy character's actions, position, and orientation can be 'driven' by input from the mouse, keyboard, or a joystick.

There can be at most one I-Guy at a time.

Arguments:

<code>name</code>	name of character to become I-Guy
<code>initial_posture</code>	this argument is not currently used, but will be implemented in a future version of DI-Guy
<code>initial_variant</code>	the initial motion variant the I-Guy character should assume

Passing `DIGUY_MOTION_VARIANT_ANY` as the initial variant will cause DI-Guy to select a default initial variant. Passing `DIGUY_MOTION_VARIANT_NORMAL` will cause the character to start in a non-aiming action. Passing `DIGUY_MOTION_VARIANT_AIM` will cause the character to start in an aiming action, if available. Other variants will be supported in a future version.

Callable From:

- ◇ C++
- ◇ Script

5.43.3 function diguyIGuyController::get_iguy_character

Prototype:

```
diguyCharacter* get_iguy_character();
```

Returns:

pointer of type diguyCharacter* to the current I-Guy character

Callable From:

- ◇ C++
- ◇ Script

Interaction Functions

5.43.4 function diguyIGuyController::get_interaction_subject

Prototype:

```
diguyCharacter* get_interaction_subject();
```

Description:

This function returns the character clicked on when in I-Guy Input Mode. The pointer is valid during CALLBACK_ID_IGUY_INTERACT_WITH_SUBJECT callbacks.

Returns:

pointer to type [diguyCharacter](#); NULL if there is no interaction subject

Callable From:

- ◇ C++
- ◇ Script

Camera Functions

5.43.5 function diguyIGuyController::set_camera_change_enabled

Prototype:

```
void set_camera_change_enabled(int enabled);
```

Description:

This function sets whether the I-Guy controller should update the primary view's camera to keep the I-Guy in view.

Arguments:

`enabled` pass 1 to enable camera changes, 0 to disable

Callable From:

- ◇ C++
- ◇ Script

5.43.6 function diguyIGuyController::get_camera_change_enabled

Prototype:

```
int get_camera_change_enabled();
```

Returns:

1 if I-Guy affects camera, 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.43.7 function diguyIGuyController::set_camera_use_1st_person_flag

Prototype:

```
void set_camera_use_1st_person_flag(int flag);
```

Description:

This function sets whether the I-Guy controlled camera should be 1st person (from-the-eyes view), or 3rd person (over the shoulder).

Arguments:

flag	pass 1 for 1st person, 0 for 3rd

Callable From:

- ◇ C++
- ◇ Script

5.43.8 function diguyIGuyController::get_camera_use_1st_person_flag

Prototype:

```
int get_camera_use_1st_person_flag();
```

Returns:

1 if I-Guy controlled camera is 1st person, 0 if 3rd

Callable From:

- ◇ C++
- ◇ Script

5.43.9 function

diguyIGuyController::set_camera_attachment_link

Prototype:

```
void set_camera_attachment_link(const char* attachment_link);
```

Description:

This function sets which link the I-Guy controlled camera will be attached to. For human characters, the best link is typically "cervical" (the neck), or NULL (to use the base position of the character). The default is "cervical".

This setting affects both 1st person and 3rd person cameras.

Arguments:

`attachment_link` name of the link to attach to

Pass NULL to have the camera be attached to the I-Guy character's base position.

Callable From:

- ◇ C++
- ◇ Script

5.43.10 function diguyIGuyController::get_camera_attachment_link

Prototype:

```
const char* get_camera_attachment_link();
```

Returns:

the name of the current camera attachment link, or the empty string ("") if there is none

Callable From:

- ◇ C++
- ◇ Script

5.43.11 function diguyIGuyController::set_camera_3rd_person_look_from_offset

Prototype:

```
void set_camera_3rd_person_look_from_offset(float x, float y, float z);
```

Description:

This function sets the offset that is used from the attachment link's position for the 3rd person camera's position.

Typically this has a slightly negative x value, to move the camera behind the I-Guy character, and a positive z value, to move the camera up so it can see over the character's head.

The default is x=-2.0, y=0.0f, z=0.35.

This setting affects only 3rd person cameras.

Arguments:

`x, y, z` offset of camera position from attachment link

Callable From:

- ◇ C++
- ◇ Script

5.43.12 function `diguyIGuyController::get_camera_3rd_person_look_from_offset`

Prototype:

```
void get_camera_3rd_person_look_from_offset(float* x, float* y, float* z);
```

Description:

This function retrieves the offset that is used from the attachment link's position for the 3rd person camera's position. The offset values should be put into the variables pointed to.

Arguments:

`x, y, z` pointers to variables in which offset should be returned

Callable From:

- ◇ C++
- ◇ Script

5.43.13 function**diguyIGuyController::set_camera_3rd_person_look_at_offset****Prototype:**

```
void set_camera_3rd_person_look_at_offset(float x, float y, float z);
```

Description:

This function is similar to [set_camera_3rd_person_look_from_offset\(\)](#), but affects the look-at point instead.

The default is x=0.0, y=0.0, z=0.3.

Arguments:

x, y, z offset of camera look-at point from attachment link

Callable From:

◇ C++
◇ Script

5.43.14 function**diguyIGuyController::get_camera_3rd_person_look_at_offset****Prototype:**

```
void get_camera_3rd_person_look_at_offset(float* x, float* y, float* z);
```

Description:

This function is similar to [get_camera_3rd_person_look_from_offset\(\)](#), but affects the look-at point instead.

Arguments:

x, y, z pointers to variables in which offset should be returned

Callable From:

◇ C++
◇ Script

Mouse Functions

5.43.15 function diguyIGuyController::set_mouse_input_enabled

Prototype:

```
void set_mouse_input_enabled(int enabled);
```

Description:

This function sets whether mouse input will affect the I-Guy character.

Arguments:

`enabled` pass 1 to enable mouse input, 0 to disable

Callable From:

- ◇ C++
- ◇ Script

5.43.16 function diguyIGuyController::get_mouse_input_enabled**Prototype:**

```
int get_mouse_input_enabled();
```

Returns:

1 if mouse input is enabled, 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.43.17 function diguyIGuyController::set_input_capture_enabled**Prototype:**

```
void set_input_capture_enabled(int enabled);
```

Description:

This function sets whether mouse input should be "captured".

A captured mouse's cursor doesn't move, but stays in the middle of the view. This mode is common in first-person shooter (FPS) style games.

A non-captured mouse's cursor is free to move anywhere on the screen.

The setting only has an effect in DI-Guy Scenario.

Arguments:

`enabled` pass 1 to enable input capture, 0 to disable

Callable From:

- ◇ C++
- ◇ Script

5.43.18 function `diguylGuyController::get_input_capture_enabled`

Prototype:

```
int get_input_capture_enabled();
```

Returns:

1 if input is captured, 0 if not

Callable From:

- ◇ C++
- ◇ Script

Keyboard Functions

5.43.19 function diguyIGuyController::set_keyboard_input_enabled

Prototype:

```
void set_keyboard_input_enabled(int enabled);
```

Description:

This function sets whether keyboard input will affect the I-Guy character.

Arguments:

`enabled` pass 1 to enable keyboard input, 0 to disable

Callable From:

- ◇ C++
- ◇ Script

5.43.20 function diguyIGuyController::get_keyboard_input_enabled

Prototype:

```
int get_keyboard_input_enabled();
```

Returns:

1 if keyboard input is enabled, 0 if not

Callable From:

- ◇ C++
- ◇ Script

Joystick Functions

5.43.21 function diguyIGuyController::set_joystick_input_enabled

Prototype:

```
void set_joystick_input_enabled(int enabled);
```

Description:

This function sets whether joystick input will affect the I-Guy character.

Arguments:

enabled pass 1 to enable joystick input, 0 to disable

Callable From:

- ◇ C++
- ◇ Script

5.43.22 function diguyIGuyController::get_joystick_input_enabled

Prototype:

```
int get_joystick_input_enabled();
```

Returns:

1 if joystick input is enabled, 0 if not

Callable From:

- ◇ C++
- ◇ Script

Posture and Variant Functions

5.43.23 function diguyIGuyController::set_posture_change_enabled

Prototype:

```
void set_posture_change_enabled(int enabled);
```

Description:

This function sets whether the I-Guy controller can affect the posture of the character.

The default is 1, posture change enabled.

Arguments:

`enabled` pass 1 to enable posture changes, 0 to disable

Callable From:

- ◇ C++
- ◇ Script

5.43.24 function diguylGuyController::get_posture_change_enabled

Prototype:

```
int get_posture_change_enabled();
```

Returns:

1 if posture changes enabled, 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.43.25 function diguylGuyController::set_posture

Prototype:

```
void set_posture(diguyMotionPosture posture);
```

Description:

This function sets the desired posture of the I-Guy character.

Note that the character may not be able to maintain this posture. For example, if the desired posture is prone but the character type has no moving prone actions, a crouched action may be used for moving instead.

This function will have no effect if posture changes are disabled, as set by [set_posture_change_enabled\(\)](#).

Arguments:

posture	desired posture

Callable From:

◇ C++
◇ Script

5.43.26 function diguyIGuyController::get_posture

Prototype:

```
diguyMotionPosture get_posture();
```

Returns:

the desired posture of the I-Guy character

Callable From:

◇ C++
◇ Script

5.43.27 function diguyIGuyController::set_variant_change_enabled

Prototype:

```
void set_variant_change_enabled(int enabled);
```

Description:

This function sets whether the I-Guy controller can affect the variant of the character.

The default is 1, variant change enabled.

Arguments:

enabled pass 1 to enable variant changes, 0 to disable

Callable From:

- ◇ C++
- ◇ Script

5.43.28 function diguyIGuyController::get_variant_change_enabled

Prototype:

```
int get_variant_change_enabled();
```

Returns:

1 if variant changes enabled, 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.43.29 function diguyIGuyController::set_variant**Prototype:**

```
void set_variant(diguyMotionVariant variant);
```

Description:

This function sets the desired variant of the I-Guy character.

This function will have no effect if variant changes are disabled, as set by [set_variant_change_enabled\(\)](#).

Most often the variant of an I-Guy character should be one of:

- ◇ [DIGUY_MOTION_VARIANT_NORMAL](#)
- ◇ [DIGUY_MOTION_VARIANT_AIM](#)

Arguments:

<code>variant</code>	desired variant

Callable From:

- ◇ C++
- ◇ Script

5.43.30 function diguyIGuyController::get_variant**Prototype:**

```
diguyMotionVariant get_variant();
```

Returns:

the desired variant of the I-Guy character

Callable From:

- ◇ C++
- ◇ Script

Misc Enable and Disable Functions

5.43.31 function diguyIGuyController::set_gaze_change_enabled**Prototype:**

```
void set_gaze_change_enabled(int enabled);
```

Description:

This function sets whether the I-Guy controller can affect the gaze state of the character.

The default is 1, gaze change enabled.

Arguments:

`enabled` pass 1 to enable gaze changes, 0 to disable

Callable From:

- ◇ C++
- ◇ Script

5.43.32 function diguyIGuyController::get_gaze_change_enabled**Prototype:**

```
int get_gaze_change_enabled();
```

Returns:

1 if gaze changes enabled, 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.43.33 function diguyIGuyController::set_orientation_change_enabled

Prototype:

```
void set_orientation_change_enabled(int enabled);
```

Description:

This function sets whether the I-Guy controller can affect the orientation of the character.

The default is 1, orientation change enabled.

Arguments:

`enabled` pass 1 to enable orientation changes, 0 to disable

Callable From:

- ◇ C++
- ◇ Script

5.43.34 function diguyIGuyController::get_orientation_change_enabled

Prototype:

```
int get_orientation_change_enabled();
```

Returns:

1 if gaze orientation enabled, 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.43.35 function diguyIGuyController::set_position_change_enabled

Prototype:

```
void set_position_change_enabled(int enabled);
```

Description:

This function sets whether the I-Guy controller can affect the position of the character.

The default is 1, position change enabled.

Arguments:

`enabled` pass 1 to enable position changes, 0 to disable

Callable From:

- ◇ C++
- ◇ Script

5.43.36 function diguyIGuyController::get_position_change_enabled

Prototype:

```
int get_position_change_enabled();
```

Returns:

1 if position changes enabled, 0 if not

Callable From:

- ◇ C++
- ◇ Script

Performance Functions

5.43.37 function diguyIGuyController::set_use_force_action_flag

Prototype:

```
void set_use_force_action_flag(int use_force_action_flag);
```

Description:

This function sets whether the I-Guy controller will change the controlled character's action abruptly or allow action changes to take place smoothly.

Forced action changes provide better responsiveness, but don't look as nice.

See the functions [diguyCharacter::force_action\(\)](#) and [diguyCharacter::set_desired_action\(\)](#).

The [set_skip_transitions_flag\(\)](#) also has an effect on responsiveness and how nice the action transitions look.

The default is 1, force actions.

Arguments:

`use_force_action_flag` pass 1 to force quick action changes, 0 to smooth action changes

Callable From:

- ◇ C++
- ◇ Script

5.43.38 function `diguyIGuyController::get_use_force_action_flag`

Prototype:

```
int get_use_force_action_flag();
```

Returns:

whether actions are forced (returns 1) or not (returns 0)

Callable From:

- ◇ C++
- ◇ Script

5.43.39 function diguyIGuyController::set_skip_transitions_flag**Prototype:**

```
void set_skip_transitions_flag(int skip_transitions_flag);
```

Description:

This function sets whether the I-Guy controller will, when new actions are forced, the transition motions from the current to the new action are skipped.

Skipping transitions provides better responsiveness, but doesn't look as nice.

This setting only has an effect if the force action flag is set to 1 (see [set_use_force_action_flag\(\)](#)).

The default is 1, skip transitions.

Arguments:

`skip_transitions_flag` pass 1 to skip transition motions, 0 to play them

Callable From:

- ◇ C++
- ◇ Script

5.43.40 function diguyIGuyController::get_skip_transitions_flag**Prototype:**

```
int get_skip_transitions_flag();
```

Returns:

whether transitions between actions are skipped (returns 1) or not (returns 0)

Callable From:

- ◇ C++
- ◇ Script

Misc Functions

5.43.41 function diguyIGuyController::set_always_run_flag**Prototype:**

```
void set_always_run_flag(int always_run_flag);
```

Description:

This function sets whether the I-Guy character will always try to run, or move at its top speed.

The default is 0, don't always run.

Arguments:

`always_run_flag` pass 1 to always run, 0 to be able to walk and/or jog when moving slower

Callable From:

- ◇ C++
- ◇ Script

5.43.42 function diguyIGuyController::get_always_run_flag**Prototype:**

```
int get_always_run_flag();
```

Returns:

whether the I-Guy character will always try to run (returns 1), or not (returns 0)

Callable From:

- ◇ C++
- ◇ Script

Facing Functions

5.43.43 function diguyIGuyController::get_face_rz

Prototype:

```
float get_face_rz();
```

Description:

This function returns the current azimuth (left/right) facing angle of the I-Guy character. This value will change as the character is moved around by the I-Guy controller.

Returns:

current rz facing angle, in degrees

Callable From:

- ◇ C++
- ◇ Script

5.43.44 function diguyIGuyController::get_face_ry**Prototype:**

```
float get_face_ry();
```

Description:

This function returns the current elevation (down/up) facing angle of the I-Guy character. This value will change as the character is moved around by the I-Guy controller.

Returns:

current ry facing angle, in degrees

Callable From:

- ◇ C++
- ◇ Script

5.43.45 function diguyIGuyController::set_face_rz_desired**Prototype:**

```
void set_face_rz_desired(float face_rz_desired);
```

Description:

This function sets the desired azimiuth (left/right) facing angle of the I-Guy character.

Most often this value is changed via mouse, keyboard, or joystick input, not by this function.

Callable From:

- ◇ C++
- ◇ Script

5.43.46 function diguyIGuyController::get_face_rz_desired**Prototype:**

```
float get_face_rz_desired();
```

Returns:

desired rz facing angle, in degrees

Callable From:

- ◇ C++
- ◇ Script

5.43.47 function diguyIGuyController::set_face_ry_desired**Prototype:**

```
void set_face_ry_desired(float face_ry_desired);
```

Description:

This function sets the desired elevation (down/up) facing angle of the I-Guy character.

Most often this value is changed via mouse, keyboard, or joystick input, not by this function.

Callable From:

- ◇ C++
- ◇ Script

5.43.48 function diguyIGuyController::get_face_ry_desired**Prototype:**

```
float get_face_ry_desired();
```

Returns:

desired ry facing angle, in degrees

Callable From:

- ◇ C++
- ◇ Script

5.43.49 function diguyIGuyController::face_character**Prototype:**

```
void face_character(const char* character_name,
                   const char* link,
                   float offset_rz = 0.0f,
                   float offset_ry = 0.0f);
```

Description:

This function sets the desired rz and ry facing angles of the I-Guy controlled character toward the specified character.

Callable From:

- ◇ C++
- ◇ Script

Deprecated Functions

The following functions are deprecated.

As these functions may disappear in a future version of DI-Guy, it is recommended that you remove calls to these functions from your application.

5.43.50 function

diguyIGuyController::set_mouse_look_changes_orientation

Prototype:

```
void set_mouse_look_changes_orientation(int val);
```

Deprecated; use [set_orientation_change_enabled\(\)](#) instead.

5.43.51 function `diguyIGuyController::get_mouse_look_changes_orientation`

Prototype:

```
int get_mouse_look_changes_orientation();
```

Deprecated; use `get_orientation_change_enabled()` instead.

5.44 diguyImpact

diguyImpact.h

DI-Guy SDK Version 12.5.2

Includes: [declspec_diguy.h](#) | [diguy_constants.h](#) | [diguy_vector_classes.h](#)

Contents:

- ◆ [Accessor functions](#)
- ◆ [Experimental DIS Detonation Functions](#)
- ◆ [Utility Functions for Intersection Functions](#)

Link against: **libdiguy**

5.44.1 class diguyImpact

```
class BDI_DECLSPEC_diguy diguyImpact
{
public:
    diguyImpactType get_impact_type();
    const char* get_impact_type_string();
    int get_num_hits();
    const char* get_attacker_name();
    diguyCharacter* get_attacker();
    const char* get_target_name();
    diguyCharacter* get_target();
    const char* get_hit_link_name();
    int get_hit_link_index();
    const char* get_hit_shape_name();
    int get_impact_number();
    void set_impact_number(int val);
    void set_impact_been_processed();
    int get_impact_been_processed();
    float get_x();
    float get_y();
    float get_z();
    void get_location(float* x, float* y, float* z);
    float get_impact_radius();
    float get_impact_speed();
    int get_location_in_link_coordinates(float* link_x,
        float* link_y,
        float* link_z);
    void reset();
    const char* get_munition_type();
    diguySceneObject* get_hit_scene_object();
    float get_normal_x();
    float get_normal_y();
    float get_normal_z();
    void get_impact_normal(float* x, float* y, float* z);
    void get_impact_velocity(float* x, float* y, float* z);
    int get_is_from_net_detonation();
    int get_DIS_munition_septet_value(int septet_index);
```

```

int get_DIS_site();
int get_DIS_host();
int get_DIS_event_number();
int get_DIS_munition_number();
int get_DIS_detonation_result();
int get_DIS_burst_warhead_type();
int get_DIS_burst_fuze();
int get_DIS_burst_quantity();
int get_DIS_burst_rate();
#ifdef CPLUSPLUS_ONLY
void set_valid_impact();
void set_target(diguyCharacter* target);
void set_location(float x, float y, float z);
void set_normal(float nx, float ny, float nz);
void set_hit_scene_object(diguySceneObject* scene_object);
void set_hit_link_index(int index);
void set_hit_shape_index(int index);

```

Accessor functions

Unless otherwise noted all functions in this class section are callable from:

- ◆ C++
- ◆ Script

5.44.2 function diguyImpact::get_impact_type

Prototype:

```
diguyImpactType get_impact_type();
```

Description:

Returns the type of impact; this is based on how the collision was generated. Possible values are:

- ◇ DIGUY_IMPACT_DIRECT_HIT
- ◇ DIGUY_IMPACT_NEARBY_DETONATION
- ◇ DIGUY_IMPACT_VEHICLE_COLLISION
- ◇ DIGUY_IMPACT_NO_IMPACT
- ◇ DIGUY_IMPACT_USER_GENERATED

Note: NO_IMPACT is generated when a character fires but fails the hit ratio test. The location of the impact is then set to the attacker's position, or if that is unavailable to: (DIGUY_DEFAULT_FLOAT, DIGUY_DEFAULT_FLOAT, DIGUY_DEFAULT_FLOAT).

Returns:

the type of impact

5.44.3 function diguyImpact::get_impact_type_string

Prototype:

```
const char* get_impact_type_string();
```

Returns:

the string version of the type of impact. This is based on how the collision was generated. Possible values are:

- ◇ "direct_hit"
- ◇ "nearby_detonation"
- ◇ "vehicle_collision"
- ◇ "no_impact"
- ◇ "user_generated"

5.44.4 function diguyImpact::get_num_hits

Prototype:

```
int get_num_hits();
```

Returns:

the number of hits made by this impact. This includes hits against characters and scene objects.

5.44.5 function diguyImpact::get_attacker_name

Prototype:

```
const char* get_attacker_name();
```

Returns:

the name of the attacker. This pointer may be NULL if there is no attacker character.

5.44.6 function diguyImpact::get_attacker

Prototype:

```
diguyCharacter* get_attacker();
```

Returns:

Returns a pointer to the attacker. This pointer may be NULL.

5.44.7 function diguyImpact::get_target_name

Prototype:

```
const char* get_target_name();
```

Returns:

the name of the target. This pointer may be NULL if there is no impact target character (e.g., impact is against a scene object).

5.44.8 function diguyImpact::get_target

Prototype:

```
diguyCharacter* get_target();
```

Returns:

the [diguyCharacter](#) impact target. This pointer may be NULL if there is no impact target character (e.g., impact is against a scene object).

5.44.9 function diguyImpact::get_hit_link_name

Prototype:

```
const char* get_hit_link_name();
```

Returns:

name of the target character link that was hit; may be NULL if there is no hit target character

5.44.10 function diguyImpact::get_hit_link_index**Prototype:**

```
int get_hit_link_index();
```

Returns:

index of the target character link that was hit

5.44.11 function diguyImpact::get_hit_shape_name**Prototype:**

```
const char* get_hit_shape_name();
```

Returns:

name of the target character shape that was hit; may be NULL if there is no hit target character

5.44.12 function diguyImpact::get_impact_number**Prototype:**

```
int get_impact_number();
```

Returns:

the number of impacts the target character has taken

5.44.13 function diguyImpact::set_impact_number**Prototype:**

```
void set_impact_number(int val);
```

Description:

Sets the impact number.

5.44.14 function diguyImpact::set_impact_been_processed**Prototype:**

```
void set_impact_been_processed();
```

Description:

Sets that the impact has been processed.

5.44.15 function diguyImpact::get_impact_been_processed**Prototype:**

```
int get_impact_been_processed();
```

Returns:

1 if the impact has been processed, 0 if not

5.44.16 function diguyImpact::get_x

Prototype:

```
float get_x();
```

Returns:

the approximate world x coordinate of the impact

5.44.17 function diguyImpact::get_y

Prototype:

```
float get_y();
```

Returns:

the approximate world y coordinate of the impact

5.44.18 function diguyImpact::get_z**Prototype:**

```
float get_z();
```

Returns:

the approximate world z coordinate of the impact

5.44.19 function diguyImpact::get_location**Prototype:**

```
void get_location(float* x, float* y, float* z);
```

Description:

Alternative method for getting the location of the impact.

5.44.20 function diguyImpact::get_impact_radius**Prototype:**

```
float get_impact_radius();
```

Returns:

the radius of the impact

5.44.21 function diguyImpact::get_impact_speed**Prototype:**

```
float get_impact_speed();
```

Returns:

the speed of the impact

5.44.22 function diguyImpact::get_location_in_link_coordinates**Prototype:**

```
int get_location_in_link_coordinates(float* link_x,
    float* link_y,
    float* link_z);
```

Description:

This function returns the location of the impact in link-relative coordinates. The function may fail and return -1 if a link was not hit by the impact.

Arguments:

link_x, link_y, link_z link-relative coordinates are
link_y,
link_z

returned in these variables

Returns:

0 on success, -1 on failure

5.44.23 function diguyImpact::reset**Prototype:**

```
void reset();
```

Description:

This function resets all of the impact data to default values.

5.44.24 function diguyImpact::get_munition_type

Prototype:

```
const char* get_munition_type();
```

Description:

This function returns the name of the munition that was fired. This is useful for creating impact callbacks that respond differently to different weapons.

Returns:

the munition type of the impact; may be NULL

5.44.25 function diguyImpact::get_hit_scene_object

Prototype:

```
diguySceneObject* get_hit_scene_object();
```

Description:

Returns the hit [diguySceneObject](#).

This pointer may be NULL if there is no impact scene object (e.g., impact is against a character).

If the octtree was used to generate the impact, this pointer will be NULL.

Returns:

pointer of type [diguySceneObject](#)

5.44.26 function diguyImpact::get_normal_x**Prototype:**

```
float get_normal_x();
```

Returns:

approximate normal x coordinate of the impact; may not be valid

5.44.27 function diguyImpact::get_normal_y**Prototype:**

```
float get_normal_y();
```

Returns:

approximate normal y coordinate of the impact; may not be valid

5.44.28 function diguyImpact::get_normal_z**Prototype:**

```
float get_normal_z();
```

Returns:

approximate normal z coordinate of the impact; may not be valid

5.44.29 function diguyImpact::get_impact_normal**Prototype:**

```
void get_impact_normal(float* x, float* y, float* z);
```

Description:

Alternative method for getting the normal of the impact.

Experimental DIS Detonation Functions

The following functions are experimental. They are for accessing data from DIS detonations that result in DI-Guy impacts being generated.

5.44.30 function diguyImpact::get_impact_velocity

Prototype:

```
void get_impact_velocity(float* x, float* y, float* z);
```

Description:

Velocity of projectile at impact. May be 0 if no velocity information is available.

5.44.31 function diguyImpact::get_is_from_net_detonation**Prototype:**

```
int get_is_from_net_detonation();
```

Returns:

1 if impact was generated as a result of a network detonation (e.g. a DIS detonation PDU), else 0

5.44.32 function diguyImpact::get_DIS_munition_septet_value**Prototype:**

```
int get_DIS_munition_septet_value(int septet_index);
```

Description:

This function returns the specified part of the DIS septet of the detonation's munition septet.

Arguments:

`septet_index` which part of the septet to return; must be between 0 (for 'kind') and 6 (for 'extra')

Returns -1 if septet isn't specified, or on error.

5.44.33 function diguyImpact::get_DIS_site**Prototype:**

```
int get_DIS_site();
```

Returns:

the site id part of the detonation PDU event id; -1 if not set

5.44.34 function diguyImpact::get_DIS_host**Prototype:**

```
int get_DIS_host();
```

Returns:

the host id part of the detonation PDU event id; -1 if not set

5.44.35 function diguyImpact::get_DIS_event_number**Prototype:**

```
int get_DIS_event_number();
```

Returns:

the event number part of the detonation PDU event id; -1 if not set

5.44.36 function diguyImpact::get_DIS_munition_number**Prototype:**

```
int get_DIS_munition_number();
```

Returns:

the event number part of the detonation PDU **munition** event id; -1 if not set

5.44.37 function diguyImpact::get_DIS_detonation_result**Prototype:**

```
int get_DIS_detonation_result();
```

Returns:

the detonation result of the detonation PDU; -1 if not set

5.44.38 function diguyImpact::get_DIS_burst_warhead_type**Prototype:**

```
int get_DIS_burst_warhead_type();
```

Returns:

the warhead type enumeration of the detonation PDU's burst descriptor; -1 if not set

5.44.39 function diguyImpact::get_DIS_burst_fuze**Prototype:**

```
int get_DIS_burst_fuze();
```

Returns:

the fuze enumeration of the detonation PDU's burst descriptor; -1 if not set

5.44.40 function diguyImpact::get_DIS_burst_quantity**Prototype:**

```
int get_DIS_burst_quantity();
```

Returns:

the burst quantity of the detonation PDU's burst descriptor; -1 if not set

5.44.41 function diguyImpact::get_DIS_burst_rate**Prototype:**

```
int get_DIS_burst_rate();
```

Returns:

the burst rate of the detonation PDU's burst descriptor; -1 if not set

Utility Functions for Intersection Functions

The following functions are meant for users who are using `diguyScenario::set_fire_weapon_intersection_function()` to handle weapon firing in the SDK.

In simple OpenGL environments

`diguy_default_fire_weapon_intersection_function()` in `diguy_graphics_ogl.h` may give you reasonable results by rendering characters into the back buffer.

Additionally there are a number of intersection programming examples that demonstrate putting the default rendering pipeline into picking mode. Handing the resulting color under the (x, y) impact location in the graphics buffer into `diguyScenario::map_color_to_impact()` allows you to retrieve an impact structure with much of this information.

These functions are not callable from script.

Example impact function:


```

int impact_function(diguyImpact* impact,
    float from_x,
    float from_y,
    float from_z,
    float to_x,
    float to_y,
    float to_z,
    diguyScenario* s)
{
    // note the impact already has both the attacker and the munition type
    // set when this function is called
    glShadeModel(GL_FLAT);

    // create narrow frustum
    // turn on scissor test
    // feed data into diguy camera if you want frustum culling to work

    diguyApp::get_app()->set_render_pass_flags(DIGUY_RENDER_PASS_INTERSECTION);
    s->set_current_render_mode("intersection");

    s->draw();

    GLubyte pixel[3];
    glReadPixels(x_center, y_center, 1, 1,
        GL_RGB,
        GL_UNSIGNED_BYTE,
        (void *) pixel);

    diguyImpact* new_impact = scenario->map_color_to_impact(pixel[0], pixel[1], pixel[2]);

    // unpack data in new_impact into the callback object

    impact->set_valid_impact();
    impact->set_target(new_impact->get_target());
    impact->set_hit_link_index(new_impact->get_hit_link_index());
    impact->set_hit_shape_index(new_impact->get_hit_shape_index());

    return 1;
}

```

5.44.42 function diguyImpact::set_valid_impact**Prototype:**

```
void set_valid_impact();
```

Description:

This function sets the impact to be valid. It must be called if the impact hits a character.

5.44.43 function diguyImpact::set_target**Prototype:**

```
void set_target(diguyCharacter* target);
```

Description:

This function allows for explicitly setting the target character of this impact object.

Arguments:

x, y, z override target of impact

Callable From:

◇ C++

5.44.44 function diguyImpact::set_location**Prototype:**

```
void set_location(float x, float y, float z);
```

Description:

This function allows for explicitly setting the position of this impact object.

Arguments:

x, y, z override position of impact

Callable From:

◇ C++

5.44.45 function diguyImpact::set_normal**Prototype:**

```
void set_normal(float nx, float ny, float nz);
```

Description:

This function allows for explicitly setting the normal of this impact object.

Arguments:

nx, ny, nz override normal of impact

Callable From:

◇ C++

5.44.46 function diguyImpact::set_hit_scene_object**Prototype:**

```
void set_hit_scene_object(diguySceneObject* scene_object);
```

Description:

Returns the hit scene object.

This function allows for explicitly setting the scene_object that the impact object hit.

If the octtree was used to generate the impact, this pointer will be NULL.

Arguments:

scene_object pointer to scene object hit

Callable From:

◇ C++

5.44.47 function diguyImpact::set_hit_link_index**Prototype:**

```
void set_hit_link_index(int index);
```

Description:

This function allows for explicitly setting the link index of this impact object.

Callable From:

◇ C++

5.44.48 function diguyImpact::set_hit_shape_index**Prototype:**

```
void set_hit_shape_index(int index);
```

Description:

This function allows for explicitly setting the shape index of this impact object.

Callable From:

◇ C++

5.45 diguyInfoPopup

diguyInfoPopup.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

Link against: libdiguy

5.45.1 class diguyInfoPopup

```
class BDI_DECLSPEC_diguy diguyInfoPopup
{
public:
    const char* get_name();
    int set_name(const char* name);
    int show(float show_duration = -1.0f);
    int hide();
    int is_showing();
    int clear();
    int append_file(const char* filename);
    int append_string(const char* string);
```

5.45.2 function diguyInfoPopup::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the info popup. This pointer will never be NULL.

Returns:

name of the info popup

Callable From:

- ◇ C++
- ◇ Script

5.45.3 function diguyInfoPopup::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.45.4 function diguyInfoPopup::show

Prototype:

```
int show(float show_duration = -1.0f);
```

Description:

This function shows the info popup.

Arguments:

<code>duration</code>	length of time in seconds to show to the info popup; the default argument of <code>-1</code> will show the info popup indefinitely, until the <code>hide()</code> function is called

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.45.5 function diguyInfoPopup::hide

Prototype:

```
int hide();
```

Description:

This function hides the info popup, if it is currently shown.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.45.6 function diguyInfoPopup::is_showing

Prototype:

```
int is_showing();
```

Returns:

Returns 1 if the info popup is showing, 0 if not.

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.45.7 function diguyInfoPopup::clear

Prototype:

```
int clear();
```

Description:

This function clears the contents of the info popup.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.45.8 function diguyInfoPopup::append_file

Prototype:

```
int append_file(const char* filename);
```

Description:

This function appends the contents of the file identified by filename to the info popup.

Arguments:

`filename` name of the file

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++

◇ Script

5.45.9 function diguyInfoPopup::append_string

Prototype:

```
int append_string(const char* string);
```

Description:

This function appends a string to the info popup.

Arguments:

string	string to append

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.46 diguyInteractionMachine

diguyInteractionMachine.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

- ◆ General Functions
- ◆ UI Functions
- ◆ Heading Functions
- ◆ Info Functions
- ◆ State Functions
- ◆ Input Functions
- ◆ Deprecated Functions

`diguyInteractionMachine` objects give DI-Guy Scenario users a way of providing interactive, UI-driven input to a scenario that can change the way a scenario progresses.

Interaction machines in DI-Guy Scenario are good for presenting users with simple yes-or-no questions, and also "dialog tree" type conversations with characters in a scenario.

It is called a "machine" because it is implemented as a straight-forward state machine. A state machine has some number of "states" it can be in. Each state responds to some number of inputs, which can advance the machine to a new state or end the interaction.

Graphically, interaction machines are represented as dialog boxes in DI-Guy Scenario. There are three areas to the dialog: the top, called the header, which provides a one-line title for the machine; the middle, which shows informational text of the current state; and the bottom, which presents the user with options they can click to proceed to a new state.

Example:

As an example, suppose that there is an interaction machine called "converse_with_victim". It has four states:

- ◆ "begin_conversation"
- ◆ "asked_about_injury"
- ◆ "goodbye"
- ◆ "angry_goodbye"

This example doesn't describe all info and inputs that this machine needs, but does give an example of some of them.

When the interaction starts the machine is in the state "begin_conversation". The user is presented with a dialog that has the header "Talking with victim". The info portion of the dialog is set to: "An accident victim is standing in front of you, looking disoriented." The inputs presented are:

If the user selects input 1, the machine proceeds to the state "asked_about_injury", where new info is presented telling the user about any injuries the victim may have, and new inputs that present further questions the user can ask of the victim or actions the user can take. (Note that not all inputs need to be conversation or questions; inputs can also describe actions that the user may take.)

If the user selects input 2, the machine proceeds to the state "angry_goodbye", where new info tells that user that the victim is angry at being ignored. There is one input, "Ok", that ends the interaction and closes the interaction machine.

5.46.1 class diguyInteractionMachine

```
class BDI_DECLSPEC diguy diguyInteractionMachine
{
public:
    const char* get_name();
    int begin_interaction(const char* initial_state = NULL,
        const char* initial_input = NULL);
    int end_interaction();
    int get_interaction_is_active();
    void set_pause_scenario_while_active(int pause_while_active);
    int get_pause_scenario_while_active();
    void reinitialize();
    void reset();
    void set_interaction_subject(diguyCharacter* interaction_subject);
    diguyCharacter* get_interaction_subject();
    int update_ui();
    enum {
        UI_APPEARANCE_CONVERSATION = 0,
        UI_APPEARANCE_INFO,
        UI_APPEARANCE_NOTIFICATION,
        UI_APPEARANCE_3,
        UI_APPEARANCE_4,
        UI_APPEARANCE_COUNT
    };
    int set_initial_ui_appearance(int initial_ui_appearance);
    int get_initial_ui_appearance();
    int set_ui_appearance(int ui_appearance);
    int get_ui_appearance();
    void set_initial_heading(const char* initial_heading);
    const char* get_initial_heading();
    void set_heading(const char* heading);
    const char* get_heading();
    void set_heading_visible(int heading_visible);
    int get_heading_visible();
    void set_heading_color_bg(int ui_appearance,
        float r,
        float g,
        float b,
        float a = 1.0f);
    void set_heading_color_fg(int ui_appearance,
        float r,
        float g,
        float b,
        float a = 1.0f);
    void set_heading_font(int ui_appearance,
        const char* font_family,
```

```

        int font_point_size,
        int font_is_bold = 0,
        int font_is_italic = 0);
void set_initial_info(const char* initial_info);
const char* get_initial_info();
void set_info(const char* info);
const char* get_info();
void set_info_visible(int info_visible);
int get_info_visible();
void set_info_color_bg(int ui_appearance,
    float r,
    float g,
    float b,
    float a = 1.0f);
void set_info_color_fg(int ui_appearance,
    float r,
    float g,
    float b,
    float a = 1.0f);
void set_info_font(int ui_appearance,
    const char* font_family,
    int font_point_size,
    int font_is_bold = 0,
    int font_is_italic = 0);
int get_num_states();
const char* get_state_id_at_index(int index);
void clear_states();
int add_state(const char* state_id,
    const char* entry_script);
int remove_state(const char* state_id);
int rename_state(const char* old_state_id, const char* new_state_id);
int set_state_entry_script(const char* state_id,
    const char* entry_script);
int set_begin_state_entry_script(const char* entry_script);
const char* get_current_state();
void set_current_state(const char* new_current_state,
    int run_entry_script = 1);
void clear_inputs();
int add_input(const char* input_id,
    const char* input_string,
    const char* resulting_state = "",
    int int_param_0 = 0);
int remove_input(const char* input_id);
int get_num_inputs();
const char* get_input_id(int index);
const char* get_input_string(const char* input_id);
const char* get_input_resulting_state(const char* input_id);
int get_input_int_param_0(const char* input_id);
int select_input(const char* input_id);
const char* get_last_selected_input();
void set_last_input_variable(const char* last_input_variable);
const char* get_last_input_variable();
void set_inputs_visible(int visible);
int get_inputs_visible();
void set_input_shortcuts_visible(int visible);
int get_input_shortcuts_visible();
void set_inputs_font(int ui_appearance,
    const char* font_family,
    int font_point_size,
    int font_is_bold = 0,
    int font_is_italic = 0);

```

```
void set_inputs_color_bg(int ui_appearance,  
    float r,  
    float g,  
    float b,  
    float a = 1.0f);  
void set_inputs_color_fg(int ui_appearance,  
    float r,  
    float g,  
    float b,  
    float a = 1.0f);  
int set_current_settings_index(int current_settings_index);  
void set_input_response_script(const char* script);
```

General Functions

5.46.2 function diguyInteractionMachine::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the object

Callable From:

- ◇ C++
- ◇ Script

5.46.3 function diguyInteractionMachine::begin_interaction

Prototype:

```
int begin_interaction(const char* initial_state = NULL,
                     const char* initial_input = NULL);
```

Description:

This function begins the interaction.

The interaction UI will appear in DI-Guy Scenario.

The interaction can be ended by calling [end_interaction\(\)](#), or by the current state being changed to "end" through an input selection or a call to [set_current_state\(\)](#).

Arguments:

initial_state initial state that interaction will be in

initial_input initial input that will be returned by
get_last_selected_input() in the first state

By default, initial state will be "begin". By default, initial input will be "begin".

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.46.4 function diguyInteractionMachine::end_interaction

Prototype:

```
int end_interaction();
```

Description:

This function ends the interaction.

The interaction UI will be hidden in DI-Guy Scenario.

Note that this call does not run the entry script of the "end" state. Call [set_current_state\("end"\)](#) to end the interaction and run the entry script of the "end" state when doing so.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++

◇ Script

5.46.5 function diguyInteractionMachine::get_interaction_is_active

Prototype:

```
int get_interaction_is_active();
```

Returns:

1 if interaction is active and visible, 0 if not

Callable From:

◇ C++
◇ Script

5.46.6 function**diguyInteractionMachine::set_pause_scenario_while_active****Prototype:**

```
void set_pause_scenario_while_active(int pause_while_active);
```

Description:

This function sets whether the scenario is paused while the interaction is active.

If the scenario is to be paused, the current playback mode will be saved when the interaction begins, the playback mode will be changed to [DIGUY_SCENARIO_PLAYBACK_MODE_STOP](#), and the original playback mode will be restored when the interaction ends.

If the scenario is not to be paused, beginning and ending the interaction will have no effect on the playback mode.

The default is to not pause the scenario.

Arguments:

`pause_while_active` pass 1 to pause scenario when interaction is active, 0 to not change playback mode

Callable From:

- ◇ C++
- ◇ Script

5.46.7 function**diguyInteractionMachine::get_pause_scenario_while_active****Prototype:**

```
int get_pause_scenario_while_active();
```

Returns:

1 if playback mode will be paused, 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.46.8 function diguyInteractionMachine::reinitialize**Prototype:**

```
void reinitialize();
```

Description:

This function resets the interaction to a clean state. It will:

- ◇ set the initial heading to ""
- ◇ set the initial info text to ""
- ◇ remove all inputs
- ◇ reset ui appearances
- ◇ set all other parameters to default values It will also do an implicit `reset( )` call.

Callable From:

- ◇ C++
- ◇ Script

5.46.9 function diguyInteractionMachine::reset

Prototype:

```
void reset();
```

Description:

This function resets the interaction to a reset state. It will:

- ◇ set the UI appearance text to initial_ui_appearance
- ◇ set the heading to be initial_heading
- ◇ set the info text to initial_info
- ◇ set the current state to "begin"

Callable From:

- ◇ C++
- ◇ Script

5.46.10 function diguyInteractionMachine::set_interaction_subject

Prototype:

```
void set_interaction_subject(diguyCharacter* interaction_subject);
```

Description:

This function stores a pointer to the passed character in the interaction machine. Access to the character is then possible in state scripts by calling [get_interaction_subject\(\)](#).

Arguments:

`interaction_subject` pointer to character that is to by the interaction's primary subject

Callable From:

- ◇ C++
- ◇ Script

5.46.11 function `diguyInteractionMachine::get_interaction_subject`

Prototype:

```
diguyCharacter* get_interaction_subject();
```

Returns:

value set by `set_interaction_subject()`

Callable From:

- ◇ C++
- ◇ Script

UI Functions

5.46.12 function diguyInteractionMachine::update_ui

Prototype:

```
int update_ui();
```

Description:

This function updates the visible interaction UI in DI-Guy Scenario. May calls that change visible settings of the UI will not take effect until this function is called.

Callable From:

- ◇ C++
- ◇ Script

5.46.13 enumeration

```
enum {
    UI_APPEARANCE_CONVERSATION = 0,
    UI_APPEARANCE_INFO,
    UI_APPEARANCE_NOTIFICATION,
    UI_APPEARANCE_3,
    UI_APPEARANCE_4,

    UI_APPEARANCE_COUNT
};
```

Description:

This enumeration lists the indices used for various UI appearance calls. These indices should be used when calling:

- ◇ `set_initial_ui_appearance()`
- ◇ `set_ui_appearance()`
- ◇ `set_heading_color_bg()`
- ◇ `set_heading_color_fg()`
- ◇ `set_heading_font()`

and equivalent functions for info and inputs.

Each UI appearance can have different colors and font settings, potentially giving the scenario user a visual cue that the interaction is showing a conversation, showing generation information, or providing an important notification,

Callable From:

- ◇ C++
- ◇ Script

5.46.14 function diguyInteractionMachine::set_initial_ui_appearance

Prototype:

```
int set_initial_ui_appearance(int initial_ui_appearance);
```

Description:

This function sets the UI appearance that will be used after a [begin_interaction\(\)](#) or [reset\(\)](#) call.

The default is UI_APPEARANCE_CONVERSATION.

Arguments:

`initial_ui_appearance` initial UI appearance

Callable From:

- ◇ C++
- ◇ Script

5.46.15 function diguyInteractionMachine::get_initial_ui_appearance

Prototype:

```
int get_initial_ui_appearance();
```

Returns:

initial UI appearance as set by [set_initial_ui_appearance\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.46.16 function diguyInteractionMachine::set_ui_appearance**Prototype:**

```
int set_ui_appearance(int ui_appearance);
```

Description:

This function sets the current UI appearance that should be used for the interaction.

The new setting will not take effect until the next call to [update_ui\(\)](#).

Call [update_ui\(\)](#) for new setting to take effect.

Arguments:

`ui_appearance` new current UI appearance

Callable From:

- ◇ C++
- ◇ Script

5.46.17 function diguyInteractionMachine::get_ui_appearance**Prototype:**

```
int get_ui_appearance();
```

Returns:

current UI appearance; default is value returned by [get_initial_ui_appearance\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

Heading Functions

5.46.18 function diguyInteractionMachine::set_initial_heading

Prototype:

```
void set_initial_heading(const char* initial_heading);
```

Description:

This function sets the heading text that will be shown after after a [begin_interaction\(\)](#) or [reset\(\)](#) call.

The default is "", or show nothing.

Arguments:

`initial_heading` initial heading text

Callable From:

- ◇ C++
- ◇ Script

5.46.19 function diguyInteractionMachine::get_initial_heading**Prototype:**

```
const char* get_initial_heading();
```

Returns:

initial heading text as set by [set_initial_heading\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.46.20 function diguyInteractionMachine::set_heading**Prototype:**

```
void set_heading(const char* heading);
```

Description:

This function sets the current heading text that will be shown.

Call [update_ui\(\)](#) for new setting to take effect.

Arguments:

heading	heading text to be shown

Callable From:

- ◇ C++
- ◇ Script

5.46.21 function diguyInteractionMachine::get_heading**Prototype:**

```
const char* get_heading();
```

Returns:

current heading text as set by [set_heading\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.46.22 function diguyInteractionMachine::set_heading_visible**Prototype:**

```
void set_heading_visible(int heading_visible);
```

Description:

This function sets whether the heading will be visible.

Call [update_ui\(\)](#) for new setting to take effect.

Arguments:

`heading_visible` 1 for visible, 0 for invisible

Callable From:

- ◇ C++
- ◇ Script

5.46.23 function diguyInteractionMachine::get_heading_visible**Prototype:**

```
int get_heading_visible();
```

Returns:

whether heading is visible

Callable From:

- ◇ C++
- ◇ Script

5.46.24 function diguyInteractionMachine::set_heading_color_bg**Prototype:**

```
void set_heading_color_bg(int ui_appearance,
    float r,
    float g,
    float b,
    float a = 1.0f);
```

Description:

This function sets the background color of the heading.

Call [update_ui\(\)](#) for new setting to take effect.

Arguments:

`ui_appearance` which UI appearance this changes
`r, g, b, a` RGBA colors, from 0.0 to 1.0

Callable From:

◇ C++
◇ Script

5.46.25 function diguyInteractionMachine::set_heading_color_fg

Prototype:

```
void set_heading_color_fg(int ui_appearance,  
    float r,  
    float g,  
    float b,  
    float a = 1.0f);
```

Description:

This function sets the foreground/font color of the heading.

Call [update_ui\(\)](#) for new setting to take effect.

Arguments:

`ui_appearance` which UI appearance this changes
`r, g, b, a` RGBA colors, from 0.0 to 1.0

Callable From:

◇ C++
 ◇ Script

5.46.26 function diguyInteractionMachine::set_heading_font

Prototype:

```
void set_heading_font(int ui_appearance,
                     const char* font_family,
                     int font_point_size,
                     int font_is_bold = 0,
                     int font_is_italic = 0);
```

Description:

This function sets the font characteristics of the heading.

Call [update_ui\(\)](#) for new setting to take effect.

Arguments:

<code>ui_appearance</code>	which UI appearance this changes
<code>font_family</code>	font family
<code>font_point_size</code>	point size of font
<code>font_is_bold</code>	pass 1 for bold, 0 for normal

`font_is_italic` pass 1 for italic, 0 for normal

On Windows, the default font family is ""MS Shell Dlg", and the default point size is 10.

Callable From:

- ◇ C++
- ◇ Script

Info Functions

5.46.27 function diguyInteractionMachine::set_initial_info

Prototype:

```
void set_initial_info(const char* initial_info);
```

Description:

Similar to `set_initial_heading( )`, but for info instead.

Note that the info text can use simple HTML-style tags. For example:

- for a paragraph break:
- for bold text: **bold**
- for underlined text: bold
- for italicized text: *bold*

The default is "", or show nothing.

5.46.28 function diguyInteractionMachine::get_initial_info

Prototype:

```
const char* get_initial_info();
```

Description:

Similar to `get_initial_heading( )`, but for info instead.

5.46.29 function diguyInteractionMachine::set_info**Prototype:**

```
void set_info(const char* info);
```

Description:

Similar to [set_heading\(\)](#), but for info instead.

5.46.30 function diguyInteractionMachine::get_info**Prototype:**

```
const char* get_info();
```

Description:

Similar to [get_heading\(\)](#), but for info instead.

5.46.31 function diguyInteractionMachine::set_info_visible**Prototype:**

```
void set_info_visible(int info_visible);
```

Description:

Similar to [set_heading_visible\(\)](#), but for info instead.

5.46.32 function diguyInteractionMachine::get_info_visible**Prototype:**

```
int get_info_visible();
```

Description:

Similar to [get_heading_visible\(\)](#), but for info instead.

5.46.33 function diguyInteractionMachine::set_info_color_bg**Prototype:**

```
void set_info_color_bg(int ui_appearance,  
    float r,  
    float g,  
    float b,  
    float a = 1.0f);
```

Description:

Similar to [get_heading_visible\(\)](#), but for info instead.

5.46.34 function diguyInteractionMachine::set_info_color_fg**Prototype:**

```
void set_info_color_fg(int ui_appearance,  
    float r,  
    float g,  
    float b,  
    float a = 1.0f);
```

Description:

Similar to [set_heading_color_fg\(\)](#), but for info instead.

5.46.35 function diguyInteractionMachine::set_info_font

Prototype:

```
void set_info_font(int ui_appearance,  
    const char* font_family,  
    int font_point_size,  
    int font_is_bold = 0,  
    int font_is_italic = 0);
```

Description:

Similar to [set_heading_font\(\)](#), but for info instead.

State Functions

States in an interaction machine represent the various states the machine can be in. The current state is typically changed by the user selecting inputs via the UI, but can also be changed by calling [set_current_state\(\)](#).

New interaction machines always have the required "begin" and "end" states.

5.46.36 function diguyInteractionMachine::get_num_states**Prototype:**

```
int get_num_states();
```

Returns:

number of states in the interaction machine

Callable From:

- ◇ C++
- ◇ Script

5.46.37 function diguyInteractionMachine::get_state_id_at_index**Prototype:**

```
const char* get_state_id_at_index(int index);
```

Returns:

string id of state with given index; the returned pointer will never be NULL

Callable From:

- ◇ C++
- ◇ Script

5.46.38 function diguyInteractionMachine::clear_states**Prototype:**

```
void clear_states();
```

Description:

This function removes all existing states from the interaction machine.

Default new states named "begin" and "end" are then automatically added. These default states do not have an entry script, but one can be added later with the [set_state_entry_script\(\)](#) or [set_begin_state_entry_script\(\)](#) functions.

More new states can be added using the [add_state\(\)](#) function.

Callable From:

- ◇ C++
- ◇ Script

5.46.39 function diguyInteractionMachine::add_state**Prototype:**

```
int add_state(const char* state_id,
              const char* entry_script);
```

Description:

This function adds a new state to the interaction machine. Each state must have a unique state_id (name of the state).

Arguments:

state_id	name of the new state
entry_script	script text that will be run every time this new state is entered

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.46.40 function diguyInteractionMachine::remove_state

Prototype:

```
int remove_state(const char* state_id);
```

Description:

This function removes the specified state from the interaction machine.

The required "begin" and "end" states cannot be removed.

Arguments:

`state_id` name of state to remove

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.46.41 function diguyInteractionMachine::rename_state

Prototype:

```
int rename_state(const char* old_state_id, const char* new_state_id);
```

Description:

This function renames the specified state. Each state must have a unique state_id (name of the state).

The required "begin" and "end" states cannot be renamed.

Arguments:

old_state_id name of state to rename
new_state_id new name of state

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.46.42 function diguyInteractionMachine::set_state_entry_script**Prototype:**

```
int set_state_entry_script(const char* state_id,  
                           const char* entry_script);
```

Description:

This function sets the entry script of this state. Each time the specified state is entered, the entry script will be run.

Arguments:

<code>state_id</code>	name of state to modify
<code>entry_script</code>	new entry script

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
 ◇ Script

5.46.43 function diguyInteractionMachine::set_begin_state_entry_script

Prototype:

```
int set_begin_state_entry_script(const char* entry_script);
```

Description:

This function sets the entry script of the required "begin" state.

Arguments:

entry_script new entry script

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

◇ Script

5.46.44 function diguyInteractionMachine::get_current_state

Prototype:

```
const char* get_current_state();
```

Description:

This function returns the current state the interaction machine is in.

Returns:

name of current state; the returned should never be NULL

Callable From:

◇ C++
◇ Script

5.46.45 function diguyInteractionMachine::set_current_state**Prototype:**

```
void set_current_state(const char* new_current_state,
    int run_entry_script = 1);
```

Description:

This function sets the current state of the interaction machine. The entry script of the new state will be run by default, but can optionally be skipped.

If the new state is "end", the interaction will end, as if the [end_interaction\(\)](#) call is made, once the entry script of the "end" state entry script is run.

Arguments:

`new_current_state` new current state
`run_entry_script` pass 1 to run the new state's entry script, 0 to not

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Input Functions

5.46.46 function diguyInteractionMachine::clear_inputs

Prototype:

```
void clear_inputs();
```

Description:

This function removes all inputs from the interaction machine. New inputs can be added by calling [add_input\(\)](#).

Callable From:

- ◇ C++
- ◇ Script

5.46.47 function diguyInteractionMachine::add_input**Prototype:**

```
int add_input(const char* input_id,
             const char* input_string,
             const char* resulting_state = "",
             int int_param_0 = 0);
```

Description:

This function adds a new input to the interaction machine. Each input must have a unique input_id (name of the input).

If the resulting state is "end", the interaction will end, as if the [end_interaction\(\)](#) call is made, once the entry script of the "end" state entry script is run.

Arguments:

input_id	name of the new input
input_string	text that will be shown in the UI
resulting_state	state id of the state the interaction machine will change to if this new input is selected
int_param_0	an optional integer parameter that can be interpreted in the new state's entry script

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.46.48 function diguyInteractionMachine::remove_input**Prototype:**

```
int remove_input(const char* input_id);
```

Description:

This function removes the specified input.

Arguments:

`input_id` name of input to remove

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.46.49 function diguyInteractionMachine::get_num_inputs**Prototype:**

```
int get_num_inputs();
```

Returns:

how many inputs there are

Callable From:

- ◇ C++
- ◇ Script

5.46.50 function diguyInteractionMachine::get_input_id**Prototype:**

```
const char* get_input_id(int index);
```

Returns:

input id of the input at the specified index; see [add_input\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.46.51 function diguyInteractionMachine::get_input_string**Prototype:**

```
const char* get_input_string(const char* input_id);
```

Returns:

input string of the specified input; see [add_input\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.46.52 function `diguyInteractionMachine::get_input_resulting_state`

Prototype:

```
const char* get_input_resulting_state(const char* input_id);
```

Returns:

resulting state of the specified input; see [add_input\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.46.53 function `diguyInteractionMachine::get_input_int_param_0`

Prototype:

```
int get_input_int_param_0(const char* input_id);
```

Returns:

integer parameter 0 of the specified input; see [add_input\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.46.54 function diguyInteractionMachine::select_input**Prototype:**

```
int select_input(const char* input_id);
```

Description:

This function "manually" selects the specified input. Most of the time inputs are selected via the UI.

A side effect of selecting an input is almost always that the interaction machine will be put into a new state.

Arguments:

`input_id` input id of input that should be selected

The passed input id must be a current available input of the interaction machine.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.46.55 function

diguyInteractionMachine::get_last_selected_input

Prototype:

```
const char* get_last_selected_input();
```

Returns:

input id of the last selected input; state entry scripts can use this information to make decisions based on how the state was entered

Callable From:

- ◇ C++
- ◇ Script

5.46.56 function

diguyInteractionMachine::set_last_input_variable

Prototype:

```
void set_last_input_variable(const char* last_input_variable);
```

Description:

This function sets the name of the scenario-level variable in which the name of the last selected input will be recorded. State entry scripts can then read determine which input caused the interaction machine to entry the state.

Arguments:

`last_input_variable` name of scenario-level variable

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.46.57 function diguyInteractionMachine::get_last_input_variable

Prototype:

```
const char* get_last_input_variable();
```

Returns:

name of last input variable; empty string ("") if there is none

Callable From:

◇ C++
◇ Script

5.46.58 function diguyInteractionMachine::set_inputs_visible**Prototype:**

```
void set_inputs_visible(int visible);
```

Description:

This function sets whether the inputs section of the UI will be visible. The inputs will be visible by default.

Arguments:

visible pass 1 to make inputs visible; 0 to not

Callable From:

- ◇ C++
- ◇ Script

5.46.59 function diguyInteractionMachine::get_inputs_visible**Prototype:**

```
int get_inputs_visible();
```

Returns:

whether the inputs section of the UI will be visible

Callable From:

- ◇ C++
- ◇ Script

5.46.60 function diguyInteractionMachine::set_input_shortcuts_visible

Prototype:

```
void set_input_shortcuts_visible(int visible);
```

Description:

This function sets whether a "shortcut" for each input will be visible in the UI, before the input text.

Currently the shortcuts are limited to numbers; the first input will be prefixed by "1.", the second by "2.", etc.

Input shortcuts are visible by default.

Arguments:

`visible` pass 1 to make shortcuts visible; 0 to not

Callable From:

- ◇ C++
- ◇ Script

5.46.61 function diguyInteractionMachine::get_input_shortcuts_visible

Prototype:

```
int get_input_shortcuts_visible();
```

Returns:

whether the input shortcuts will be visible in the UI

Callable From:

- ◇ C++
- ◇ Script

5.46.62 function diguyInteractionMachine::set_inputs_font

Prototype:

```
void set_inputs_font(int ui_appearance,  
    const char* font_family,  
    int font_point_size,  
    int font_is_bold = 0,  
    int font_is_italic = 0);
```

Description:

Similar to [set_heading_font\(\)](#), but for inputs instead.

5.46.63 function diguyInteractionMachine::set_inputs_color_bg**Prototype:**

```
void set_inputs_color_bg(int ui_appearance,  
    float r,  
    float g,  
    float b,  
    float a = 1.0f);
```

Description:

Similar to [set_heading_color_bg\(\)](#), but for inputs instead.

5.46.64 function diguyInteractionMachine::set_inputs_color_fg**Prototype:**

```
void set_inputs_color_fg(int ui_appearance,  
    float r,  
    float g,  
    float b,  
    float a = 1.0f);
```

Description:

Similar to [set_heading_color_fg\(\)](#), but for inputs instead.

Deprecated Functions

The following functions are deprecated.

As these functions will disappear in a future version of DI-Guy.

5.46.65 function `diguyInteractionMachine::set_current_settings_index`

Prototype:

```
int set_current_settings_index(int current_settings_index);
```

5.46.66 function diguyInteractionMachine::set_input_response_script

Prototype:

```
void set_input_response_script(const char* script);
```

5.47 diguyLoadManager

diguyLoadManager.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

◆ Deprecated Functions

Link against: libdiguy

5.47.1 class diguyLoadManager

```
class BDI_DECLSPEC_diguy diguyLoadManager
{
public:
    void set_enabled(int enabled);
    int get_enabled();
    int set_minimum_full_update_period(float period);
    float get_minimum_full_update_period();
    int set_zone_far_extent(int zone, float far_extent);
    float get_zone_far_extent(int zone);
    void set_scale_zone_extents_from_view_settings(int do_scale);
    int get_scale_zone_extents_from_view_settings();
    int set_zone_position_update_rate(int zone, int position_update_rate);
    int get_zone_position_update_rate(int zone);
    int set_zone_altitude_update_rate(int zone, int update_rate);
    int get_zone_altitude_update_rate(int zone);
    int set_zone_pose_update_rate(int zone, int pose_update_rate);
    int get_zone_pose_update_rate(int zone);
    int set_zone_motion_lod(int zone, int motion_lod);
    int get_zone_motion_lod(int zone);
    int set_zone_minimum_cpl(int zone, int minimum_cpl);
    int get_zone_minimum_cpl(int zone);
    int set_zone_all_blends_disabled(int zone, int disabled_flag);
    int get_zone_all_blends_disabled(int zone);
    int set_zone_shape_switches_disabled(int zone, int disabled_flag);
    int get_zone_shape_switches_disabled(int zone);
    int set_zone_shape_callbacks_disabled(int zone, int disabled_flag);
    int get_zone_shape_callbacks_disabled(int zone);
    int set_zone_graphics_api_shape_update_disabled(int zone, int disabled_flag);
    int get_zone_graphics_api_shape_update_disabled(int zone);
```

The DI-Guy Load Manager is responsible for adjusting each character's motion level of detail, as well as various update rates, depending on where the character's are in relation to cameras.

5.47.2 function diguyLoadManager::set_enabled

Prototype:

```
void set_enabled(int enabled);
```

Description:

This is the top-level on/off switch for this load manager. By default, load managers are disabled.

Callable From:

- ◇ C++
- ◇ Script

5.47.3 function diguyLoadManager::get_enabled

Prototype:

```
int get_enabled();
```

Description:

Returns whether this load manager is enabled, as set by the [set_enabled\(\)](#) call.

Returns:

1 if this load manager is enabled, 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.47.4 function

diguyLoadManager::set_minimum_full_update_period

Prototype:

```
int set_minimum_full_update_period(float period);
```

Description:

Sets the minimum full update period to be used for all characters. This potentially overrides the update rates otherwise set for the character.

For example, if a character's position update rate were so low that it would not be updated within this minimum update period, it would be forced to update.

The default minimum update period is 0.5 seconds.

Arguments:

<code>period</code>	the minimum update period, in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.47.5 function diguyLoadManager::get_minimum_full_update_period

Prototype:

```
float get_minimum_full_update_period();
```

Returns:

load manager's minimum full update period as set by
[set_minimum_full_update_period\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.47.6 function diguyLoadManager::set_zone_far_extent

Prototype:

```
int set_zone_far_extent(int zone, float far_extent);
```

Description:

This function sets the far extents of load manager zone bounding planes, in terms of their distance from the camera. Zone 0 is between the camera near clipping plane and bounding plane 0. Zone 1 is between bounding plane 0 and bounding plane 1. Zone 2 is between bounding plane 1 and the camera far clipping plane.

The defaults for the zone bounds should work reasonably well in most cases, but some tweaking may be required in some cases.

Defaults:

- zone 0: 10 meters
- zone 1: 20 meters
- zone 2: (camera far plane; cannot be set by this function)

Arguments:

zone	zone to modify
far_extent	new far extent of zone

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.47.7 function diguyLoadManager::get_zone_far_extent

Prototype:

```
float get_zone_far_extent(int zone);
```

Returns:

far extent of zone as set by [set_zone_far_extent\(\)](#), -1.0 on failure

Arguments:

zone	zone to query

Callable From:

- ◇ C++
- ◇ Script

5.47.8 function

diguyLoadManager::set_scale_zone_extents_from_view_settings

Prototype:

```
void set_scale_zone_extents_from_view_settings(int do_scale);
```

Description:

This function sets whether view and camera settings such as field-of-view (FOV) and window size should be taken into account when determining the zone of a character.

The base zone extents are distance-based, but this approach can be problematic if the current camera's FOV is very narrow; a narrow FOV essentially acts like a telephoto lens. In this case a character that is far away, and would therefore be rendered at low resolution, can actually be quite large in the 3D view.

Setting this flag to 1 will cause the FOV to be taken into account when calculating effective zone extents.

In general zone extents assume that the vertical FOV of the view is around 40. Smaller FOVs will scale the thresholds larger so that zone changes happen further out. Likewise larger FOVs will scale the extents smaller; in this case even nearby characters will appear small and therefore need less resolution.

Arguments:

do_scale – pass 1 to enable scaling, 0 to disable

Callable From:

- ◇ C++
- ◇ Script

5.47.9 function diguyLoadManager::get_scale_zone_extents_from_view_settings

Prototype:

```
int get_scale_zone_extents_from_view_settings();
```

Returns:

most recent setting of
[set_scale_zone_extents_from_view_settings\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.47.10 function diguyLoadManager::set_zone_position_update_rate

Prototype:

```
int set_zone_position_update_rate(int zone, int position_update_rate);
```

Description:

This function set the position update rate for the given zone.

The position update rate is the percentage of frames in which the overall position of the character will be updated. Passing 100 will cause character positions to be updated every frame; passing 50 will cause character positions to be updated every other frame.

Defaults:

- zone 0: 100
- zone 1: 100
- zone 2: 100

Arguments:

zone	zone to modify
position_update_rate	new position update rate

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

C++ Example:

```
set_zone_position_update_rate(0, 100);
set_zone_position_update_rate(1, 80);
set_zone_position_update_rate(2, 50);
```

In this example characters in zone 0 will have their positions updated every frame (100%). Characters in zone 1 will have their positions updated every four out of five frames (80%). Characters in zone 2 will have their positions updated every other frame (50%).

5.47.11 function diguyLoadManager::get_zone_position_update_rate

Prototype:

```
int get_zone_position_update_rate(int zone);
```

Returns:

position update rate of zone as set by
`set_zone_position_update_rate()`; -1 on failure

Arguments:

zone	zone to query

Callable From:

◇ C++
◇ Script

5.47.12 function

diguyLoadManager::set_zone_altitude_update_rate

Prototype:

```
int set_zone_altitude_update_rate(int zone, int update_rate);
```

Description:

This function set the altitude update rate for the given zone.

The position update rate is the percentage of frames in which the altitude, or Z position, of the character will be updated.

Passing 100 will cause character altitudes to be updated every frame; passing 50 will cause character altitudes to be updated every other frame.

Defaults:

- zone 0: 100
- zone 1: 100
- zone 2: 100

Arguments:

zone	zone to modify
update_rate	new update rate

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.47.13 function diguyLoadManager::get_zone_altitude_update_rate

Prototype:

```
int get_zone_altitude_update_rate(int zone);
```

Returns:

altitude update rate of zone as set by
[set_zone_altitude_update_rate\(\)](#); -1 on failure

Arguments:

zone	zone to query

Callable From:

- ◇ C++
- ◇ Script

5.47.14 function diguyLoadManager::set_zone_pose_update_rate

Prototype:

```
int set_zone_pose_update_rate(int zone, int pose_update_rate);
```

Description:

This function set the pose update rate for the given zone.

The pose update rate is the percentage of frames in which the pose of the character will be updated. The pose includes all joint angles of the character, but does not include the overall position.

Defaults:

- zone 0: 100
- zone 1: 50
- zone 2: 33

Arguments:

zone zone to modify
pose_update_rate new pose update rate

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.47.15 function diguyLoadManager::get_zone_pose_update_rate

Prototype:

```
int get_zone_pose_update_rate(int zone);
```

Returns:

pose update rate of zone as set by [set_zone_pose_update_rate\(\)](#);
-1 on failure

Arguments:

zone zone to query

Callable From:

◇ C++
◇ Script

5.47.16 function diguyLoadManager::set_zone_motion_lod**Prototype:**

```
int set_zone_motion_lod(int zone, int motion_lod);
```

Description:

This function set the motion LOD for the given zone.

See [diguyCharacter::set_motion_lod\(\)](#) for a discussion of motion LODs.

Defaults:

- zone 0: 1
- zone 1: 2
- zone 2: 3

Arguments:

zone	zone to modify
motion_lod	new motion LOD

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.47.17 function diguyLoadManager::get_zone_motion_lod**Prototype:**

```
int get_zone_motion_lod(int zone);
```

Returns:

motion LOD of zone as set by [set_zone_motion_lod\(\)](#); -1 on failure

Arguments:

zone	zone to query

Callable From:

- ◇ C++
- ◇ Script

5.47.18 function diguyLoadManager::set_zone_minimum_cpl**Prototype:**

```
int set_zone_minimum_cpl(int zone, int minimum_cpl);
```

Description:

See [diguyCharacter::set_minimum_cpl\(\)](#) for a description of what affect this parameter will have on load managed characters.

Arguments:

zone	zone to modify
minimum_cpl	new minimum CPL

Returns:

0 on success, -1 on failure

Callable From:

- ◆ C++
- ◆ Script

5.47.19 function diguyLoadManager::get_zone_minimum_cpl

Prototype:

```
int get_zone_minimum_cpl(int zone);
```

Returns:

minimum CPL of zone as set by `get_zone_minimum_cpl()`; -1 on failure

Arguments:

zone	zone to query

Callable From:

- ◆ C++
- ◆ Script

5.47.20 function `diguyLoadManager::set_zone_all_blends_disabled`

Prototype:

```
int set_zone_all_blends_disabled(int zone, int disabled_flag);
```

Description:

See `diguyCharacter::set_all_blends_disabled()` for a description of what affect this parameter will have on load managed characters.

Arguments:

<code>zone</code>	zone to modify
<code>disabled_flag</code>	pass 1 to disable, 0 to enable

Returns:

0 on success, -1 on failure

Callable From:

- ◆ C++
- ◆ Script

5.47.21 function diguyLoadManager::get_zone_all_blends_disabled

Prototype:

```
int get_zone_all_blends_disabled(int zone);
```

Returns:

whether all blends are disabled in the specified zone; -1 on failure

Arguments:

zone zone to query

Callable From:

- ◇ C++
- ◇ Script

5.47.22 function diguyLoadManager::set_zone_shape_switches_disabled

Prototype:

```
int set_zone_shape_switches_disabled(int zone, int disabled_flag);
```

Description:

See [diguyCharacter::set_shape_switches_disabled\(\)](#) for a description of what affect this parameter will have on load managed characters.

Arguments:

zone zone to modify
disabled_flag pass 1 to disable, 0 to enable

Returns:

0 on success, -1 on failure

Callable From:

- ◆ C++
- ◆ Script

5.47.23 function diguyLoadManager::get_zone_shape_switches_disabled

Prototype:

```
int get_zone_shape_switches_disabled(int zone);
```

Returns:

whether all shape switches are disabled in the specified zone; -1 on failure

Arguments:

zone	zone to query

Callable From:

- ◇ C++
- ◇ Script

5.47.24 function

`diguyLoadManager::set_zone_shape_callbacks_disabled`

Prototype:

```
int set_zone_shape_callbacks_disabled(int zone, int disabled_flag);
```

Description:

See `diguyCharacter::set_shape_callbacks_disabled()` for a description of what affect this parameter will have on load managed characters.

Arguments:

<code>zone</code>	zone to modify
<code>disabled_flag</code>	pass 1 to disable, 0 to enable

Returns:

0 on success, -1 on failure

Callable From:

- ◆ C++
- ◆ Script

5.47.25 function**diguyLoadManager::get_zone_shape_callbacks_disabled****Prototype:**

```
int get_zone_shape_callbacks_disabled(int zone);
```

Returns:

whether all shape callbacks are disabled in the specified zone; -1 on failure

Arguments:

zone zone to query

Callable From:

- ◇ C++
- ◇ Script

5.47.26 function**diguyLoadManager::set_zone_graphics_api_shape_update_disab****Prototype:**

```
int set_zone_graphics_api_shape_update_disabled(int zone, int disabled_flag);
```

Description:

See
[diguyCharacter::set_graphics_api_shape_update_disabled\(\)](#)
 for a description of what affect this parameter will have on load managed
 characters.

Arguments:

zone zone to modify
 disabled_flag pass 1 to disable, 0 to enable

Returns:

0 on success, -1 on failure

Callable From:

- ◆ C++
- ◆ Script

5.47.27 function diguyLoadManager::get_zone_graphics_api_shape_update_disab

Prototype:

```
int get_zone_graphics_api_shape_update_disabled(int zone);
```

Returns:

whether all graphics API shape updates are disabled in the specified zone; -1 on failure

Arguments:

zone	zone to query

Callable From:

- ◇ C++
- ◇ Script

Deprecated Functions

The following functions are deprecated.

As these functions may disappear in a future version of DI-Guy, it is recommended that you remove calls to these functions from your application.

5.48 diguyMotionDirection Type Declaration

diguyMotionDirection.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

5.48.1 enumeration diguyMotionDirection

```
typedef enum {
    DIGUY_MOTION_DIRECTION_NONE = 0,
    DIGUY_MOTION_DIRECTION_FORWARD,
    DIGUY_MOTION_DIRECTION_BACKWARD,
    DIGUY_MOTION_DIRECTION_LEFT,
    DIGUY_MOTION_DIRECTION_RIGHT,

    DIGUY_MOTION_DIRECTION_UP,
    DIGUY_MOTION_DIRECTION_DOWN,
    DIGUY_MOTION_DIRECTION_TURN_LEFT,
    DIGUY_MOTION_DIRECTION_TURN_RIGHT,
    DIGUY_MOTION_DIRECTION_NONLINEAR,
    DIGUY_MOTION_DIRECTION_LINEAR_OFFAXIS,
    DIGUY_MOTION_DIRECTION_INHERIT,
    DIGUY_MOTION_DIRECTION_UNKNOWN,
    DIGUY_MOTION_DIRECTION_COUNT
} diguyMotionDirection;
```

5.48.2 function diguy_motion_direction_to_string**Prototype:**

```
const char* diguy_motion_direction_to_string(diguyMotionDirection direction);
```

Description:

This function converts the passed direction enumeration into a NULL-terminated C-style string.

Returns:

C-style string identifying direction

Callable From:

- ◇ C++
- ◇ Script

5.48.3 function diguy_motion_direction_from_string**Prototype:**

```
diguyMotionDirection diguy_motion_direction_from_string(const char* string);
```

Description:

This function converts the passed C-style string into an enumerated value.

Directions are used in functions such as
[diguyCharacter::get_action_from_description\(\)](#).

Returns:

enumeration value of direction

Callable From:

- ◇ C++
- ◇ Script

5.48.4 function diguy_motion_direction_to_angle

Prototype:

```
float diguy_motion_direction_to_angle(diguyMotionDirection direction);
```

Description:

This function converts the passed direction enumeration into a floating point angle. The returned angle is in degrees.

Currently only the following directions return useful angles:

- ◇ `DIGUY_MOTION_DIRECTION_FORWARD` (angle = 0)
- ◇ `DIGUY_MOTION_DIRECTION_BACKWARD` (angle = 180)
- ◇ `DIGUY_MOTION_DIRECTION_LEFT` (angle = 90)
- ◇ `DIGUY_MOTION_DIRECTION_RIGHT` (angle = 270)

All other directions return an angle of 0.

Returns:

yaw angle of direction in degrees

Callable From:

- ◇ C++
- ◇ Script

5.48.5 function diguy_motion_direction_from_angle

Prototype:

```
diguyMotionDirection diguy_motion_direction_from_angle(float angle,
float allowed_error = 1.0f);
```

Description:

This function converts the passed angle into a direction enumeration.

Currently only the following directions will be returned:

- ◇ `DIGUY_MOTION_DIRECTION_FORWARD` (angle approx. 0)
- ◇ `DIGUY_MOTION_DIRECTION_BACKWARD` (angle approx. 180)
- ◇ `DIGUY_MOTION_DIRECTION_LEFT` (angle approx. 90)
- ◇ `DIGUY_MOTION_DIRECTION_RIGHT` (angle approx. 270)
- ◇ `DIGUY_MOTION_DIRECTION_UNKNOWN` (everything else)

If the passed angle isn't within the specified error amount of any direction's "exact" angle, the value `DIGUY_MOTION_DIRECTION_UNKNOWN` will be returned.

Arguments:

<code>angle</code>	angle for which to retrieve direction, in degrees
<code>allowed_error</code>	how much error there can be from the direction's "exact" angle

Returns:

enumeration value of direction

Callable From:

- ◇ C++
- ◇ Script

5.49 diguyMotionPosture Type Declaration

diguyMotionPosture.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

5.49.1 enumeration diguyMotionPosture

```
typedef enum {
    DIGUY_MOTION_POSTURE_UPRIGHT = 0,
    DIGUY_MOTION_POSTURE_CROUCHED,
    DIGUY_MOTION_POSTURE_PRONE,
    DIGUY_MOTION_POSTURE_SITTING,

    DIGUY_MOTION_POSTURE_UNKNOWN,
    DIGUY_MOTION_POSTURE_ANY,
    DIGUY_MOTION_POSTURE_COUNT
} diguyMotionPosture;
```

5.49.2 function diguy_motion_posture_to_string

Prototype:

```
const char* diguy_motion_posture_to_string(diguyMotionPosture posture);
```

Description:

This function converts the passed posture enumeration into a NULL-terminated C-style string.

Returns:

C-style string identifying posture

Callable From:

- ◇ C++
- ◇ Script

5.49.3 function `diguy_motion_posture_from_string`

Prototype:

```
diguyMotionPosture diguy_motion_posture_from_string(const char* string);
```

Description:

This function converts the passed C-style string into an enumerated value.

Postures are used in functions such as

`diguyCharacter::get_action_from_description()`.

Returns:

enumeration value of posture

Callable From:

- ◇ C++
- ◇ Script

5.50 diguyMotionSpeedType Type Declaration

diguyMotionSpeedType.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

5.50.1 typedef diguyMotionSpeedType

```
typedef int diguyMotionSpeedType;
```

Description:

DI-Guy character speed types.

There are times when it is convenient to refer to actions based on roughly how fast they will move a character. When this is possible, DI-Guy uses "speed types".

DIGUY_MOTION_SPEED_TYPE_UNKNOWN

Error return value.

DIGUY_MOTION_SPEED_TYPE_STILL

Specifies that character/action should not move. For upright characters, this is typically some kind of stand.

DIGUY_MOTION_SPEED_TYPE_SLOW

Specifies that character/action should move slowly. For upright characters, this is typically a shuffle.

DIGUY_MOTION_SPEED_TYPE_MEDIUM

Specifies that character/action should move at medium, or average, speed. For upright characters, this is typically a walk.

DIGUY_MOTION_SPEED_TYPE_FAST

Specifies that character/action should move at a speed faster than a walk, but slower than its fastest speed. For upright characters, this is typically a jog.

DIGUY_MOTION_SPEED_TYPE_FASTEST

Specifies that character/action should move at its fastest speed. For upright characters, this is typically a run.

5.50.2 enumeration

```
enum {
    DIGUY_MOTION_SPEED_TYPE_UNKNOWN = -1,

    DIGUY_MOTION_SPEED_TYPE_STILL = 0,
    DIGUY_MOTION_SPEED_TYPE_SLOW,
    DIGUY_MOTION_SPEED_TYPE_MEDIUM,
    DIGUY_MOTION_SPEED_TYPE_FAST,
    DIGUY_MOTION_SPEED_TYPE_FASTEST,

    DIGUY_MOTION_SPEED_TYPE_COUNT
};
```


5.50.3 function diguy_motion_speed_type_to_string

Prototype:

```
const char* diguy_motion_speed_type_to_string(diguyMotionSpeedType motion_speed_type)
```

Description:

Utility function that converts a [diguyMotionSpeedType](#) enum into a string.

5.50.4 function diguy_motion_speed_type_from_string

Prototype:

```
diguyMotionSpeedType diguy_motion_speed_type_from_string(const char* string);
```

Description:

Utility function that converts a string into a [diguyMotionSpeedType](#) enum.

5.51 diguyMotionVariant Type Declaration

diguyMotionVariant.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

5.51.1 enumeration diguyMotionVariant

```
typedef enum {
    DIGUY_MOTION_VARIANT_NORMAL = 0,
    DIGUY_MOTION_VARIANT_AIM,
    DIGUY_MOTION_VARIANT_DEAD,

    DIGUY_MOTION_VARIANT_ANGRY,
    DIGUY_MOTION_VARIANT_AMBIENT,
    DIGUY_MOTION_VARIANT_ASSAULT,
    DIGUY_MOTION_VARIANT_CURIOUS,
    DIGUY_MOTION_VARIANT_EXERCISE,
    DIGUY_MOTION_VARIANT_FUNKY,
    DIGUY_MOTION_VARIANT_INJURED,
    DIGUY_MOTION_VARIANT_READY,
    DIGUY_MOTION_VARIANT_SICK,
    DIGUY_MOTION_VARIANT_SOCIALIZE,
    DIGUY_MOTION_VARIANT_OTHER,

    DIGUY_MOTION_VARIANT_MISC0,
    DIGUY_MOTION_VARIANT_MISC1,
    DIGUY_MOTION_VARIANT_MISC2,
    DIGUY_MOTION_VARIANT_MISC3,
    DIGUY_MOTION_VARIANT_MISC4,
    DIGUY_MOTION_VARIANT_MISC5,
    DIGUY_MOTION_VARIANT_MISC6,
    DIGUY_MOTION_VARIANT_MISC7,
    DIGUY_MOTION_VARIANT_MISC8,
    DIGUY_MOTION_VARIANT_MISC9,

    DIGUY_MOTION_VARIANT_UNKNOWN,
    DIGUY_MOTION_VARIANT_ANY,
    DIGUY_MOTION_VARIANT_COUNT
} diguyMotionVariant;
```

5.51.2 function diguy_motion_variant_to_string

Prototype:

```
const char* diguy_motion_variant_to_string(diguyMotionVariant variant);
```

Description:

This function converts the passed variant enumeration into a NULL-terminated C-style string.

Returns:

C-style string identifying variant

Callable From:

- ◇ C++
- ◇ Script

5.51.3 function diguy_motion_variant_from_string

Prototype:

```
diguyMotionVariant diguy_motion_variant_from_string(const char* string);
```

Description:

This function converts the passed C-style string into an enumerated value.

Variants are used in functions such as [diguyCharacter::get_action_from_description\(\)](#).

Returns:

enumeration value of variant

Callable From:

- ◇ C++
- ◇ Script

5.52 DI-Guy Graphics API, diguyOglGraphicsLink

diguyOglGraphicsLink.cpp

DI-Guy SDK Version 12.5.2

This file was automatically generated from `diguyOglGraphicsLink.cpp`. Do not edit this file directly; the changes will be lost.

Introduction

DI-Guy 'links' are essentially the same as 'bones' in other APIs.

There are two primary purposes of links:

- ◆ to serve as attachment points for shapes, and
- ◆ to form a skeleton that can be animated using motion data.

The class **diguyOglGraphicsLink** is an example of a [diguyGraphicsLink](#) subclass for an **immediate mode renderer** using the DI-Guy Loader. The example subclass shows how to use the accessors of the [diguyGraphicsLink](#) class and which virtual functions should be overridden for an immediate mode renderer such as OpenGL.

Header files and forward declarations

```
#include "diguyOglGraphicsLink.h"
#include "diguyOglGraphicsShaderProgram.h"
#include "ogl_headers.h"

#include <glm/glm.hpp>
#include <glm/gtc/type_ptr.hpp>
#include <glm/core/func_matrix.hpp>
```

```

#include <assert.h>

#include <libbdilog.h>
#include <diguyApp.h>
#include <diguyCharacter.h>
#include <diguyScenario.h>
#include <diguyViewCamera.h>

/*
 * Top-level static variable that allows for demonstrating two different
 * ways of getting transformation data:
 *
 * - querying for individual floats for translation and orientation, or
 * - querying for the final matrix.
 */
enum {
    FLOATS,
    DOUBLES,
    DERIVED_MATRIX,
    RECURSIVE_DRAW_MATRIX,
    LINEAR_DRAW_MATRIX
};

int s_transformation_mode = DERIVED_MATRIX;

```

Constructor

Description:

Initialize data members of the class to a known state.

Base class data will not be fully initialized until during the Build Stage, so base class accessors should **not** be called here.

```

diguyOglGraphicsLink::diguyOglGraphicsLink(void* internal_data)
:
    diguyGraphicsLink(internal_data)
{
    /*
     * Nothing else to do after calling base class constructor and
     * initializing data members.
     */
}

```

begin_character_draw

Draw Stage

Description:

Override of `diguyGraphicsLink::begin_character_draw()`.

This function is commonly overridden for immediate mode renderers. It will be called once per draw on the position link of the character.

The current transformation state of the renderer should be saved so that it can be restored after

the link's shapes and child links have been drawn. Typically the restore will happen in `end_character_draw()`.

This is typically done, as here, by pushing the current transformation of the character onto a matrix stack.

```
void
diguyOglGraphicsLink::begin_character_draw()
{
    /*
     * Choose if we are doing a linear or recursive draw based off of the
     * graphics API init structure.
     */
    if (s_transformation_mode == DERIVED_MATRIX)
    {
        if (diguyOglGraphicsLink::get_use_linear_traversal_draw())
        {
            s_transformation_mode = LINEAR_DRAW_MATRIX;
        }
        else
        {
            s_transformation_mode = RECURSIVE_DRAW_MATRIX;
        }
    }

    glPushMatrix();

    diguyCharacter* character = get_character();

    if (get_attached_to_link() &(s_transformation_mode == RECURSIVE_DRAW_MATRIX))
    {
        /*
         * This method is deprecated as of DI-Guy 12. If the character has
         * no parent, no need to get the transformation matrix.
         */

        if (character)
        {
            float pscale[3];
            character->get_parent()->get_scale(
                pscale);

            float scale[3];
            character->get_scale(
                scale);

            glScalef(scale[0]/pscale[0], scale[1]/pscale[1], scale[2]/pscale[2]);
        }
    }
    else
    {
        /*
         * For DI-Guy 12 there's no difference in the linear pipeline
         * between drawing parents and children. Read the base position
         * transformation matrix for this link/character.
         */
        float link_matrix[4][4];

        get_transformation_matrix_4x4_ptr(link_matrix);

        glMultTransposeMatrixf((GLfloat*) link_matrix);
    }
}
```

```

/*
 * Apply scale for character (optional).
 */
if (character)
{
    float scale[3];
    character->get_scale(

        if (scale[0] != 1.0f || scale[1] != 1.0f || scale[2] != 1.0f)
        {
            glScalef(scale[0], scale[1], scale[2]);
        }
    }

/*
 * Bump map support.
 */
diguyOglGraphicsShaderProgram* shader_program =
    (diguyOglGraphicsShaderProgram*) diguyApp::get_app()->get_current_bound_shader

if (shader_program &character)
{
    /*
     * If the shader program has object space light and camera
     * uniforms then calculate inverse of the characters rotation
     * matrix and use that to transform the lights/camera into the
     * local space of the character and pass that in as a uniform.
     * These values are needed to make bump mapping work. End users
     * will probably not want to use the light and camera values in
     * the scenario but these are convenient places to stash these
     * values. These have been set by view_set_camera() and
     * view_set_light() in set_up_view.h
     * diguy 12 mistakenly transposed the matrix here.
     */
    glm::mat4x4 inverse_matrix;
    get_inverse_rotation_matrix_4x4_ptr(glm::value_ptr(inverse_matrix), 0);

    if(shader_program->m_bind_location_ufrm_inverse_transform_matrix != -1)
    {
        glUniformMatrix4fv(shader_program->m_bind_location_ufrm_inverse_transform
            1, // number of matrices
            GL_TRUE, // whether to transpose matrix,
            (GLfloat*) ][0]);
    }

    glm::mat4x4 inverse_matrix_no_trans = inverse_matrix;
    inverse_matrix_no_trans[3][0] = 0.0f;
    inverse_matrix_no_trans[3][1] = 0.0f;
    inverse_matrix_no_trans[3][2] = 0.0f;

    if(shader_program->m_bind_location_ufrm_light_dir_os != -1)
    {
        diguyViewLight* light = character->get_scenario()->get_scenario_light(0);
        float light_x, light_y, light_z;
        light->get_direction(

        glm::vec4 light_dir(light_x, light_y, light_z, 0.0f);
        light_dir = glm::normalize(light_dir);
        light_dir = light_dir * inverse_matrix_no_trans;
        glUniform4f(shader_program->m_bind_location_ufrm_light_dir_os,
            light_dir.x, light_dir.y, light_dir.z, light_dir.w);
    }
}

```

```

    }

    if(shader_program->m_bind_location_ufrm_eye_point_os != -1)
    {
        /*
        * Calculate the local space position of the camera
        */
        diguyViewCamera* camera = character->get_scenario()->get_scenario_camera(
        glm::vec4 camera_pos(camera->get_position_x(), camera->get_position_y(), c
        //float char_pos_x, char_pos_y, char_pos_z;
        //character->get_position(
        //camera_pos = camera_pos - glm::vec4(char_pos_x, char_pos_y, char_pos_z,
        //camera_pos = camera_pos * inverse_matrix_no_trans;
        camera_pos = camera_pos * inverse_matrix;
        glUniform3f(shader_program->m_bind_location_ufrm_eye_point_os,
            camera_pos.x, camera_pos.y, camera_pos.z);
    }
}
}
}
}

```

end_character_draw

Draw Stage

Description:

Override of `diguyGraphicsLink::end_character_draw()`.

This function is commonly overridden for immediate mode renderers. It will be called on the position link of the character and typically clears the matrix state and rendering state associated with the character.

This is typically done, as here, by popping the current transformation of the character off the OpenGL matrix stack.

```

void
diguyOglGraphicsLink::end_character_draw()
{
    glPopMatrix();
}

```

push

Draw Stage

Description:

Override of `diguyGraphicsLink::push()`.

This function is commonly overridden for immediate mode renderers. It should save the current transformation state of the renderer so that it can be restored after the link's shapes and child links have been drawn.

This is typically done, as here, by pushing the current transformation state onto a stack.

```
void
diguyOglGraphicsLink::push()
{
    glPushMatrix();
}
```

pop

Draw Stage

Description:

Override of `diguyGraphicsLink::pop()`

The `pop()` function should undo whatever state modifications were made in the `push()` function. For each `push()` call there will be a `pop()` call after the link's shapes and child links have been drawn.

```
void
diguyOglGraphicsLink::pop()
{
    glPopMatrix();
}
```

draw

Draw Stage

Description:

The `draw()` function should perform the necessary actions to apply offsets contained in this link to its shapes and child links.

There are two types of offsets:

- ◆ Static offsets that specify link lengths from the parent link (e.g., distance from shoulder to elbow).
- ◆ Runtime transformations, usually joint rotations, read from DI-Guy motion data.

Since links don't have any actual geometry (shapes and meshes attached to them do), usually no actual drawing occurs in this function.

```
void
diguyOglGraphicsLink::draw()
{
    /*
     * The code below shows two ways to query the link for transformation
     * data: by getting individual values for translations, rotations,
     * etc., or by getting a fully precomputed matrix.
     */
}
```

```

if (s_transformation_mode == LINEAR_DRAW_MATRIX)
{
    diguyOglGraphicsShaderProgram* shader_program = (diguyOglGraphicsShaderProgram*) (

    /*
     * This code shows how to query DI-Guy character links for their
     * local-coordinate matrices, how to load those into the OpenGL
     * link-local-coordinate matrices, and how to load those into
     * the OpenGL transformation matrix or a uniform.
     *
     * Note that linear drawing requires the use of begin and end
     * character draw functions.
     */
    float link_matrix[4][4];
    get_local_transformation_matrix_4x4_ptr();

    /*
     * Multiply by the matrix for this link. If the shader includes
     * ufrm_link_matrix use that instead of changing the modelview
     * matrix.
     *
     * DI-Guy transformation matrices are not in OpenGL's preferred
     * format; they must be transposed using the
     * glMultTransposeMatrixf extension.
     */
    if (shader_program &
        (shader_program->m_bind_location_ufrm_link_matrix != -1))
    {
        glUniformMatrix4fv(shader_program->m_bind_location_ufrm_link_matrix,
            1,          // number of matrices
            GL_TRUE,    // whether to transpose matrix,
            (GLfloat*) link_matrix);
    }
    else
    {
        glMultTransposeMatrixf((GLfloat*) link_matrix);
    }
}
else if (s_transformation_mode == RECURSIVE_DRAW_MATRIX)
{
    /*
     * This code shows how to query DI-Guy character links for their
     * coordinate matrices, and how to load those into the OpenGL
     * transformation matrix.
     *
     * This is effectively the same as building the matrix in code path
     * above.
     */
    float link_matrix[4][4];

    /*
     * Note: if the begin and end character draw calls are not made you
     * would need to do the following:
     */
    /*
    if (get_is_position_link() &!get_attached_to_link())
    {
        // Read the transformation matrix for the 1st link in the
        // character chain. This should be the character's location
        // in the world.
        get_transformation_matrix(link_matrix);
    }
    */
}

```

```

    }
    else ...
    */

    /*
     * Read the local transformation matrix for this link. This is the
     * same as the result of all the glTranslate, glRotates above.
     */
    get_link_transformation_matrix_4x4_ptr();

    /*
     * See note on use of glMultTransposeMatrixf above.
     */
    glMultTransposeMatrixf((GLfloat*) link_matrix);
}
else if ((s_transformation_mode == FLOATS) || (s_transformation_mode == DOUBLES))
{
    /*
     * This code shows how to query DI-Guy for link-local translations
     * and rotations.
     */

    /*
     * First query the base class for the offsets of this link from its
     * parent, and apply those offsets.
     */
    float offset_t[3];
    float offset_r[3];

    get_offset_translation(
    get_offset_rotation(

    glTranslatef(offset_t[0], offset_t[1], offset_t[2]);

    if (offset_r[0] != 0.0f)
        glRotatef(offset_r[0], 0.0, 0.0, 1.0);
    if (offset_r[1] != 0.0f)
        glRotatef(offset_r[1], 1.0, 0.0, 0.0);
    if (offset_r[2] != 0.0f)
        glRotatef(offset_r[2], 0.0, 1.0, 0.0);

    /*
     * Next query for the current values of the dynamic transformations
     * as read from DI-Guy motion data. Apply these transformations.
     */
    if (s_transformation_mode == FLOATS)
    {
        float t[3];
        get_translation(
        glTranslatef(t[0], t[1], t[2]);
    }
    else if (s_transformation_mode == DOUBLES)
    {
        double t[3];
        get_translation_double(
        glTranslated(t[0], t[1], t[2]);
    }

    float r[3];
    float s[3];

```

```

    get_rotation(
    get_scale(

    if (r[0] != 0.0f)
        glRotatef(r[0], 0.0, 0.0, 1.0);
    if (r[1] != 0.0f)
        glRotatef(r[1], 1.0, 0.0, 0.0);
    if (r[2] != 0.0f)
        glRotatef(r[2], 0.0, 1.0, 0.0);

    if ((s[0] != 1.0f) ||
        (s[1] != 1.0f) ||
        (s[2] != 1.0f))
    {
        glScalef(s[0], s[1], s[2]);
    }
}

/*
 * If this is an intersection render pass, set the gl color based
 * on this link's unique color.
 */
if (diguyApp::get_app()->get_render_pass_flags() &DIGUY_RENDER_PASS_INTERSECTION)
{
    unsigned char r, g, b;
    get_unique_color(
    glColor3ub(r, g, b);
}
}

```

Create function definition

Create and return an object of a subclass of [diguyGraphicsLink](#). This function will be called whenever DI-Guy requires a link object.

```

diguyGraphicsLink*
diguyOglGraphicsLink_create_func(void* internal_data)
{
    return new diguyOglGraphicsLink(internal_data);
}

```

5.53 DI-Guy Graphics API, diguyOglGraphicsMaterial

diguyOglGraphicsMaterial.cpp

DI-Guy SDK Version 12.5.2

This file was automatically generated from `diguyOglGraphicsMaterial.cpp`. Do not edit this file directly; the changes will be lost.

Introduction

Materials modify the color and lighting characteristics of meshes. They work in concert with textures to determine the final appearance of visible geometry.

The class **diguyOglGraphicsMaterial** is an example of a **diguyGraphicsMaterial** subclass for an **immediate mode renderer** using the DI-Guy Loader. The example subclass shows how to use the accessors of the **diguyGraphicsMaterial** class and which virtual functions should be overridden for an immediate mode renderer such as OpenGL.

Header files and forward declarations

```
#include "diguyOglGraphicsMaterial.h"
#include <diguyApp.h>
```

Constructor

Description:

Initialize data members of the class to a known state.

Most base class data will not be fully initialized until during the Build Stage, so most base class accessors should **not** be called here.

```
diguyOglGraphicsMaterial::diguyOglGraphicsMaterial(void* internal_data)
:
    diguyGraphicsMaterial(internal_data)
{
    /*
     * Nothing else to do after calling base class constructor and
     * initializing data members.
     */
}
```

build

Build Stage

Description:

Override of **diguyGraphicsMaterial::build()**.

The **build()** function should create renderer objects that will set the proper material state in **bind()** calls made during the Draw Stage.

Pre 11.0.2 this function stored the material state change calls in an OpenGL display list. Display lists, however, have been deprecated as part of the OpenGL spec and now have higher function overhead than just calling the material functions directly.

```
void
```

```
diguyOglGraphicsMaterial::build()
{
    /*
     * Nothing to do.
     */
}
```

unbuild

Unbuild Stage

Description:

Override of `diguyGraphicsMaterial::unbuild()`.

The `unbuild()` function should destroy the objects created in the `build()` function.

```
void
diguyOglGraphicsMaterial::unbuild()
{
    /*
     * Nothing to do.
     */
}
```

bind_now

Draw Stage

Description:

Override of `diguyGraphicsMaterial::bind_now()`.

The `bind_now()` function should do whatever is necessary to change the current state of the renderer to use this material's setting.

This implementation applies the display list created in `build()`, changing OpenGL material settings.

```
void
diguyOglGraphicsMaterial::bind_now()
{
    /*
     * Check to see if this material was the most recently bound material.
     * If so there isn't any need to make additional calls.
     */
    if (diguyOglGraphicsMaterial::get_last_bound_material() == this)
    {
        return;
    }

    /*
     * If this is a render pass for checking for intersections, no need to
     * apply the material.
     */
}
```

```

    */

    if (diguyApp::get_app()->get_render_pass_flags() &DIGUY_RENDER_PASS_INTERSECTION)
        return;

    apply_material();
}

```

apply_material

Description:

Protected utility function that makes the appropriate OpenGL calls to add material properties to the OpenGL state.

```

void
diguyOglGraphicsMaterial::apply_material()
{
    float ambient[4];
    float diffuse[4];
    float specular[4];
    float emission[4];
    float shininess;

    /*
     * Call base class accessors to get material properties.
     */
    get_ambient(ambient);
    get_diffuse(diffuse);
    get_specular(specular);
    get_emission(emission);
    get_shininess(

    /*
     * Make OpenGL calls to apply material properties.
     */
    glMaterialfv(GL_FRONT, GL_AMBIENT, ambient);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR, specular);
    glMaterialfv(GL_FRONT, GL_EMISSION, emission);
    glMaterialf(GL_FRONT, GL_SHININESS, shininess);
}

```

Create function definition

Create and return an object of a subclass of [diguyGraphicsMaterial](#).

```

diguyGraphicsMaterial*
diguyOglGraphicsMaterial_create_func(void* internal_data)
{
    return new diguyOglGraphicsMaterial(internal_data);
}

```

5.54 DI-Guy Graphics API, diguyOglGraphicsMesh

diguyOglGraphicsMesh.cpp

DI-Guy SDK Version 12.5.2

This file was automatically generated from `diguyOglGraphicsMesh.cpp`. Do not edit this file directly; the changes will be lost.

Introduction

Meshes contain the visual geometry of a DI-Guy character. They are attached to shapes, and are associated with materials and textures.

The class **diguyOglGraphicsMesh** is an example of a [diguyGraphicsMesh](#) subclass for an **immediate mode renderer** using the DI-Guy Loader. The example subclass shows how to use the accessors of the [diguyGraphicsMesh](#) class and which virtual functions should be overridden for an immediate mode renderer such as OpenGL.

Header files and forward declarations

```
#include "diguyOglGraphicsMesh.h"
#include "diguyOglGraphicsShaderProgram.h"
#include "diguyOglGraphicsShaderInstance.h"

#include <diguyGraphicsState.h>
#include <stdlib.h>
#include <libbdiolog.h>

#define BUFFER_OFFSET(i) ((char*) NULL + (i))
```

Constructor

Description:

Initialize data members of the class to a known state.

Base class data will not be fully initialized until during the Build Stage, so base class accessors should **not** be called here.

```
diguyOglGraphicsMesh::diguyOglGraphicsMesh(void* internal_data)
:
    diguyGraphicsMesh(internal_data),
    m_is_skinned_mesh(false),
    m_has_tangents(false),
    m_vertex_buffer(0),
    m_index_buffer(0),
    m_gl_index_type(GL_UNSIGNED_SHORT)
{
    /*
     * Nothing else to do after calling base class constructor and
     * initializing data members.
     */
}
```

build*Build Stage***Description:**

Override of `diguyGraphicsMesh::build()`.

The `build()` function should create objects necessary to render the visible geometry of this mesh in the Draw Stage.

This implementation creates OpenGL vertex buffer objects (VBOs) for drawing the visible geometry of the mesh. The contents of the VBO are set based on vertex, normal, and texture index data read using object accessors.

No objects are created if this mesh has no vertices or indices.

```
void
diguyOglGraphicsMesh::build()
{
    /*
     * Don't try to build if there aren't any vertices or indices.
     */
    if (get_vertex_buffer_size() == 0)
        return;

    if (get_index_buffer_size() == 0)
        return;

    /*
     * See if this mesh is skinned.
     */
    m_is_skinned_mesh = get_is_skinned_mesh();
}
```

```

/*
 * See if this mesh has vertex tangents for normal mapping.
 */
int vf = get_vertex_format();
if ((vf == DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_T4) ||
    (vf == DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_T4_I4F_W4))
{
    m_has_tangents = true;
}

/*
 * Determine the index type, and from that the OpenGL index type.
 * Index types are listed in diguyGraphicsVertexFormats.h, in
 * the diguyGraphicsIndexType enumeration.
 */
switch (get_index_type())
{
    case DIGUY_GRAPHICS_INDEX_TYPE_UINT:
        m_gl_index_type = GL_UNSIGNED_INT;
        break;
    case DIGUY_GRAPHICS_INDEX_TYPE_USHORT:
    case DIGUY_GRAPHICS_INDEX_TYPE_D3D_WORD:
        m_gl_index_type = GL_UNSIGNED_SHORT;
        break;
    case DIGUY_GRAPHICS_INDEX_TYPE_UNKNOWN:
    default:
        break;
}

/*
 * Allocate a vertex buffer object (VBO) for vertex data.
 *
 * Bind the new VBO so that subsequent commands are applied to it.
 *
 * If calling this function fails, make sure the glew library is
 * initialized.
 */
if (glGenBuffers == NULL)
{
    static bool s_printed_glew_warning = false;
    if (!s_printed_glew_warning)
    {
        bdi_log_printf(BDI_LOG_ERROR,
            "\n"
            "ERROR: glGenBuffers function is NULL. Make sure the glew library\n"
            "    is initialized after OpenGL is initialized, but before DI-Guy\n"
            "    is initialized.\n"
            "\n");
        s_printed_glew_warning = true;
    }
    return;
}

glGenBuffers(1, ;
glBindBuffer(GL_ARRAY_BUFFER, m_vertex_buffer);

glBufferData(GL_ARRAY_BUFFER,
    get_vertex_buffer_size(),
    get_vertex_buffer_data(),
    GL_STATIC_DRAW);

```

```

/*
 * Make sure the buffer was created successfully.
 */
int n_param_array_object_size;

glGetBufferParameteriv(GL_ARRAY_BUFFER,
    GL_BUFFER_SIZE,
    &n_param_array_object_size);

if (n_param_array_object_size <= 0)
{
    bdi_log_printf(BDI_LOG_ERROR,
        "WARNING: Failed to allocate Vertex Buffer Object of %d bytes for mesh '%s'.\n",
        n_param_array_object_size,
        get_name());
    return;
}
glBindBuffer(GL_ARRAY_BUFFER, 0);

/*
 * Allocate a VBO for index data.
 *
 * Bind the new VBO so that subsequent commands are applied to it.
 */
glGenBuffers(1,
    &m_index_buffer);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, m_index_buffer);

glBufferData(GL_ELEMENT_ARRAY_BUFFER,
    get_index_buffer_size(),
    get_index_buffer_data(),
    GL_STATIC_DRAW);

glGetBufferParameteriv(GL_ELEMENT_ARRAY_BUFFER,
    GL_BUFFER_SIZE,
    &n_index_buffer_size);

/*
 * Make sure the buffer was created successfully.
 */
if (n_index_buffer_size <= 0)
{
    bdi_log_printf(BDI_LOG_ERROR,
        "WARNING: Failed to allocate Index Buffer Object of %d bytes for mesh '%s'.\n",
        n_index_buffer_size,
        get_name());
    return;
}
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

```

unbuild

Unbuild Stage

Description:

Override of `diguyGraphicsMesh::unbuild()`.

The `unbuild()` function should delete the objects created in the `build()` function.

```
void
diguyOglGraphicsMesh::unbuild()
{
    if (m_vertex_buffer)
        glDeleteBuffers(1, &m_vertex_buffer);
    if (m_index_buffer)
        glDeleteBuffers(1, &m_index_buffer);
}
```

draw

Draw Stage

Description:

Override of `diguyGraphicsMesh::draw()`.

The `draw()` function should perform the steps necessary to draw the visible geometry of this link.

In this implementation the VBOs created in `build()` are bound, and other OpenGL state is set up so that the mesh can be drawn.

```
void
diguyOglGraphicsMesh::draw()
{
    if (!m_vertex_buffer || !m_index_buffer)
        return;

    /*
     * Bind the vertex data and index buffers.
     */
    glBindBuffer(GL_ARRAY_BUFFER, m_vertex_buffer);
    glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, m_index_buffer);

    diguyOglGraphicsShaderProgram* shader_program = (diguyOglGraphicsShaderProgram*) get_shader_program();
    diguyOglGraphicsShaderInstance* shader_instance = (diguyOglGraphicsShaderInstance*) get_shader_instance();

    diguyGraphicsState* state = get_current_graphics_state();

    if (state & shader_program)
    {
        int textures_available = state->get_textures_available();

        if (shader_program->m_bind_location_ufrm_bump_map_enabled != -1)
        {
            glUniform1i(shader_program->m_bind_location_ufrm_bump_map_enabled,
                        textures_available & DIGUY_TEXTURE_MAP_MASK_BUMP);
        }
        if (shader_program->m_bind_location_ufrm_specular_map_enabled != -1)
        {
            glUniform1i(shader_program->m_bind_location_ufrm_specular_map_enabled,
                        textures_available & DIGUY_TEXTURE_MAP_MASK_SPECULAR);
        }
        if (shader_program->m_bind_location_ufrm_component_map_enabled != -1)
        {
            glUniform1i(shader_program->m_bind_location_ufrm_component_map_enabled,
                        textures_available & DIGUY_TEXTURE_MAP_MASK_COMPONENT);
        }
    }
}
```

```

    {
        glUniform1i(shader_program->m_bind_location_ufrm_component_map_enabled,
                    !!(textures_available &DIGUY_TEXTURE_MAP_MASK_COMPONENTS));
    }
    if (shader_program->m_bind_location_ufrm_misc_1_map_enabled != -1)
    {
        glUniform1i(shader_program->m_bind_location_ufrm_misc_1_map_enabled,
                    !!(textures_available &DIGUY_TEXTURE_MAP_MASK_MISC_1));
    }

    if (shader_program->m_bind_location_ufrm_misc_2_map_enabled != -1)
    {
        glUniform1i(shader_program->m_bind_location_ufrm_misc_2_map_enabled,
                    !!(textures_available &DIGUY_TEXTURE_MAP_MASK_MISC_2));
    }
}

/*
 * Set base geometry pointers.
 */
glNormalPointer(      GL_FLOAT, get_normal_stride(),          BUFFER_OFFSET(get_normal_stride()*0));
glTexCoordPointer( 2, GL_FLOAT, get_texture_indices_stride(), BUFFER_OFFSET(get_texture_indices_stride()*0));
glVertexPointer(    3, GL_FLOAT, get_vertex_stride(),         BUFFER_OFFSET(get_vertex_stride()*0));

/*
 * If this is a skinned mesh, we need to set some OpenGL state.
 */
if (m_is_skinned_mesh)
{
    if (!shader_program )
    {
        static bool printed_shader_warning = false;

        if (!printed_shader_warning)
        {
            bdi_log_printf(BDI_LOG_WARN,
                           "WARNING: Attempt to draw skinned mesh '%s',\n"
                           "      but mesh has no shader. This can happen if DI-Guy Graphics shade
                           "      program or shader instance subclasses have not been registered.\n"
                           get_name());
            printed_shader_warning = true;
        }
    }

    return;
}

shader_program->set_apply_matrices_flag(1);

/*
 * Set shader attribute pointers.
 */
if (shader_program->m_bind_location_attr_weight_array != -1)
{
    /*
     * From glVertexAttribPointer() doc:
     *
     * "If a non-zero named buffer object is bound to the
     * GL_ARRAY_BUFFER target (see glBindBufferARB) while a
     * generic vertex attribute array is specified, pointer
     * is treated as a byte offset into the buffer object's
     * data store."
     */
}

```

```

        */
        glVertexAttribPointer(shader_program->m_bind_location_attr_weight_array,
            4, // size
            GL_FLOAT, // type
            GL_FALSE, // normalized
            get_vertex_stride(), // stride
            BUFFER_OFFSET(get_link_weight_offset())); // pointer/offset

        glEnableVertexAttribArray(shader_program->m_bind_location_attr_weight_array);
    }

    if (shader_program->m_bind_location_attr_index_array != -1)
    {
        glVertexAttribPointer(shader_program->m_bind_location_attr_index_array,
            4, // size
            GL_FLOAT, // type
            GL_FALSE, // normalized
            get_vertex_stride(), // stride
            BUFFER_OFFSET(get_link_index_offset())); // pointer/offset

        glEnableVertexAttribArray(shader_program->m_bind_location_attr_index_array);
    }
}
else
{
    if (shader_program)
        shader_program->set_apply_matrices_flag(0);
}

if (m_has_tangents & shader_program->m_bind_location_attr_tangent != -1)
{
    glVertexAttribPointer(shader_program->m_bind_location_attr_tangent,
        4, // size
        GL_FLOAT, // type
        GL_TRUE, // normalized??
        get_vertex_stride(), // stride
        BUFFER_OFFSET(get_tangent_offset())); // pointer/offset

    glEnableVertexAttribArray(shader_program->m_bind_location_attr_tangent);
}
// (don't know if we need to do this every time, or just once per frame...)
glEnableClientState(GL_VERTEX_ARRAY);
glEnableClientState(GL_NORMAL_ARRAY);
glEnableClientState(GL_TEXTURE_COORD_ARRAY);

/*
 * DI-Guy's binary format has good sharing of vertices, which means
 * the number of vertices isn't necessarily the same as the number of
 * indices.
 */
//glDrawElements(GL_TRIANGLES, get_vertex_count(), m_gl_index_type, BUFFER_OFFSET(0));

glDrawElements(GL_TRIANGLES, get_index_count(), m_gl_index_type, BUFFER_OFFSET(0));

glDisableClientState(GL_TEXTURE_COORD_ARRAY);
glDisableClientState(GL_NORMAL_ARRAY);
glDisableClientState(GL_VERTEX_ARRAY);

/*
 * Disable attribute arrays if necessary.
 */

```

```

    if (shader_program)
    {
        if (m_is_skinned_mesh)
        {
            if (shader_program->m_bind_location_attr_index_array != -1)
                glDisableVertexAttribArray(shader_program->m_bind_location_attr_index_array);
            if (shader_program->m_bind_location_attr_weight_array != -1)
                glDisableVertexAttribArray(shader_program->m_bind_location_attr_weight_array);
        }

        if (m_has_tangents &
            (shader_program->m_bind_location_attr_tangent != -1))
        {
            glDisableVertexAttribArray(shader_program->m_bind_location_attr_tangent);
        }
    }
    /*
     * Unbind the vertex data and index buffers.
     */
    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
}

```

Create function definition

Create and return an object of a subclass of [diguyGraphicsMesh](#). This function will be called whenever DI-Guy requires a mesh object.

```

diguyGraphicsMesh*
diguyOglGraphicsMesh_create_func(void* internal_data)
{
    return new diguyOglGraphicsMesh(internal_data);
}

```

5.55 DI-Guy Graphics API, diguyOglGraphicsShaderInstance

diguyOglGraphicsShaderInstance.cpp

DI-Guy SDK Version 12.5.2

This file was automatically generated from diguyOglGraphicsShaderInstance.cpp. Do not edit this file directly; the changes will be lost.

Introduction

Unlike *shader programs*, which are shared by many characters, shader instances allow non-shared, character-specific shader state to be maintained. An example is having different uniform variable values for different characters, such as a highlight color that is applied during shading.

See *diguyOglGraphicsShaderProgram* for information on shader programs.

The class **diguyOglGraphicsShaderInstance** is an example of a **diguyGraphicsShaderInstance** subclass for an **immediate mode renderer** using the DI-Guy Loader. The example subclass shows how to use the accessors of the **diguyGraphicsShaderInstance** class and which virtual functions should be overridden for an immediate mode renderer such as OpenGL.

Header files and forward declarations

```
#include <stdlib.h>
#include <stdio.h>

#include "diguyOglGraphicsShaderInstance.h"
#include "diguyOglGraphicsShaderProgram.h"
#include "diguyOglGraphicsShaderTechnique.h"

#include <libbdilog.h>
```

Constructor

Description:

Initialize data members of the class to a known state.

Most base class data will not be fully initialized until during the Build Stage, so most base class accessors should **not** be called here.

```
diguyOglGraphicsShaderInstance::diguyOglGraphicsShaderInstance(void* internal_data)
:
    diguyGraphicsShaderInstance(internal_data)
{
    /*
     * Get a pointer to the shader program this instance is associated with.
     * The shader program will exist before this shader instance is
     * created, so it is safe to call the get_shader_technique() accessor
     * here in the constructor.
     */
    m_shader_technique = dynamic_cast<diguyOglGraphicsShaderTechnique*>(get_shader_technique());
    if (!m_shader_technique)
        return;

    /*
     * Test code for diguyGraphicsShaderInstance variable functions.
     */
    //int num_variables = get_num_variables();
```



```

    //int i;
    //for (i=0; i<num_variables; i++)
    //{
    //    const char* variable_name = get_variable_name_at_index(i);
    //    const char* variable_type = get_variable_type(variable_name);
    //    //
    //    bdi_log_printf(          BDI_LOG_WARN,
    //                    "Variable %d of shader instance is '%s', with type '%s'.\n",
    //                    i,
    //                    variable_name,
    //                    variable_type);
    //}
}

```

bind

Draw Stage

Description:

Override of `diguyGraphicsShaderInstance::bind()`.

Before this `bind()` function is called the `bind()` function for the shader program this instance uses will have been called. In this example this means that the GLSL program object will be loaded.

The `bind()` function can set shader state such as instance level uniform values, etc., that will be used in the shader.

Note that link matrix data should not be set here, but rather in the `bind_link_matrices()` function. The same applies for the link color array and the apply matrices flag.

```

int
diguyOglGraphicsShaderInstance::bind()
{
    /*
     * Nothing to be done in this example. See
     * diguyOglGraphicsShaderProgram::bind().
     */
    return 0;
}

```

unbind

Draw Stage

Description:

Override of `diguyGraphicsShaderInstance::unbind()`.

```

int
diguyOglGraphicsShaderInstance::unbind()
{
    /*

```

```

    * Nothing to be done in this example. See
    * diguyOglGraphicsShaderProgram::unbind().
    */
    return 0;
}

```

Create function definition

Create and return an object of a subclass of [diguyGraphicsShaderInstance](#). This function will be called whenever DI-Guy requires a shader instance object.

```

diguyGraphicsShaderInstance*
diguyOglGraphicsShaderInstance_create_func(void* internal_data)
{
    return new diguyOglGraphicsShaderInstance(internal_data);
}

```

5.56 DI-Guy Graphics API, diguyOglGraphicsShaderProgram

diguyOglGraphicsShaderProgram.cpp

DI-Guy SDK Version 12.5.2

This file was automatically generated from diguyOglGraphicsShaderProgram.cpp. Do not edit this file directly; the changes will be lost.

Introduction

Most rendering environments use some type of shader system that offloads rendering operations from the CPU to a GPU. Terminology varies, but DI-Guy uses the term "shader" to refer to a pair of sub-programs: a vertex shader and a pixel shader (AKA fragment shader in OpenGL).

Shader programs are shared among characters; if two characters use the same shader, the same shader object is used for both. If character-specific data is needed it can be stored in a [diguyGraphicsShaderInstance](#) object.

The class **diguyOglGraphicsShaderProgram** is an example of a [diguyGraphicsShaderProgram](#) subclass for an immediate mode renderer using the DI-Guy Loader. The example subclass shows how to use the accessors of the [diguyGraphicsShaderProgram](#) class and which virtual functions should be overridden for an immediate mode renderer such as OpenGL.

Header files and forward declarations

```
#include <stdlib.h>
#include <stdio.h>
#include <string>
#include <cstring>
#include <math.h>
#include <libbdiolog.h>

// needed for diguyTextureMapType enum
#include <diguyGraphicsTexture.h>
#include <diguyApp.h>
#include <diguyScenario.h>

#include "diguyOglGraphicsShaderProgram.h"
/*
 * Static variable showing whether an OpenGL shader extension is available.
 */
extern GLboolean s_shader_extension_available;
```

Constructor

Description:

Initialize data members of the class to a known state.

Base class data will not be fully initialized until during the Build Stage, so base class accessors should **not** be called here.

```
diguyOglGraphicsShaderProgram::diguyOglGraphicsShaderProgram(void* internal_data)
:
    diguyGraphicsShaderProgram(internal_data),
    m_gl_program_object(0),

    m_bind_location_attr_weight_array(-1),
    m_bind_location_attr_index_array(-1),
```

```

m_bind_location_attr_tangent(-1),
m_bind_location_ufrm_apply_matrices_flag(-1),
m_bind_location_ufrm_link_matrix(-1),
m_bind_location_ufrm_link_matrices(-1),
m_bind_location_ufrm_inverse_transform_matrix(-1),
m_bind_location_ufrm_link_colors(-1),
m_bind_location_ufrm_light_dir_os(-1),
m_bind_location_ufrm_eye_point_os(-1),
m_bind_location_ufrm_textures_enabled(-1),
m_bind_location_ufrm_bump_map_enabled(-1),
m_bind_location_ufrm_specular_map_enabled(-1),
m_bind_location_ufrm_component_map_enabled(-1),
m_bind_location_ufrm_misc_1_map_enabled(-1),
m_bind_location_ufrm_misc_2_map_enabled(-1),
m_bind_location_ufrm_light_array(-1),

m_vertex_shader_loaded(false),
m_gl_vertex_shader_object(-1),

m_pixel_shader_loaded(false),
m_gl_pixel_shader_object(-1),

m_linked(false),
m_include_point_lights(0)
{
    m_vertex_shader_filename[0] = '\0';
    m_pixel_shader_filename[0] = '\0';

    /*
     * Create an OpenGL program object. Note that we don't access any data
     * from the base class here.
     */
    m_gl_program_object = glCreateProgramObjectARB();

    /*
     * Create OpenGL shader objects, one for the vertex shader, one for the
     * pixel (AKA fragment) shader. Delay loading and linking source
     * code until the appropriate virtual functions are called.
     */
    m_gl_vertex_shader_object = glCreateShaderObjectARB(GL_VERTEX_SHADER);
    m_gl_pixel_shader_object = glCreateShaderObjectARB(GL_FRAGMENT_SHADER);

    /*
     * Test code for diguyGraphicsShaderInstance variable functions.
     */
    int num_variables = get_num_variables();
    int i;
    for (i=0; i<num_variables; i++)
    {
        const char* variable_name = get_variable_name_at_index(i);
        const char* variable_type = get_variable_type(variable_name);

        bdi_log_printf(BDI_LOG_WARN,
            "Variable %d of shader instance is '%s', with type '%s'.\n",
            i,
            variable_name,
            variable_type);
    }
    /*
}

```

Destructor**Description:**

Free OpenGL objects allocated in the constructor.

```
diguyOglGraphicsShaderProgram::~diguyOglGraphicsShaderProgram()
{
    glDeleteObjectARB(m_gl_program_object);

    glDeleteObjectARB(m_gl_vertex_shader_object);
    glDeleteObjectARB(m_gl_pixel_shader_object);
}
```

load_vertex_shader_source

Build Stage

Description:

Override of

[diguyGraphicsShaderProgram::load_vertex_shader_source\(\)](#).

The `load_vertex_shader_source( )` function should load the source code of the shader's vertex shader part. The source code should be linked in the `link( )` function instead of here, as usually the pixel shader source needs to be loaded too before the `link( )` call.

This implementation calls the helper function `load_and_compile_shader( )`.

```
int
diguyOglGraphicsShaderProgram::load_vertex_shader_source(const char* filename_base)
{
    /*
     * Use the function
     *   diguyGraphicsShaderProgram::derive_diguy_shader_source_filename()
     * to derive default DI-Guy shader filenames.
     */
    derive_diguy_shader_source_filename(m_vertex_shader_filename,
        511,
        filename_base,
        "glsl",
        "vert",
        1);

    int retval = load_and_compile_shader(m_gl_vertex_shader_object,
        m_vertex_shader_filename,
        m_vertex_shader_loaded);

    return retval;
}
```

load_pixel_shader_source

Build Stage

Description:

Override of `diguyGraphicsShaderProgram::load_pixel_shader_source()`.

The `load_pixel_shader_source()` function should load the source code of the shader's pixel shader part. The source code should be linked in the `link()` function instead of here, as usually the vertex shader source needs to be loaded too before the `link()` call.

This implementation calls the helper function `load_and_compile_shader()`.

```
int
diguyOglGraphicsShaderProgram::load_pixel_shader_source(const char* filename_base)
{
    derive_diguy_shader_source_filename(m_pixel_shader_filename,
        511,
        filename_base,
        "glsl",
        "frag",
        1);

    int retval = load_and_compile_shader(m_gl_pixel_shader_object,
        m_pixel_shader_filename,
        m_pixel_shader_loaded);

    return retval;
}
```

link

Build Stage

Description:

Override of `diguyGraphicsShaderProgram::link()`.

The `link()` function should take the necessary actions to link the shader sub-programs loaded in the `load_vertex_shader_source()` and `load_pixel_shader_source()` functions.

This implementation links the shaders objects. The status of the link is then checked; errors are reported to the DI-Guy log.

```
int
diguyOglGraphicsShaderProgram::link()
{
    int retval = 0;
    m_linked = true;

    /*
     * If shader objects aren't loaded, can't link.
     */
    if (!m_vertex_shader_loaded &
        !m_pixel_shader_loaded)
    {
        bdi_log_printf(BDI_LOG_WARN,
            "WARNING: Failed to link shader '%s' because neither a vertex "
```

```

        "nor pixel shader source file was loaded.\n",
        get_name());
    return -1;
}

/*
 * Attach shader objects to program object.
 */
if (m_vertex_shader_loaded)
    glAttachObjectARB(m_gl_program_object, m_gl_vertex_shader_object);
if (m_pixel_shader_loaded)
    glAttachObjectARB(m_gl_program_object, m_gl_pixel_shader_object);

/*
 * Link the program object.
 */
glLinkProgram(m_gl_program_object);

/*
 * Check the info log to see if there were any problems.
 */
GLint    info_log_length = 0;
GLint    link_status = 0;

glGetObjectParameterivARB(m_gl_program_object, GL_OBJECT_LINK_STATUS_ARB,
glGetObjectParameterivARB(m_gl_program_object, GL_OBJECT_INFO_LOG_LENGTH_ARB, ;

bdi_log_printf(BDI_LOG_INFO,
    "INFO: Shader program '%s' link status: %d\n",
    get_name(),
    link_status);

if (info_log_length > 1)
{
    if (info_log_length > 1000)
        info_log_length = 1000;

    char* info_log = new char[info_log_length];

    glGetInfoLogARB(m_gl_program_object, info_log_length, NULL, info_log);
    info_log[info_log_length-1] = '\0';
    info_log_length--;

    /*
     * Radeon adds some extra newlines.
     */
    while ((info_log[info_log_length-1] == '\n') &
        (info_log_length > 0))
    {
        info_log[info_log_length-1] = '\0';
        info_log_length--;
    }

    if (strstr(info_log, "error"))
    {
        bdi_log_printf(BDI_LOG_WARN,
            "WARNING: Shader program '%s' link info log: %s\n",
            get_name(),
            info_log);
    }
    else

```

```

    {
        bdi_log_printf(BDI_LOG_DEBUG,
            "DEBUG: Shader program '%s' link info log: %s\n",
            get_name(),
            info_log);
    }

    delete[] info_log;
}

glValidateProgramARB(m_gl_program_object);

GLint validated = 0;
glGetObjectParameterivARB(m_gl_program_object, GL_OBJECT_VALIDATE_STATUS_ARB,
if (!validated)
{
    glGetObjectParameterivARB(m_gl_program_object, GL_OBJECT_INFO_LOG_LENGTH_ARB, ;

    char* info_log = new char[info_log_length+1];
    glGetInfoLogARB(m_gl_program_object, info_log_length, info_log);
    info_log[info_log_length] = '\0';

    bdi_log_printf(BDI_LOG_WARN,
        "WARNING: Shader program '%s' validate info log: %s\n",
        get_name(),
        info_log);

    delete[] info_log;

    m_linked = false;
    retval = -1;
}
else
{
    m_linked = true;
    retval = 0;
}

if (m_linked)
{
    /*
     * Use the program so the defaults set below take effect
     */
    glUseProgramObjectARB(m_gl_program_object);

    /*
     * Look up bind locations for diguy_object shader attribute and
     * uniform variables.
     */
    m_bind_location_attr_weight_array = glGetAttribLocation(m_gl_program_object, "a
    m_bind_location_attr_index_array = glGetAttribLocation(m_gl_program_object, "a
    m_bind_location_attr_tangent = glGetAttribLocation(m_gl_program_object, "a
    m_bind_location_ufrm_link_matrix = glGetUniformLocation(m_gl_program_object, "u

    m_bind_location_ufrm_link_matrices = glGetUniformLocation(m_gl_program_object, "u
    if (m_bind_location_ufrm_link_matrices == -1)
    {
        // some drivers aren't happy with [0] above
        m_bind_location_ufrm_link_matrices = glGetUniformLocation(m_gl_program_object,
    }
    m_bind_location_ufrm_inverse_transform_matrix = glGetUniformLocation(m_gl_program

```



```

m_bind_location_ufrm_link_colors = glGetUniformLocation(m_gl_program_object, "ufrm_link_colors");
if (m_bind_location_ufrm_link_colors == -1)
{
    m_bind_location_ufrm_link_colors = glGetUniformLocation(m_gl_program_object, "ufrm_link_colors");
}

m_bind_location_ufrm_light_dir_os = glGetUniformLocation(m_gl_program_object, "ufrm_light_dir_os");
m_bind_location_ufrm_eye_point_os = glGetUniformLocation(m_gl_program_object, "ufrm_eye_point_os");

/*
 * In order to use the same shader program for both skinned and
 * segmented meshes, we need a way to tell the shader whether or not
 * to apply the ufrm_link_matrices values.
 */
m_bind_location_ufrm_apply_matrices_flag = glGetUniformLocation(m_gl_program_object, "ufrm_apply_matrices_flag");

/*
 * m_bind_location_ufrm_textures_enabled is not directly set by
 * DI-Guy. It can be used to manually turn textures on and off in
 * most DI-Guy stock shaders. Its value should be set in bind().
 */
m_bind_location_ufrm_textures_enabled = glGetUniformLocation(m_gl_program_object, "ufrm_textures_enabled");

/*
 * needed by advanced lighting examples
 */
m_bind_location_ufrm_light_array = glGetUniformLocation(m_gl_program_object, "ufrm_light_array");
if (m_bind_location_ufrm_light_array == -1)
{
    m_bind_location_ufrm_light_array = glGetUniformLocation(m_gl_program_object, "ufrm_light_array");
}

/*
 * Various secondary textures.
 */
m_bind_location_ufrm_bump_map_enabled = glGetUniformLocation(m_gl_program_object, "ufrm_bump_map_enabled");
m_bind_location_ufrm_specular_map_enabled = glGetUniformLocation(m_gl_program_object, "ufrm_specular_map_enabled");

m_bind_location_ufrm_component_map_enabled = glGetUniformLocation(m_gl_program_object, "ufrm_component_map_enabled");
m_bind_location_ufrm_misc_1_map_enabled = glGetUniformLocation(m_gl_program_object, "ufrm_misc_1_map_enabled");
m_bind_location_ufrm_misc_2_map_enabled = glGetUniformLocation(m_gl_program_object, "ufrm_misc_2_map_enabled");

/*
 * Hook up the uniform with the desired texture unit. Update a bit
 * field that diguy needs to keep track of which textures to bind
 * for a shader.
 */
int textures_used_mask = 0;
int val = glGetUniformLocation(m_gl_program_object, "ufrm_diffuse_map");
if (val != -1)
{
    glUniform1i(val, (int) DIGUY_TEXTURE_MAP_TYPE_DIFFUSE);
    textures_used_mask |= DIGUY_TEXTURE_MAP_MASK_DIFFUSE;
}

val = glGetUniformLocation(m_gl_program_object, "ufrm_specular_map");
if (val != -1)
{
    glUniform1i(val, (int) DIGUY_TEXTURE_MAP_TYPE_SPECULAR);
    textures_used_mask |= DIGUY_TEXTURE_MAP_MASK_SPECULAR;
}

```

```

    }

    val = glGetUniformLocation(m_gl_program_object, "ufrm_bump_map");
    if (val != -1)
    {
        glUniform1i(val, (int) DIGUY_TEXTURE_MAP_TYPE_BUMP);
        textures_used_mask |= DIGUY_TEXTURE_MAP_MASK_BUMP;
    }

    val = glGetUniformLocation(m_gl_program_object, "ufrm_bump_map");
    if (val != -1)
    {
        glUniform1i(val, (int) DIGUY_TEXTURE_MAP_TYPE_BUMP);
        textures_used_mask |= DIGUY_TEXTURE_MAP_MASK_BUMP;
    }

    val = glGetUniformLocation(m_gl_program_object, "ufrm_component_map");
    if (val != -1)
    {
        glUniform1i(val, (int) DIGUY_TEXTURE_MAP_TYPE_COMPONENTS);
        textures_used_mask |= DIGUY_TEXTURE_MAP_MASK_COMPONENTS;
    }

    val = glGetUniformLocation(m_gl_program_object, "ufrm_component_ramp");
    if (val != -1)
    {
        glUniform1i(val, DIGUY_TEXTURE_MAP_TYPE_COMPONENT_RAMP);
    }

    val = glGetUniformLocation(m_gl_program_object, "ufrm_misc_1_map");
    if (val != -1)
    {
        glUniform1i(val, (int) DIGUY_TEXTURE_MAP_TYPE_MISC_1);
        textures_used_mask |= DIGUY_TEXTURE_MAP_MASK_MISC_1;
    }

    val = glGetUniformLocation(m_gl_program_object, "ufrm_misc_2_map");
    if (val != -1)
    {
        glUniform1i(val, (int) DIGUY_TEXTURE_MAP_TYPE_MISC_2);
        textures_used_mask |= DIGUY_TEXTURE_MAP_MASK_MISC_2;
    }
    /*
     * let diguyGraphicsShaderProgram know what textures are available
     */
    set_textures_used_bitmask(textures_used_mask);
}

glUseProgramObjectARB(0);

return retval;
}

```

get_attrb_bind_location

Build Stage

Description:

Override of `diguyGraphicsShaderProgram::get_attrbind_location()`. Lets DI-Guy core engine know more about this shader's capabilities.

```
int
diguyOglGraphicsShaderProgram::get_attrbind_location(const char* name)
{
    return glGetUniformLocation(m_gl_program_object, name);
}
```

get_uniform_bind_location

Build Stage

Description:

Override of `diguyGraphicsShaderProgram::get_uniform_bind_location()`. Lets diguy core engine store uniform locations internally to allow setting material and shape uniforms in the future via `bind_uniform_with_location()`.

```
int
diguyOglGraphicsShaderProgram::get_uniform_bind_location(const char* name)
{
    return glGetUniformLocation(m_gl_program_object, name);
}
```

bind

Draw Stage

Description:

Override of `diguyGraphicsShaderProgram::bind()`.

The `bind()` function should perform the necessary steps to load this shader program into the GPU. It can also set other shader state such as program level uniform values, etc., that will be used in the shader.

Description:

Demonstrates using weapon fire flashes to create dynamic point lights

```
int
diguyOglGraphicsShaderProgram::bind()
{
    if (m_gl_program_object == 0)
        return -1;

    /*
     * Bind the shader program object so that subsequent shader calls such
     * as bind_link_matrices() use this program.
     */
    glUseProgramObjectARB(m_gl_program_object);
}
```

```

    if (m_bind_location_ufrm_textures_enabled != -1)
    {
        glUniform1i(m_bind_location_ufrm_textures_enabled, 1);
    }

    update_light_uniforms();

    return 0;
}

void
diguyOglGraphicsShaderProgram::update_light_uniforms()
{
    diguyScenario * scenario = diguyApp::get_app()->get_scenario_at_index(0);

    // don't send multiple times a frame
    if(fabs(m_last_light_update_time - scenario->get_t()) < .001){
        return;
    }
    m_last_light_update_time = scenario->get_t();

    int num_effects = scenario->get_num_active_fire_effects();

    // Note: light index 0 is reserved for the global directional light.
    // This makes all the logic and indexing slightly wonky.
    float light_positions[8][4];
    light_positions[0][0] = 0.0f; light_positions[0][1] = 0.0f; light_positions[0][2] = 0
    light_positions[0][3] = 1.0f;

    int i;
    for(i = 1; i < (num_effects+1) &i < 8; i++)
    {
        float radius = 0;
        float color[4];
        float falloff_r = 1;
        float falloff_rsq = 1;
        scenario->get_weapon_fire_effect_data(i-1,
            i][0], i][1], i][2],

        color[3] = 0;
        light_positions[i][3] = 1;
        glLightfv(GL_LIGHT0+i, GL_DIFFUSE,
        glLightfv(GL_LIGHT0+i, GL_POSITION, i][0]);
        glLightf (GL_LIGHT0+i, GL_CONSTANT_ATTENUATION, falloff_r);
        glLightf (GL_LIGHT0+i, GL_QUADRATIC_ATTENUATION, falloff_rsq);
        glLightf (GL_LIGHT0+i, GL_SPOT_EXPONENT, 0);
    }
    // zero out any additional lights
    for(; i < 8; i++)
    {
        float color[4]={0,0,0,0};
        glLightfv(GL_LIGHT0+i, GL_DIFFUSE,
        glLightf (GL_LIGHT0+i, GL_CONSTANT_ATTENUATION, 1);
        glLightf (GL_LIGHT0+i, GL_QUADRATIC_ATTENUATION, 1);
        glLightf (GL_LIGHT0+i, GL_SPOT_EXPONENT, 0);
    }
    // this uniform is required for bump mapped characters
    if (m_bind_location_ufrm_light_array != -1)
    {

```

```

        glUniform4fv(m_bind_location_ufrm_light_array, 8, 0)[0]);
    }
}

```

unbind

Draw Stage

Description:

Override of `diguyGraphicsShaderProgram::unbind()`.

The `unbind()` function should perform the necessary steps to unload this shader program from the GPU.

```

int
diguyOglGraphicsShaderProgram::unbind()
{
    if (m_gl_program_object == 0)
        return -1;

    m_last_matrix_data = NULL;

    /*
     * Unbind the shader program object so that subsequent shader calls
     * don't use this program.
     */
    glUseProgramObjectARB(0);

    return 0;
}

```

reload

Description:

Override of `diguyGraphicsShaderProgram::reload()`.

The `reload()` function should reload previously loaded shader source code, and relink the program. It can be called during any stage, but is usually called during the Update Stage.

```

int
diguyOglGraphicsShaderProgram::reload()
{
    if (m_linked)
    {
        m_linked = false;
    }

    /*
     * Reload the vertex shader using the previously derived filenames.
     */
    int retval = load_and_compile_shader(m_gl_vertex_shader_object,
        m_vertex_shader_filename,
        m_vertex_shader_loaded);
}

```

```

    if (retval != 0)
    {
        bdi_log_printf(BDI_LOG_WARN,
            "WARNING: Failed to reload vertex shader source file '%s' for "
            "shader program '%s'.\n",
            m_vertex_shader_filename,
            get_name());
        return -1;
    }

    /*
     * Reload the pixel shader using the previously derived filenames.
     */
    retval = load_and_compile_shader(m_gl_pixel_shader_object,
        m_pixel_shader_filename,
        m_pixel_shader_loaded);

    if (retval != 0)
    {
        bdi_log_printf(BDI_LOG_WARN,
            "WARNING: Failed to reload pixel shader source file '%s' for "
            "shader program '%s'.\n",
            m_pixel_shader_filename,
            get_name());
        return -1;
    }

    /*
     * Relink the shader program.
     */
    retval = link();

    if (retval != 0)
    {
        bdi_log_printf(BDI_LOG_WARN,
            "WARNING: Failed to relink shader program '%s'.\n",
            get_name());
        return -1;
    }

    return 0;
}

```

load_and_compile_shader

Description:

Protected utility function for loading and compiling a shader object.

```

int
diguyOglGraphicsShaderProgram::load_and_compile_shader(GLhandleARB shader_object,
    const char* shader_filename,
    bool&shader_loaded)
{
    /*
     * Needed to suport reloading.
     */
    if (shader_loaded)

```

```

        glDetachObjectARB(m_gl_program_object, shader_object);

int retval = 0;

shader_loaded = false;

/*
 * Load source file using utility function load_shader_source().
 */
char* source_text = load_shader_source(shader_filename);
if (!source_text)
{
    /*
     * Note that this might not be an error if the shader file doesn't
     * exist, or does exist but contains no code.
     */
    return 0;
}

/*
 * Set up a #define for gamma correction factor.
 */
const char* gamma_defines = NULL;

// Set up #defines to enable various shader features before compiling
std::string my_defines = "";
std::string my_version = "#version 120\n";

if (diguyApp::get_app()->get_use_sRGB_textures())
{
    my_defines = "#define GAMMA_FACTOR 2.2 \n";
}
else
{
    my_defines = "#define GAMMA_FACTOR 1.0 \n";
}

// the point light support is slow it's worth having a second set of shaders
// and just activating when needed.
if(m_include_point_lights){
    my_defines += "#define POINT_LIGHT_SUPPORT \n";
}
if(get_use_4x3_shader_matrix_data()){
    my_defines += "#define MATRIX_4x3_SUPPORT\n";
}

my_defines += "#define COMPONENT_MAP_SUPPORT \n";

my_defines += "#define SHADOW_SUPPORT \n";

const char * shader_strings[4];
shader_strings[0] = my_version.c_str();
shader_strings[1] = my_defines.c_str();
// reset the line number for error handling
shader_strings[2] = "#line 0 0\n";
shader_strings[3] = source_text;

glShaderSource(shader_object, 4, shader_strings, NULL);

/*
 * Compile shader_object.

```

```

    */
{
    glCompileShader(shader_object);

    /*
     * Do this to see if the compile succeeded.
     */
    GLchar* info_log;
    GLint   info_log_length = 0;
    GLint   compile_status = 0;

    glGetObjectParameterivARB(shader_object, GL_OBJECT_COMPILE_STATUS_ARB,
    glGetObjectParameterivARB(shader_object, GL_OBJECT_INFO_LOG_LENGTH_ARB, ;

    bdi_log_printf(BDI_LOG_INFO,
        "INFO: Shader source '%s' compile status: %d\n",
        shader_filename,
        compile_status);

    if (info_log_length > 1)
    {
        info_log = (GLchar*) malloc(info_log_length);

        glGetInfoLogARB(shader_object, info_log_length, NULL, info_log);

        info_log[info_log_length-1] = '\0';
        info_log_length--;

        /*
         * Radeon adds some extra newlines.
         */
        while ((info_log[info_log_length-1] == '\n') &
            (info_log_length > 0))
        {
            info_log[info_log_length-1] = '\0';
            info_log_length--;
        }

        /*
         * Take into account that Radeon compiler returns an info log
         * even on success.
         */
        if ((strstr(info_log, "error")) &
            (!strstr(info_log, "No errors")))
        {
            bdi_log_printf(BDI_LOG_WARN,
                "WARNING: Shader source file '%s' compile info: %s\n",
                shader_filename,
                info_log);
            retval = -1;
        }
        else
        {
            bdi_log_printf(BDI_LOG_DEBUG,
                "DEBUG: Shader source file '%s' compile info: %s\n",
                shader_filename,
                info_log);
            retval = 0;
        }

        free(info_log);
    }
}

```



```

    }
    else
    {
        retval = 0;
    }
}

/*
 * Free source.
 */
free_shader_source(source_text);

shader_loaded = true;
return retval;
}

```

load_shader_source

Description:

Protected utility function for loading shader program source code.

```

char*
diguyOglGraphicsShaderProgram::load_shader_source(const char* shader_filename)
{
    FILE* fp = NULL;
    char* shader_source;
    int length;

    /*
     * Try shader_filename straight up.
     */
    fp = fopen(shader_filename, "r");

    if (!fp)
    {
        bdi_log_printf(BDI_LOG_WARN, "WARNING: Failed to load shader source "
            "file '%s'.\n",
            shader_filename);
        return NULL;
    }

    /*
     * Check the length of the file.
     */
    fseek(fp, 0, SEEK_END);
    length = ftell(fp);
    fseek(fp, 0, SEEK_SET);

    /*
     * If length is less than some small amount (less than a legal shader
     * program length), decide that the shader program isn't valid and
     * return NULL.
     */
    const int MIN_LENGTH = 24;
    if (length < MIN_LENGTH)
    {
        bdi_log_printf(BDI_LOG_DEBUG,
            "DEBUG: Shader source file '%s' length is less than %d, and will "

```

```

        "not be loaded.\n",
        shader_filename,
        MIN_LENGTH);
    fclose(fp);

    return NULL;
}

shader_source = new char[length+1];
memset(shader_source, 0, length);

fread(shader_source, 1, length, fp);
shader_source[length] = '\0'; // make it a regular C string so str* functions work
fclose(fp);

return shader_source;
}

```

free_shader_source

Description:

Protected utility function for freeing shader program source code allocated during load_shader_source().

```

void
diguyOglGraphicsShaderProgram::free_shader_source(char* shader_source)
{
    delete[] shader_source;
}

```

bind_link_matrices

Draw Stage

Description:

Override of diguyGraphicsShaderInstance::bind_link_matrices().

The bind_link_matrices() function should pass link transformation matrices generated from DI-Guy motion data to the GPU.

```

int
diguyOglGraphicsShaderProgram::bind_link_matrices(int num_mats,
    const float* mat_data)
{
    if (m_gl_program_object == 0)
        return -1;

    if (m_bind_location_ufrm_link_matrices == -1)
        return -1;

    if(mat_data == m_last_matrix_data){
        return 0;
    }
}

```

```

    m_last_matrix_data = mat_data;

    /*
     * Pass the link matrices generated from DI-Guy motion data to the GPU.
     */
    if(get_use_4x3_shader_matrix_data())
    {
        glUniformMatrix4x3fv(m_bind_location_ufrm_link_matrices,    // uniform var location
                             num_mats,    // number of matrices
                             GL_TRUE,    // whether to transpose matrix
                             mat_data);    // pointer to matrix data
    }
    else{
        glUniformMatrix4fv(m_bind_location_ufrm_link_matrices,    // uniform var location
                           num_mats,    // number of matrices
                           !get_transposed_matrix_data(),    // whether to transpose matrix
                           mat_data);    // pointer to matrix data
    }

    return 0;
}

```

bind_color_array

Draw Stage

Description:

Override of diguyGraphicsShaderInstance::bind_color_array().

The bind_color_array() function should pass the link colors array generated by DI-Guy to the GPU.

```

int
diguyOglGraphicsShaderProgram::bind_color_array(int num_colors,
const float* color_data)
{
    if (m_bind_location_ufrm_link_colors == -1)
        return -1;

    glUniform4fv(m_bind_location_ufrm_link_colors,
                 num_colors,    // number of colors
                 color_data);    // pointer to color data

    return 0;
}

```

set_apply_matrices_flag

Draw Stage

Description:

Override of diguyGraphicsShaderInstance::set_apply_matrices_flag().

The `set_apply_matrices_flag( )` function should pass the specified `apply_matrices_flag` to the GPU, typically through a uniform variable.

Note that this implementation is specific to the reference DI-Guy shader implementation, which checks the flag to determine if the matrices passed in the `bind_link_matrices( )` call will be applied.

The matrices should be applied for skinned appearances, not applied for unskinned/segmented appearances. The reference shader will do the right thing based on the value of this flag. Custom shader code should do something similar.

```
void
diguyOglGraphicsShaderProgram::set_apply_matrices_flag(int apply_matrices_flag)
{
    if (m_bind_location_ufrm_apply_matrices_flag == -1)
        return;

    glUniform1f(m_bind_location_ufrm_apply_matrices_flag,
        (float) apply_matrices_flag);
}
```

bind_uniform_with_location

Draw Stage

Description:

Used to send float values to the shader. Should be used when sending DI-Guy shader constants to the shader. This will only happen if `get_uniform_bind_location( )` is implemented and returns something other than `-1`. This function must be implemented for advanced DI-Guy materials to work.

```
int
diguyOglGraphicsShaderProgram::bind_uniform_with_location(int location,
    const char* name,
    int value_size,
    float* values)
{
    if (value_size == 1)
        glUniform1f(location, values[0]);
    else if (value_size == 2)
        glUniform2f(location, values[0], values[1]);
    else if (value_size == 3)
        glUniform3f(location, values[0], values[1], values[2]);
    else if (value_size == 4)
        glUniform4f(location, values[0], values[1], values[2], values[3]);

    return 0;
}
```

bind_uniform_with_location

Draw Stage

Description:

Used to send int values to the shader.

```
int
diguyOglGraphicsShaderProgram::bind_uniform_with_location(int location,
    const char* name,
    int value_size,
    int* values)
{
    if (value_size == 1)
        glUniform1i(location, values[0]);
    else if (value_size == 2)
        glUniform2i(location, values[0], values[1]);
    else if (value_size == 3)
        glUniform3i(location, values[0], values[1], values[2]);
    else if (value_size == 4)
        glUniform4i(location, values[0], values[1], values[2], values[3]);

    return 0;
}
```

Create function definition

Create and return an object of a subclass of [diguyGraphicsShaderProgram](#). This function will be called whenever DI-Guy requires a shader program object.

```
diguyGraphicsShaderProgram*
diguyOglGraphicsShaderProgram_create_func(void* internal_data)
{
    if (!s_shader_extension_available)
    {
        bdi_log_printf(BDI_LOG_ERROR,
            "ERROR: Cannot create shader program; OpenGL shader extension "
            "not available.\n");
        return NULL;
    }

    return new diguyOglGraphicsShaderProgram(internal_data);
}
```

diguyOglGraphicsShaderTechnique.cpp**DI-Guy SDK Version 12.5.2**

This file was automatically generated from diguyOglGraphicsShaderTechnique.cpp. Do not edit this file directly; the changes will be lost.

```
#include <string>

#include <diguyScenario.h>
#include <diguyApp.h>

#include "diguyOglGraphicsShaderTechnique.h"
#include "diguyOglGraphicsShaderProgram.h"
```

```

diguyOglGraphicsShaderTechnique::diguyOglGraphicsShaderTechnique(void* internal_data) :
    diguyGraphicsShaderTechnique(internal_data)
{
    ;
}

diguyOglGraphicsShaderTechnique::~diguyOglGraphicsShaderTechnique()
{
    ;
}

//virtual
diguyGraphicsShaderProgram *
diguyOglGraphicsShaderTechnique::pick_shader_program(const diguyShaderPickDescription &desc)
{
    /* You can just do this if you want default behavior
    * return diguyGraphicsShaderProgram::pick_shader_program(desc);
    */

    if(desc.m_current_render_flags &DIGUY_RENDER_PASS_INTERSECTION) {
        return get_intersection_shader();
    }
    else if(desc.m_current_render_flags &DIGUY_RENDER_PASS_SHADOW ){
        return get_shadow_pass_shader();
    }
    else
    {
        // automatic shader LOD are specified
        if(desc.m_shader_lod == -1)
        {
            //The simple case is just to do:
            //return get_shader_by_distance( desc.m_render_distance, desc.m_max_quality_lo

            int index = get_program_index_by_distance(desc.m_render_distance / desc.m_shad
            if(index != -1)
            {
                diguyScenario * scenario = diguyApp::get_app()->get_scenario_at_index(0);
                int num_effects = scenario->get_num_active_fire_effects();
                if(num_effects > 0){
                    diguyGraphicsShaderProgram * program = m_shaders_with_point_lights[in
                    if(program){
                        return program;
                    }
                }
                else{
                    return get_shader_lod_program_at_index(index);
                }
            }
        }
        else
        {
            if(desc.m_shader_lod > get_num_shader_lod_programs()){
                return get_shader_lod_program_at_index(get_num_shader_lod_programs()-1);
            }
            else{
                return get_shader_lod_program_at_index(desc.m_shader_lod-1);
            }
        }
    }
    // falls back to the default behavior

```

```

        return NULL;
    };

    void
    diguyOglGraphicsShaderTechnique::build()
    {
        int i;
        for(i = 0; i < get_num_shader_lod_programs(); i++)
        {
            diguyOglGraphicsShaderProgram* program =
                dynamic_cast<diguyOglGraphicsShaderProgram *>(get_shader_lod_program_at_index(i));

            diguyOglGraphicsShaderProgram * new_program = NULL;
            // the base program defaults to off so we can detect and turn on
            if(program &program->m_bind_location_ufrm_light_array != -1 ||
               program->has_additional_feature("POINT_LIGHT_SUPPORT"))
            {
                std::string new_program_name = program->get_name();
                new_program_name += ":point_lights";
                //printf("Compiling additional shader '%s' \n", new_program_name.c_str());
                new_program = dynamic_cast<diguyOglGraphicsShaderProgram *>(find_or_create_shader(new_program_name));
                new_program->m_include_point_lights = 1;
                new_program->load();
            }

            m_shaders_with_point_lights.push_back(new_program);
            if(new_program){
                add_additional_shader(new_program);
            }
        }
    }
}

```

Create function definition

Create and return an object of a subclass of [diguyGraphicsShaderTechnique](#). This function will be called whenever DI-Guy requires a shader technique object.

```

diguyGraphicsShaderTechnique*
diguyOglGraphicsShaderTechnique_create_func(void* internal_data)
{
    return new diguyOglGraphicsShaderTechnique(internal_data);
}

```

5.57 DI-Guy Graphics API, diguyOglGraphicsShape

diguyOglGraphicsShape.cpp

DI-Guy SDK Version 12.5.2

This file was automatically generated from diguyOglGraphicsShape.cpp. Do not edit this file directly; the changes will be lost.

Introduction

Shapes are the high-level containers of visible geometry. They are attached to links, and contain zero or more mesh objects.

The class **diguyOglGraphicsShape** is an example of a **diguyGraphicsShape** subclass for an **immediate mode renderer** using the DI-Guy Loader. The example subclass shows how to use the accessors of the **diguyGraphicsShape** class and which virtual functions should be overridden for an immediate mode renderer such as OpenGL.

Header files and forward declarations

```
#include "diguyOglGraphicsShape.h"
#include "ogl_headers.h"

#include <stdlib.h>
#include <diguyApp.h>
#include <diguyGraphicsShaderProgram.h>
#include <diguyGraphicsTexture.h>

GLuint diguyOglGraphicsShape::sm_default_variation_texture_object = 0;
```

Constructor

Description:

Initialize data members of the class to a known state.

Base class data will not be fully initialized until the Build Stage is done, so base class accessors should **not** be called here.

```
diguyOglGraphicsShape::diguyOglGraphicsShape(void* internal_data)
:
    diguyGraphicsShape(internal_data),
    m_display_list(0),
    m_unique_texture_object(0)
{
    /*
     * Nothing else to do after calling base class constructor and
     * initializing data members.
     */
}
```

build

Build Stage

Description:

Override of **diguyGraphicsShape::build()**.

The `build()` function should create the necessary objects that will apply offset data of this shape from its parent link during the Draw Stage.

This implementation checks to see if this shape has any offsets from its link. If so, it defines a display list that will apply the offsets each time the shape is drawn. Shape offsets are always static, so they can safely be encoded in a display list. Note that no visible geometry is built. This will be done when the mesh objects attached to this shape have their `build()` functions called.

No display list is created if this shape has no offsets.

```
void
diguyOglGraphicsShape::build()
{
    if (get_has_offset_translation() ||
        get_has_offset_rotation() ||
        get_has_offset_scale())
    {
        m_display_list = glGenLists(1);
        if (m_display_list == 0)
            return;

        /*
         * Compile the transformations into m_display_list.
         */
        glNewList(m_display_list, GL_COMPILE);
        {
            if (get_has_offset_translation())
            {
                float offset_translation[3];
                get_offset_translation(on[0],
                                      on[1],
                                      on[2]);

                if ((offset_translation[0] != 0.0f) ||
                    (offset_translation[1] != 0.0f) ||
                    (offset_translation[2] != 0.0f))
                {
                    glTranslatef(offset_translation[0],
                                offset_translation[1],
                                offset_translation[2]);
                }
            }

            if (get_has_offset_rotation())
            {
                float offset_rotation [3];
                get_offset_rotation ([0],
                                    [1],
                                    [2]);

                if (offset_rotation [0] != 0.0f)
                    glRotatef(offset_rotation [0], 0.0, 0.0, 1.0);
                if (offset_rotation [1] != 0.0f)
                    glRotatef(offset_rotation [1], 1.0, 0.0, 0.0);
                if (offset_rotation [2] != 0.0f)
                    glRotatef(offset_rotation [2], 0.0, 1.0, 0.0);
            }
        }
    }
}
```

```

        if (get_has_offset_scale())
        {
            float offset_scale[3];
            get_offset_scale(

                ;

            if ((offset_scale[0] != 1.0f) ||
                (offset_scale[1] != 1.0f) ||
                (offset_scale[2] != 1.0f))
            {
                glScalef(offset_scale[0],
                        offset_scale[1],
                        offset_scale[2]);
            }
        }
    }
    glEndList();
}

```

unbuild

Unbuild Stage

Description:

Override of `diguyGraphicsShape::unbuild()`.

The `unbuild()` function should destroy the objects created in the `build()` function.

This implementation destroys the display list created in the `build()` function.

```

void
diguyOglGraphicsShape::unbuild()
{
    if (m_display_list != 0)
        glDeleteLists(m_display_list, 1);

    m_display_list = 0;

    if(m_unique_texture_object){
        glDeleteTextures(1, _object);
    }
    m_unique_texture_object = 0;
}

```

draw

Draw Stage

Description:

Override of `diguyGraphicsShape::draw()`.

The `draw()` function needs to apply offsets created in the Build Stage so that subsequent drawing of child meshes will be correctly placed.

This implementation applies the display list created in the Build Stage. This will change the OpenGL matrix stack to include this shape's offsets, which will change the position and orientation of meshes attached to this shape.

For this immediate mode example all of the visible geometry is in mesh objects attached to this shape, so the lod argument isn't needed.

```
void
diguyOglGraphicsShape::draw(int lod)
{
    if (m_display_list != 0)
        glCallList(m_display_list);

    /*
     * If this is a render pass for finding intersections, look up and apply
     * the unique color of this shape.
     */
    if (diguyApp::get_app()->get_render_pass_flags() & DIGUY_RENDER_PASS_INTERSECTION)
    {
        unsigned char r, g, b;
        get_unique_color(
            glColor3ub(r, g, b);
        )
    }
    /*
     * If this shader support the variation texture system bind either the degenerate tex
     * or the unique one.
     */
    diguyGraphicsShaderProgram * shader = diguyApp::get_app()->get_current_bound_shader_p
    if(shader & shader->get_textures_used_bitmask() & DIGUY_TEXTURE_MAP_MASK_COMPONENTS)
    {
        if (glActiveTexture){
            glActiveTexture(GL_TEXTURE0 + DIGUY_TEXTURE_MAP_TYPE_COMPONENT_RAMP);
        }

        if(m_unique_texture_object)
        {
            glBindTexture(GL_TEXTURE_2D, m_unique_texture_object);
        }
        else{
            if(sm_default_variation_texture_object == 0){
                create_default_variation_texture();
            }
            glBindTexture(GL_TEXTURE_2D, sm_default_variation_texture_object);
        }
    }
}
}
```

```
void
diguyOglGraphicsShape::update_variation_texture(int num_pixels, unsigned char * pixel_data)
{
    /*
     * Override from the base class. Send a unique texture to the graphics card.
     */
    if(m_unique_texture_object == 0){
```

```

        glGenTextures(1, _object);
    }
    glBindTexture(GL_TEXTURE_2D, m_unique_texture_object);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, num_pixels, 1, 0, GL_RGBA, GL_UNSIGNED_BYTE, p);

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
    glGenerateMipmapEXT(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, 0);
}

```

```

void
diguyOglGraphicsShape::create_default_variation_texture()
{
    /*
     * If there's no variation texture and we're using a variation based shader
     * create one as a place holder.
     */
    if(sm_default_variation_texture_object == 0){
        glGenTextures(1, &m_default_variation_texture_object);
    }
    glBindTexture(GL_TEXTURE_2D, m_default_variation_texture_object);
    unsigned char buf[] = {128, 128, 128, 255};
    glTexImage2D(GL_TEXTURE_2D, 0, 4, 1, 1, 0, GL_RGBA, GL_UNSIGNED_BYTE, buf);

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
    glGenerateMipmapEXT(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, 0);
}

```

Create function definition

Create and return an object of a subclass of [diguyGraphicsShape](#). This function will be called whenever DI-Guy requires a shape object.

```

diguyGraphicsShape*
diguyOglGraphicsShape_create_func(void* internal_data)
{
    return new diguyOglGraphicsShape(internal_data);
}

```

5.58 DI-Guy Graphics API, diguyOglGraphicsTexture

diguyOglGraphicsTexture.cpp

DI-Guy SDK Version 12.5.2

This file was automatically generated from `diguyOglGraphicsTexture.cpp`. Do not edit this file directly; the changes will be lost.

Introduction

Textures modify the visible appearance of meshes. They work in concert with materials to determine the final appearance of visible geometry.

The class **diguyOglGraphicsTexture** is an example of a [diguyGraphicsTexture](#) subclass for an immediate mode renderer using the DI-Guy Loader. The example subclass shows how to use the accessors of the [diguyGraphicsTexture](#) class and which virtual functions should be overridden for an immediate mode renderer such as OpenGL.

Header files and forward declarations

```
#include "diguyOglGraphicsTexture.h"
#include <libbdiolog.h>
#include <stdlib.h>
#include <stdio.h>
#include <assert.h>
```

Constructor

Description:

Initialize data members of the class to a known state.

Most base class data will not be fully initialized until during the Build Stage, so most base class accessors should **not** be called here.

```
diguyOglGraphicsTexture::diguyOglGraphicsTexture(void* internal_data)
:
    diguyGraphicsTexture(internal_data),
    m_texture_object(0)
{
    /*
     * Nothing else to do after calling base class constructor and
     * initializing data members.
     */
}
```

build

Build Stage

Description:

Override of `diguyGraphicsTexture::build()`.

The `build()` function should create renderer objects that will set the proper texture state in `bind()` calls made during the Draw Stage.

This implementation creates an OpenGL texture object for setting texture properties of meshes.

```
void
diguyOglGraphicsTexture::build()
{
    const char* fname = get_filename();
    unsigned char* texture_map_data = get_texture_map_data();

    /*
     * If there is texture data, create a mip-mapped texture object.
     */
    if (texture_map_data)
    {
        int width = get_width();
        int height = get_height();
        int num_components = get_num_components();

        int pixel_format_as_int = get_gl_format();
        if (pixel_format_as_int == -1)
        {
            bdi_log_printf(BDI_LOG_WARN, "WARNING: Failed to obtain legal "
                "pixel format for texture '%s'.\n",
                get_name());
            goto MAKE_NULL_TEXTURE;
        }

        GLenum pixel_format = (GLenum) pixel_format_as_int;

        /*
         * Choose reasonable mag and min filters.
         */
    }
```

```

GLenum texture_mag_filter = GL_LINEAR;
GLenum texture_min_filter = GL_LINEAR_MIPMAP_LINEAR;

glGenTextures(1, );
glBindTexture(GL_TEXTURE_2D, m_texture_object);

glPixelStorei(GL_UNPACK_ALIGNMENT, 4);

if (get_clamp_s())
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
else
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);

if (get_clamp_t())
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
else
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, texture_mag_filter);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, texture_min_filter);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);

/*
 * Get the texture data into OpenGL. The calls we use depend
 * on whether the texture data is compressed or not.
 */
diguyGraphicsTextureCompressionType compression_type = get_texture_compression_type();

//printf("%s %s\n", get_filename(), get_gl_format_as_string());

if (compression_type == DIGUY_GRAPHICS_TEXTURE_COMPRESSION_TYPE_NONE)
{
    /*
     * Currently uncompressed textures are assumed to not have
     * included mipmaps. This doesn't take into account .dds
     * files that aren't compressed, which DI-Guy doesn't yet
     * support.
     *
     * If we've set the min filter to one of the "MIPMAP" options
     * (e.g., GL_LINEAR_MIPMAP_LINEAR) we should have OpenGL create
     * mipmaps. The filters GL_NEAREST and GL_LINEAR don't require
     * mipmaps.
     *
     * http://www.opengl.org/wiki/Common\_Mistakes says never use
     * gluBuild2DMipmaps.
     */
    //gluBuild2DMipmaps(GL_TEXTURE_2D,
    // num_components,
    // width,
    // height,
    // pixel_format,
    // GL_UNSIGNED_BYTE,
    // texture_map_data);

    int color_format = -1;
    switch (num_components)
    {
        case 1: color_format = GL_LUMINANCE; break;
        case 2: color_format = GL_LUMINANCE_ALPHA; break;
        case 3: color_format = GL_RGB; break;
        case 4: color_format = GL_RGBA; break;
    }
}

```

```

    }

    glTexImage2D(GL_TEXTURE_2D,
        0,
        pixel_format,
        width,
        height,
        0,
        color_format,
        GL_UNSIGNED_BYTE,
        texture_map_data);

    if ((texture_min_filter == GL_NEAREST_MIPMAP_NEAREST) ||
        (texture_min_filter == GL_NEAREST_MIPMAP_LINEAR) ||
        (texture_min_filter == GL_LINEAR_MIPMAP_NEAREST) ||
        (texture_min_filter == GL_LINEAR_MIPMAP_LINEAR))
    {
        glGenerateMipmap(GL_TEXTURE_2D);
    }
}
else
{
    /*
     * Compressed textures may already include mipmaps. If this
     * is the case, we need to manually tell OpenGL the mipmap
     * offsets.
     */
    int num_included_mipmaps = get_num_included_mipmaps();

    /*
     * There's always at least 1 included mipmap: the base image.
     */
    if (num_included_mipmaps == 0)
        num_included_mipmaps = 1;

    /*
     * Get mipmap offsets.
     *
     * "Mipmap 0" is really the base-level image. Each successive
     * mipmap should decrease in width and height by a factor of 2.
     */
    int mipmap_width = width;
    int mipmap_height = height;

    assert(glCompressedTexImage2D);

    int k;
    for (k=0; k<num_included_mipmaps; k++)
    {
        if (mipmap_width == 0) mipmap_width = 1;
        if (mipmap_height == 0) mipmap_height = 1;

        glCompressedTexImage2D(GL_TEXTURE_2D,
            k,
            pixel_format,
            mipmap_width,
            mipmap_height,
            0,
            get_mipmap_data_size(k),
            texture_map_data + get_mipmap_data_offset(k));
    }
}

```



```

        mipmap_width >= 1;
        mipmap_height >= 1;
    }

    /*
     * If there are missing mipmaps, the following keeps GL working
     * right. Otherwise the mipmap levels that are missing would
     * show up as white.
     */
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAX_LEVEL, num_included_mipmaps - 1);
}

/*
 * Else create a "null" texture object.
 */
else
{
MAKE_NULL_TEXTURE:

    glGenTextures(1, &m_texture_object);
    glBindTexture(GL_TEXTURE_2D, m_texture_object);
    unsigned char buf[] = {255,255,255,255};
    glTexImage2D(GL_TEXTURE_2D, 0, 4, 1, 1, 0, GL_RGBA, GL_UNSIGNED_BYTE, buf);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);

}

glBindTexture(GL_TEXTURE_2D, 0);

/*
 * Free memory allocated by the loader. Now that the texture data is
 * in an OpenGL texture object, the raw data loaded from the texture
 * file can be freed.
 */
free_loader_memory();
}

```

unbuild

Unbuild Stage

Description:

Override of `diguyOglGraphicsTexture::unbuild()`.

The `unbuild()` function should destroy the objects created in the `build()` function.

This implementation deletes the OpenGL texture object created in `build()`.

```

void
diguyOglGraphicsTexture::unbuild()
{
    if (m_texture_object != 0)
        glDeleteTextures(1, &m_texture_object);
}

```

bind_now*Draw Stage***Description:**

Override of virtual function of `diguyGraphicsTexture::bind_now()`, called during Draw Stage.

The `bind_now()` function should do whatever is necessary to change the current state of the renderer to use this texture.

The texture object created in `build()` is bound, changing OpenGL texture settings.

```
void
diguyOglGraphicsTexture::bind_now(diguyTextureMapType texture_type)
{
    if (diguyOglGraphicsTexture::get_last_bound_texture(texture_type) == this)
        return;

    if (m_texture_object == 0)
    {
        return;
    }
    glActiveTexture(GL_TEXTURE0 + texture_type);
    glBindTexture(GL_TEXTURE_2D, m_texture_object);
}
```

Create function definition

Create and return an object of a subclass of `diguyGraphicsTexture`. This function will be called whenever DI-Guy requires a texture object.

```
diguyGraphicsTexture*
diguyOglGraphicsTexture_create_func(void* internal_data)
{
    return new diguyOglGraphicsTexture(internal_data);
}
```

5.59 DI-Guy Graphics API, diguyOsgExtGraphicsFile**diguyOsgExtGraphicsFile.cpp****DI-Guy SDK Version 12.5.2**

This file was automatically generated from `diguyOsgExtGraphicsFile.cpp`. Do not edit this file directly; the changes will be lost.

Introduction

Files are the high-level representation of loaded geometry files. The geometry files contain parts of visible geometry that will be attached to shapes.

The class **diguyOsgExtGraphicsFile** is an example of a **diguyGraphicsFile** subclass for a **retained mode renderer** using an external loader. The example subclass shows how to use the accessors of the **diguyGraphicsFile** class and which virtual functions should be overridden for a retained mode renderer such as Open Scene Graph.

Header files and forward declarations

```
#include "diguyOsgExtGraphicsFile.h"

#include <string>

#include <stdlib.h>
#include <string.h>

#include <libbdiolog.h>

#include <osg/Geode>
#include <osg/Notify>
#include <osg/ShapeDrawable>
#include <osgDB/ReadFile>
#include <osgDB/Registry>

int diguyOsgExtGraphicsFile::s_osgdb_path_modified = 0;
```

Constructor

Description:

Initialize data members of the class to a known state.

Base class data will not be fully initialized until the Build Stage is done, so base class accessors should **not** be called here.

```
diguyOsgExtGraphicsFile::diguyOsgExtGraphicsFile(void* internal_data)
:
  diguyGraphicsFile(internal_data)
{
  /*
   * Add some DI-Guy directories to the list of directories OSG will
   * search for files.
   */
  if (!s_osgdb_path_modified)
  {
    char* diguy_env_var = getenv("DIGUY");
    char* bdigeometry_env_var = getenv("BDIGEOMETRY");
    char* diguy_geometry_env_var = getenv("DIGUY_GEOMETRY");
    char sprintf_temp[1024];

    /*
     * Get the current data file path.
```

```

    */
    osgDB::FilePathList list = osgDB::Registry::instance()->getDataFilePathList();

    /*
    *   Add to it.
    */
    sprintf(sprintf_temp, "%s", diguy_geometry_env_var);
    list.push_back(std::string(sprintf_temp));

    sprintf(sprintf_temp, "%s/flt", diguy_geometry_env_var);
    list.push_back(std::string(sprintf_temp));

    sprintf(sprintf_temp, "%s", bdigeometry_env_var);
    list.push_back(std::string(sprintf_temp));

    sprintf(sprintf_temp, "%s/flt", bdigeometry_env_var);
    list.push_back(std::string(sprintf_temp));

    sprintf(sprintf_temp, "%s/geometry", diguy_env_var);
    list.push_back(std::string(sprintf_temp));

    sprintf(sprintf_temp, "%s/geometry/flt", diguy_env_var);
    list.push_back(std::string(sprintf_temp));

    osgDB::Registry::instance()->setDataFilePathList(list);

    s_osgdb_path_modified = 1;
}

```

load

Build Stage

Description:

Override of `diguyGraphicsFile::load()`.

The `load()` function should load the specified file, and be ready to provide pointers to parts from the file.

This implementation uses an OSG plugin to load .flt files.

```

int
diguyOsgExtGraphicsFile::load(const char* filename, const char* extension)
{
    /*
    *   Don't try to load anything but a ".flt" file.
    */
    if (strcmp(extension, "flt"))
        return -1;

    /*
    *   Read the flt file.
    */
    osg::NotifySeverity original_notify_level = osg::getNotifyLevel();
    osg::setNotifyLevel(osg::WARN);
    {

```

```

        m_file_root = osgDB::readNodeFile(filename);
    }
    osg::setNotifyLevel(original_notify_level);

    if (!m_file_root)
    {
        bdi_log_printf(BDI_LOG_WARN,
            "WARNING: Failed to load geometry file '%s'.\n",
            filename);
        return -1;
    }

    bdi_log_printf(BDI_LOG_INFO,
        "INFO: Loaded geometry file '%s'.\n",
        filename);

    return 0;
}

```

unload

Unbuild Stage

Description:

Override of `diguyGraphicsFile::load()`.

The `unload()` function should undo the actions of the `load()` function. Whatever objects were created in `load()` should be deleted here.

This implementation dereferences the root node of the loaded file, potentially deleting it.

```

int
diguyOsgExtGraphicsFile::unload()
{
    m_file_root = NULL;
    return 0;
}

```

find_part

Build Stage

Description:

Override of `diguyGraphicsFile::find_part()`.

The `find_part()` function should find and return a geometry part from this file. This part will be attached to a shape.

This implementation looks for a sub-node with the given name in the hierarchy returned by the `load()` function.

```

void*
diguyOsgExtGraphicsFile::find_part(const char* part_name)
{
    if (!m_file_root)
        return NULL;

    osg::Node* part = NULL;

    if (strlen(part_name) == 0)
    {
        /*
         * Empty part name? Return entire hierarchy.
         */
        part = m_file_root;
        if (!part)
        {
            bdi_log_printf(BDI_LOG_DEBUG, "DEBUG: Root-node part not found in geometry fi
            get_filename());
        }
    }
    else
    {
        /*
         * Use the utility function recursive_find_part() to find a node
         * in the hierarchy with the given name.
         */
        part = recursive_find_part(part_name, m_file_root);
        if (!part)
        {
            bdi_log_printf(BDI_LOG_DEBUG, "DEBUG: Part '%s' not found in geometry file '%s
            part_name,
            get_filename());
        }
    }

    return (void *) part;
}

```

recursive_find_part

Description:

Protected utility function for finding a sub-node with the given name.

So far I haven't found an OSG function that recursively searches the hierarchy for a node with a particular name, so I've written my own version here.

```

osg::Node*
diguyOsgExtGraphicsFile::recursive_find_part(const char* part_name,
    osg::Node* parent)
{
    /*
     * See if parent itself is the desired part.
     */
    std::string parent_name = parent->getName();

    if (parent_name == part_name)
        return parent;
}

```

```

/*
 * Check to see if parent is a group. If not, part not found.
 */
osg::Group* parent_as_group = parent->asGroup();
if (!parent_as_group)
    return NULL;

/*
 * Recursively check group children.
 */
unsigned int child_index;
for (child_index=0;
     child_index<parent_as_group->getNumChildren();
     child_index++)
{
    osg::Node* child = parent_as_group->getChild(child_index);
    if (child)
    {
        osg::Node* part = recursive_find_part(part_name, child);
        if (part)
            return part;
    }
}

return NULL;
}

```

Create function definition

Create and return an object of a subclass of [diguyGraphicsFile](#). This function will be called whenever DI-Guy requires a file object.

```

diguyGraphicsFile*
diguyOsgExtGraphicsFile_create_func(void* internal_data)
{
    return new diguyOsgExtGraphicsFile(internal_data);
}

```

5.60 DI-Guy Graphics API, diguyOsgExtGraphicsLink

diguyOsgExtGraphicsLink.cpp

DI-Guy SDK Version 12.5.2

This file was automatically generated from diguyOsgExtGraphicsLink.cpp. Do not edit this file directly; the changes will be lost.

Introduction

DI-Guy 'links' are essentially the same as 'bones' in other APIs.

There are two primary purposes of links:

- ◆ to serve as attachment points for shapes, and
- ◆ to form a skeleton that can be animated using motion data.

The class **diguyOsgExtGraphicsLink** is an example of a **diguyGraphicsLink** subclass for a **scene graph renderer** using an external loader. The example subclass shows how to use the accessors of the **diguyGraphicsLink** class and which virtual functions should be overridden for a retained mode renderer such as Open Scene Graph.

Transformation Coordinate Systems

This example shows two ways that link transform matrix data can be retrieved from the **diguyGraphicsLink** base class: transforms in character-local space, and transforms in link-local space.

Character Local

For the character-local approach the top-level position link is given the world space transform; this becomes the origin of the character local coordinate system. All child link transforms are in that coordinate system.

This approach will be faster than the link-local approach, as DI-Guy has already computed the character-local matrices, and they can be retrieved and applied as-is to OSG Matrixf nodes.

Link Local

For the link-local approach all links are given a transform relative to their parent link's coordinate system.

This approach will be slower than the character-local approach, for in each update call the translate, rotate, and scale values must be retrieved, and a link-local matrix generated from these values.

(This approach was used in DI-Guy versions prior to DI-Guy 11.

Header files and forward declarations

```
#define USE_CHARACTER_LOCAL_TRANSFORMS 1

#include "diguyOsgExtGraphicsLink.h"

#include <libbdilog.h>
#include <diguyCharacter.h>
```



```

#include <stdlib.h>

#include <osg/Group>
#include <osg/MatrixTransform>
#include <osg/Matrixf>

#define CONSTANT_PI ((float) 3.1415926535897932384626433832795028841971693993751058209749)
#define CONSTANT_DEG2RAD (CONSTANT_PI / 180.0f)

/*
 * Pointer to an osg::Group to which DI-Guy character geometry should be
 * attached.
 */
osg::Group* diguyOsgExtGraphicsLink::s_diguy_attach_pt = NULL;

```

Constructor

Description:

Initialize data members of the class to a known state.

Base class data will not be fully initialized until the Build Stage is done, so base class accessors should **not** be called here.

```

diguyOsgExtGraphicsLink::diguyOsgExtGraphicsLink(void* internal_data)
:
    diguyGraphicsLink(internal_data),
    m_offset_node(NULL),
    m_dof_node(NULL)
{
    /*
     * Nothing else to do after calling base class constructor and
     * initializing data members.
     */
}

```

Character Local Approach

```
#if USE_CHARACTER_LOCAL_TRANSFORMS
```

build

Build Stage

Description:

Override of `diguyGraphicsLink::build()`.

The `build()` function should create the objects necessary to apply offsets contained in this link to its shapes and child links.

There are two types of offsets:

- ◆ Static offsets that specify link lengths from the parent link (e.g., distance from shoulder to elbow).
- ◆ Runtime transformations, usually joint rotations, read from DI-Guy motion data.

Since links don't have any actual geometry (shapes and meshes attached to them do), usually no actual drawing occurs in this function.

This implementation creates `osg::MatrixTransform` that will allow offsets from the parent link to be applied, and transformations from DI-Guy motion data to be applied.

```
void
diguyOsgExtGraphicsLink::build()
{
    /*
     * If we're using the character-local transformation approach, the
     * offset nodes aren't needed, and the DOF nodes are attached to the
     * top-level position node.
     */
    if (get_is_position_link())
    {
        /*
         * The top-level position link needs an offset node. This node
         * is attached to the scene via s_diguy_attach_pt. It does not
         * have any actual offset values, however; it just serves as a
         * top-level handle to the node hierarchy.
         */
        m_offset_node = new osg::MatrixTransform;

        /*
         * Build a DOF (degree of freedom) matrix transform for this
         * link that will be updated each frame from motion data.
         *
         * The position link DOF node applies a transform from world
         * space to character-local space.
         */
        m_dof_node = new osg::MatrixTransform;

        m_offset_node->addChild(m_dof_node);
    }
    else
    {
        /*
         * Build a DOF (degree of freedom) matrix transform for this
         * link that will be updated each frame from motion data.
         *
         * Non-top-level links are attached to the position link's DOF
         * node. They will contain character-local transforms.
         */
        m_dof_node = new osg::MatrixTransform;

        /*
         * Attach the DOF node to the position link's DOF node.
         */
        diguyCharacter* character = get_character();
        if (character)
        {
            diguyGraphicsLink* position_link = character->get_position_link();
            diguyOsgExtGraphicsLink* osg_position_link =
                (diguyOsgExtGraphicsLink *) position_link;
        }
    }
}
```

```

        osg_position_link->m_dof_node->addChild(m_dof_node);
    }
}

```

unbuild

Unbuild Stage

Description:

Override of `diguyGraphicsLink::unbuild()`.

The `unbuild()` function should destroy the objects created in the `build()` function.

This implementation dereference the nodes created in `build()`.

```

void
diguyOsgExtGraphicsLink::unbuild()
{
    if (get_is_position_link())
    {
        m_offset_node->removeChild(m_dof_node);
        m_offset_node = NULL;

        m_dof_node = NULL;
    }
    else
    {
        diguyCharacter* character = get_character();
        if (character)
        {
            diguyGraphicsLink* position_link = character->get_position_link();
            diguyOsgExtGraphicsLink* osg_position_link =
                (diguyOsgExtGraphicsLink *) position_link;

            osg_position_link->m_dof_node->removeChild(m_dof_node);

            m_dof_node = NULL;
        }
    }
}

```

update

Update Stage

Description:

Override of `diguyGraphicsLink::update()`.

The `update()` function should update objects created in `build()` with up-to-date values from motion data.

This implementation reads transformation data from the base class and loads it into the `osg::MatrixTransform` nodes created in `build()`.

```
void
diguyOsgExtGraphicsLink::update()
{
    /*
     * Update the scene graph nodes created during build() with new
     * values. The position link matrix is in world space, all others
     * in character-local space.
     */
    osg::Matrixf mat;
    float link_matrix[4][4];

    if (get_is_position_link())
        get_transformation_matrix(link_matrix);
    else
        get_local_transformation_matrix(link_matrix);

    /*
     * DI-Guy transformation matrices are transposed from what OSG
     * uses.
     */
    mat.set(
        link_matrix[0][0], link_matrix[1][0], link_matrix[2][0], link_matrix[3][0],
        link_matrix[0][1], link_matrix[1][1], link_matrix[2][1], link_matrix[3][1],
        link_matrix[0][2], link_matrix[1][2], link_matrix[2][2], link_matrix[3][2],
        link_matrix[0][3], link_matrix[1][3], link_matrix[2][3], link_matrix[3][3]);

    /*
     * Set the m_dof_node matrix.
     */
    m_dof_node->setMatrix(mat);
}
```

Link Local Approach

```
#else
```

build

Build Stage

Description:

Override of `diguyGraphicsLink::build()`.

Look in the Character Local Approach `build()` comment above for what this function should do.

```
void
diguyOsgExtGraphicsLink::build()
{
    /*
     * Create a matrix that has the link's static offset
     * transformations. This matrix will not be changed during
```

```

    *   update() calls.
    */
    osg::Matrixf mat;
    osg::Matrixf mat_temp;

    mat.makeIdentity();

    /*
     *   Add explicit offset translation.
     */
    float offset_t[3];
    get_offset_translation(

    mat_temp.makeTranslate(offset_t[0],
        offset_t[1],
        offset_t[2]);
    mat.preMult(mat_temp);

    /*
     *   Add explicit offset rotation.
     */
    float offset_r[3];
    get_offset_rotation(

    if ((offset_r[0] != 0.0f) ||
        (offset_r[1] != 0.0f) ||
        (offset_r[2] != 0.0f))
    {
        mat_temp.makeRotate(offset_r[2] * CONSTANT_DEG2RAD,
            osg::Vec3f(0.0f, 1.0f, 0.0f),
            offset_r[1] * CONSTANT_DEG2RAD,
            osg::Vec3f(1.0f, 0.0f, 0.0f),
            offset_r[0] * CONSTANT_DEG2RAD,
            osg::Vec3f(0.0f, 0.0f, 1.0f));
        mat.preMult(mat_temp);
    }

    /*
     *   The runtime translation values of each link may be marked as
     *   always static.  If this is the case for this link, add the
     *   runtime translation into the offset matrix.
     */
    if (get_translation_is_static())
    {
        float t[3];
        get_translation(

        mat_temp.makeTranslate(t[0],
            t[1],
            t[2]);
        mat.preMult(mat_temp);
    }

    /*
     *   Runtime rotation values may also be marked as static.
     */
    if (get_rotation_is_static())
    {
        float r[3];
        get_rotation(

```

```

        if ((r[0] != 0.0f) ||
            (r[1] != 0.0f) ||
            (r[2] != 0.0f))
        {
            mat_temp.makeRotate(r[2] * CONSTANT_DEG2RAD,
                               osg::Vec3f(0.0f, 1.0f, 0.0f),
                               r[1] * CONSTANT_DEG2RAD,
                               osg::Vec3f(1.0f, 0.0f, 0.0f),
                               r[0] * CONSTANT_DEG2RAD,
                               osg::Vec3f(0.0f, 0.0f, 1.0f));
            mat.preMult(mat_temp);
        }
    }

    /*
     * Runtime scale values may also be marked as static.
     */
    if (get_scale_is_static())
    {
        float s[3];
        get_scale(

            if ((s[0] != 1.0f) ||
                (s[1] != 1.0f) ||
                (s[2] != 1.0f))
            {
                mat_temp.makeScale(osg::Vec3f(s[0], s[1], s[2]));
                mat.preMult(mat_temp);
            }
        )

    }

    /*
     * Build an offset node for this link based on the above computed
     * offset matrix.
     */
    m_offset_node = new osg::MatrixTransform;
    m_offset_node->setMatrix(mat);

    /*
     * Build a DOF (degree of freedom) matrix transform for this link
     * that will be updated each frame from motion data.
     *
     * This DOF node will be updated with link-local transform data.
     */
    m_dof_node = new osg::MatrixTransform;

    m_offset_node->addChild(m_dof_node);

    /*
     * See if we should attach to parent link. The top-level position
     * link should not start out attached. When necessary it will be
     * attached to the scene graph in the attach_to_scene() function.
     */
    if (!get_is_position_link())
    {
        diguyGraphicsLink* parent_link = get_parent_link();
        diguyOsgExtGraphicsLink* osg_parent_link =
            (diguyOsgExtGraphicsLink *) parent_link;
        osg_parent_link->m_dof_node->addChild(m_offset_node);
    }
}

```

unbuild*Unbuild Stage***Description:**

Override of `diguyGraphicsLink::unbuild()`.

Look in the Character Local Approach `unbuild()` comment above for what this function should do.

```
void
diguyOsgExtGraphicsLink::unbuild()
{
    m_offset_node->removeChild(m_dof_node);

    if (!get_is_position_link())
    {
        diguyGraphicsLink* parent_link = get_parent_link();
        diguyOsgExtGraphicsLink* osg_parent_link =
            (diguyOsgExtGraphicsLink *) parent_link;

        osg_parent_link->m_dof_node->removeChild(m_offset_node);
    }

    /*
     * Setting the m_offset_node and m_dof_node ref_ptrs to NULL
     * dereferences the objects they point to, possibly deleting them.
     */
    m_offset_node = NULL;
    m_dof_node = NULL;
}
```

update*Update Stage***Description:**

Override of `diguyGraphicsLink::update()`.

Look in the Character Local Approach `update()` comment above for what this function should do.

```
void
diguyOsgExtGraphicsLink::update()
{
    /*
     * Update the scene graph nodes created during build() with new
     * values.
     *
     * These values should be obtained by calling get_translation(),
     * get_rotation(), and get_scale().
     */
    osg::Matrixf mat;
    osg::Matrixf mat_temp;
```

```

/*
 * Start with identity.
 */
mat.makeIdentity();

if (!get_translation_is_static())
{
    float t[3];
    get_translation(

        mat_temp.makeTranslate(t[0],
                                t[1],
                                t[2]);
    mat.preMult(mat_temp);
}

if (!get_rotation_is_static())
{
    float r[3];
    get_rotation(

        if ((r[0] != 0.0f) ||
            (r[1] != 0.0f) ||
            (r[2] != 0.0f))
        {
            mat_temp.makeRotate(r[2] * CONSTANT_DEG2RAD,
                                osg::Vec3f(0.0f, 1.0f, 0.0f),
                                r[1] * CONSTANT_DEG2RAD,
                                osg::Vec3f(1.0f, 0.0f, 0.0f),
                                r[0] * CONSTANT_DEG2RAD,
                                osg::Vec3f(0.0f, 0.0f, 1.0f));
            mat.preMult(mat_temp);
        }
}

if (!get_scale_is_static())
{
    float s[3];
    get_scale(

        if ((s[0] != 1.0f) ||
            (s[1] != 1.0f) ||
            (s[2] != 1.0f))
        {
            mat_temp.makeScale(osg::Vec3f(s[0], s[1], s[2]));
            mat.preMult(mat_temp);
        }
}

/*
 * Set the m_dof_node matrix.
 */
m_dof_node->setMatrix(mat);
}

```

Common to Character Local and Link Local Approach

#endif

attach_to_scene*Update Stage***Description:**

Override of `diguyGraphicsLink::attach_to_scene()`.

The `attach_to_scene()` function should perform the actions necessary to attach this link to somewhere on the scene graph.

This implementation attaches this link's offset node as a child to the node specified by `s_diguy_attach_pt`.

This function will only be called on the position link.

```
void
diguyOsgExtGraphicsLink::attach_to_scene()
{
    /*
     * Need m_offset_node to do attach.
     */
    if (!m_offset_node)
        return;

    if (m_offset_node->getNumParents() > 0)
    {
        bdi_log_printf(BDI_LOG_WARN,
            "WARNING: Link '%s' already attached to something.  "
            "Call to attach_to_scene() failed.\n",
            get_name());
        return;
    }

    /*
     * Make the attachment.
     */
    if (s_diguy_attach_pt)
    {
        s_diguy_attach_pt->addChild(m_offset_node);
    }
}
```

detach_from_scene*Update Stage***Description:**

Override of `diguyGraphicsLink::detach_from_scene()`.

The `detach_to_scene()` function should perform the actions necessary to detach this link from the scene graph.

This implementation removes this link's offset node as a child from the node specified by `s_diguy_attach_pt`.

This function will only be called on the position link.

```
void
diguyOsgExtGraphicsLink::detach_from_scene()
{
    /*
     * Need m_offset_node to do detach.
     */
    if (!m_offset_node)
        return;

    /*
     * If m_offset_node has no parents, it is already detached.
     */
    if (m_offset_node->getNumParents() == 0)
    {
        bdi_log_printf(BDI_LOG_WARN,
            "WARNING: Link '%s' not attached to anything.  "
            "Call to detach_from_scene() failed.\n",
            get_name());
        return;
    }

    if (s_diguy_attach_pt)
    {
        s_diguy_attach_pt->removeChild(m_offset_node);
    }
}
```

attach_to_link

Update Stage

Description:

Override of `diguyGraphicsLink::attach_to_link()`.

This function is similar to `attach_to_scene()`, but the link's object should be attached to another link instead of the scene graph.

```
void
diguyOsgExtGraphicsLink::attach_to_link(diguyGraphicsLink* link)
{
    /*
     * Need m_offset_node to do attach.
     */
    if (!m_offset_node)
        return;

    /*
     * If m_offset_node already has parents, it is already attached.
     */
    if (m_offset_node->getNumParents() > 0)
    {

```

```

        bdi_log_printf(BDI_LOG_WARN,
            "WARNING: Link '%s' already attached to something.  "
            "Call to attach_to_link() failed.\n",
            get_name());
        return;
    }

    /*
     * Attach m_offset_node of this link to m_dof_node of the
     * attachment link.
     */
    diguyOsgExtGraphicsLink* osg_link = (diguyOsgExtGraphicsLink*) link;
    if (osg_link->m_dof_node)
    {
        osg_link->m_dof_node->addChild(m_offset_node);
    }
}

```

detach_from_link

Update Stage

Description:

Override of `diguyGraphicsLink::detach_from_link()`.

This function is similar to `detach_from_scene()`, but the link's object should be detached from another link instead of the scene graph.

```

void
diguyOsgExtGraphicsLink::detach_from_link()
{
    /*
     * Need m_offset_node to do detach.
     */
    if (!m_offset_node)
        return;

    /*
     * If m_offset_node has no parents, it is already detached.
     */
    if (m_offset_node->getNumParents() == 0)
    {
        bdi_log_printf(BDI_LOG_WARN,
            "WARNING: Link '%s' not attached to anything.  "
            "Call to detach_from_link() failed.\n",
            get_name());
        return;
    }

    diguyGraphicsLink* link = get_attached_to_link();

    diguyOsgExtGraphicsLink* osg_link = (diguyOsgExtGraphicsLink*) link;
    if (osg_link->m_dof_node)
    {
        osg_link->m_dof_node->removeChild(m_offset_node);
    }
}

```

Create function definition

Create and return an object of a subclass of [diguyGraphicsLink](#). This function will be called whenever DI-Guy requires a link object.

```
diguyGraphicsLink*
diguyOsgExtGraphicsLink_create_func(void* internal_data)
{
    return new diguyOsgExtGraphicsLink(internal_data);
}
```

5.61 DI-Guy Graphics API, diguyOsgExtGraphicsShape**diguyOsgExtGraphicsShape.cpp****DI-Guy SDK Version 12.5.2**

This file was automatically generated from `diguyOsgExtGraphicsShape.cpp`. Do not edit this file directly; the changes will be lost.

Introduction

Shapes are the high-level containers of visible geometry. They are attached to links, and contain zero or more mesh objects.

The class **diguyOsgExtGraphicsShape** is an example of a [diguyGraphicsShape](#) subclass for a **retained renderer** using an external loader. The example subclass shows how to use the accessors of the [diguyGraphicsShape](#) class and which virtual functions should be overridden for an immediate mode renderer such as Open Scene Graph.

Header files and forward declarations

```
#include "diguyOsgExtGraphicsShape.h"
#include "diguyOsgExtGraphicsLink.h"

#include <libbdiolog.h>

#include <diguyScenario.h>
#include <diguyCharacter.h>

/*
 * Top-level static variable that determines whether LOD calculations are
 * based strictly on distance of object from camera, or on size of object,
 * in pixels, in frame.
```

```

*/
bool diguyOsgExtGraphicsShape::s_use_pixel_lods = false;

```

Constructor

Description:

Initialize data members of the class to a known state.

Base class data will not be fully initialized until the Build Stage is done, so base class accessors should **not** be called here.

```

diguyOsgExtGraphicsShape::diguyOsgExtGraphicsShape(void* internal_data)
:
    diguyGraphicsShape(internal_data),
    m_switch_node(NULL),
    m_lod_node(NULL),
    m_manual_lod_switch_node(NULL)
{
    /*
     * Nothing else to do after calling base class constructor and
     * initializing data members.
     */
}

```

build

Build Stage

Description:

Override of `diguyGraphicsShape::build()`.

The `build()` function should look up the shape's parts from previously loaded geometry files, and create the necessary objects that will apply offset data of this shape from its parent link during the Draw Stage.

This implementation creates the nodes necessary to switch the shape geometry on and off, apply lod calculations, and look up and attach nodes found in a DI-Guy geometry file.

```

void
diguyOsgExtGraphicsShape::build()
{
    int lod;

    diguyGraphicsLink* diguy_link = get_link();
    diguyOsgExtGraphicsLink* osg_link = (diguyOsgExtGraphicsLink *) diguy_link;

    osg::MatrixTransform* link_dof_node = osg_link->get_dof_node();

    if (!link_dof_node)
        return;

    /*
     * By default we'll attach meshes to the link's dof node.  If this

```

```

    *   shape is switched, however (e.g. for switchable hands), we'll create
    *   a switch node and attach them to the switch.
    */
osg::Group* lod_attach_node = link_dof_node;

/*
 *   See if we should create a switch.
 */
if (get_switch_var_name() != NULL)
{
    m_switch_node = new osg::Switch();
    m_switch_node->setNewChildDefaultValue(true);
    link_dof_node->addChild(m_switch_node);

    lod_attach_node = m_switch_node;
}

/*
 *   Determine whether automatic graphics LOD switching is enabled.
 *   This will determine which kind of node we create for LOD switching.
 */
diguyCharacter* character = get_character();
diguyScenario* scenario = character ? character->get_scenario() : NULL;
int automatic_graphics_lod_switching = scenario ? scenario->get_automatic_graphics_lo

/*
 *   If automatic switching is enabled, create an osg::LOD() node.
 */
if (automatic_graphics_lod_switching)
{
    /*
     *   Determine the lod_ranges.
     */
    float* lod_ranges = get_lod_ranges();

    /*
     *   Create an lod node.
     */
    m_lod_node = new osg::LOD();
    m_lod_node->setCenter(osg::Vec3f(0.0f, 0.0f, 0.0f));

    if (s_use_pixel_lods)
        m_lod_node->setRangeMode(osg::LOD::PIXEL_SIZE_ON_SCREEN);
    else
        m_lod_node->setRangeMode(osg::LOD::DISTANCE_FROM_EYE_POINT);

    lod_attach_node->addChild(m_lod_node);

    /*
     *   Attach shape parts.
     */
    for (lod=1; lod<=get_num_lods(); lod++)
    {
        void* diguy_part = get_diguy_graphics_file_part(lod);
        if (diguy_part)
        {
            osg::Node* osg_part = (osg::Node*) diguy_part;

            if (s_use_pixel_lods)
            {
                /*

```

```

        * Because DI-Guy LOD ranges must increase at each index,
        * but unlike with distances the earlier numbers represent
        * low LODs, we have to reverse the order of lod ranges.
        *
        * LOD 1: lod_ranges[6] to lod_ranges[7]
        * LOD 2: lod_ranges[5] to lod_ranges[6]
        * etc.
        */
        m_lod_node->addChild(osg_part, lod_ranges[7 - lod], lod_ranges[8 - lod]);
    }
    else
    {
        /*
        * LOD 1: lod_ranges[0] to lod_ranges[1]
        * LOD 2: lod_ranges[1] to lod_ranges[2]
        * etc.
        */
        m_lod_node->addChild(osg_part, lod_ranges[lod-1], lod_ranges[lod]);
    }
}

/*
 * Else automatic LOD switching is not enabled. Create a switch node
 * for manual LOD switching.
 */
else
{
    m_manual_lod_switch_node = new osg::Switch();
    m_manual_lod_switch_node->setNewChildDefaultValue(false);

    lod_attach_node->addChild(m_manual_lod_switch_node);

    /*
    * Create a geode for each lod.
    * Attach mesh drawables to geodes.
    */
    for (lod=1; lod<=get_num_lods(); lod++)
    {
        void* diguy_part = get_diguy_graphics_file_part(lod);

        if (diguy_part)
        {
            osg::Node* osg_part = (osg::Node*) diguy_part;
            m_manual_lod_switch_node->addChild(osg_part);
            m_manual_lod_switch_node->setSingleChildOn(get_graphics_lod() - 1);
        }
    }
}

/// /*
/// * Create a switch node.
/// */
/// m_switch_node = new osg::Switch ();
/// m_switch_node->setNewChildDefaultValue(true);
/// link_dof_node->addChild(m_switch_node);
///
/// /*
/// * Determine the lod_ranges.
/// */

```

```

///     float* lod_ranges = get_lod_ranges      ();
///
///     /*
///      *   Create an lod node.
///      */
///     m_lod_node = new osg::LOD      ();
///     m_lod_node->setCenter(osg::Vec3f(0.0f, 0.0f, 0.0f));
///
///     m_switch_node->addChild(m_lod_node);
///
///     /*
///      *   Attach shape parts.
///      */
///     for (lod=1; lod<=get_num_lods      (); lod++)
///     {
///         void* diguy_part = get_diguy_graphics_file_part(lod);
///         if (diguy_part)
///         {
///             osg::Node* osg_part = (osg::Node*) diguy_part;
///             m_lod_node->addChild(osg_part, lod_ranges[lod-1], lod_ranges[lod]);
///         }
///     }
/// }

```

unbuild

Unbuild Stage

Description:

Override of `diguyGraphicsShape::unbuild()`.

The `unbuild()` function should destroy the objects created in the `build()` function.

This implementation detaches and dereferences the nodes created and attached in the `build()` function.

```

void
diguyOsgExtGraphicsShape::unbuild()
{
    diguyGraphicsLink* diguy_link = get_link();
    diguyOsgExtGraphicsLink* osg_link = (diguyOsgExtGraphicsLink *) diguy_link;

    osg::MatrixTransform* link_dof_node = osg_link->get_dof_node();

    if (!link_dof_node)
        return;

    osg::Group* lod_detach_node = link_dof_node;

    /*
     * Remove all of the group nodes we created and added in the build()
     * call; this should reduce the ref count of the added nodes by 1, to
     * 1. Then set the pointers to NULL; this should reduce the ref count
     * to 0, resulting in the group nodes being deleted.
     *
     * Setting the m_switch_node ref_ptr to NULL dereferences the Switch
     * object it points to, possibly deleting it. Ditto for m_lod_node and

```



```

    *   m_manual_lod_switch_node.
    */
    if (m_switch_node != NULL)
        lod_detach_node = m_switch_node;

    if (m_lod_node != NULL)
    {
        lod_detach_node->removeChild(m_lod_node);
        m_lod_node = NULL;
    }
    else if (m_manual_lod_switch_node != NULL)
    {
        lod_detach_node->removeChild(m_manual_lod_switch_node);
        m_manual_lod_switch_node = NULL;
    }

    if (m_switch_node != NULL)
    {
        link_dof_node->removeChild(m_switch_node);
        m_switch_node = NULL;
    }

    ///   diguyGraphicsLink* diguy_link = get_link      ();
    ///   diguyOsgExtGraphicsLink* osg_link = (diguyOsgExtGraphicsLink *) diguy_link;
    ///
    ///   osg::MatrixTransform* link_dof_node = osg_link->get_dof_node      ();
    ///
    ///   if (!link_dof_node)
    ///       return;
    ///
    ///   /*
    ///    *   Detach shape parts.
    ///    */
    ///   int lod;
    ///   for (lod=1; lod<=get_num_lods      (); lod++)
    ///   {
    ///       void* diguy_part = get_diguy_graphics_file_part(lod);
    ///       if (diguy_part)
    ///       {
    ///           osg::Node* osg_part = (osg::Node*) diguy_part;
    ///           m_lod_node->removeChild(osg_part);
    ///       }
    ///   }
    ///
    ///   /*
    ///    *   Dereference lod node.
    ///    */
    ///   m_switch_node->removeChild(m_lod_node);
    ///   m_lod_node = NULL;
    ///
    ///   /*
    ///    *   Dereference switch.
    ///    */
    ///   link_dof_node->removeChild(m_switch_node);
    ///   m_switch_node = NULL;
    }

```

[show](#)
Update Stage

Description:

Override of `diguyGraphicsShape::show()`.

The `show()` function should take the necessary steps to make the shape's visible geometry visible.

This implementation turns all children of the shape's switch node on.

```
void
diguyOsgExtGraphicsShape::show()
{
    if (m_switch_node)
        m_switch_node->setAllChildrenOn();
}
```

hide

Update Stage

Description:

Override of `diguyGraphicsShape::hide()`.

The `hide()` function should take the necessary steps to make the shape's visible geometry hidden.

This implementation turns all children of the shape's switch node off.

```
void
diguyOsgExtGraphicsShape::hide()
{
    if (m_switch_node)
        m_switch_node->setAllChildrenOff();
}
```

set_graphics_lod_ranges**Description:**

Override of `diguyGraphicsShape::set_graphics_lod_ranges()`, called as needed.

The `set_graphics_lod_ranges()` function should update the ranges of lod objects created in the `build()` function with the passed ranges.

In this implementation the ranges of the `osg::LOD` created in `build()` are updated. If automatic LOD switching is enabled, the LOD node will not exist and this function will have no effect.

```
void
```

```

diguyOsgExtGraphicsShape::set_graphics_lod_ranges(float* lod_ranges)
{
    if (m_lod_node != NULL)
    {
        int lod;
        for (lod=1; lod<=get_num_lods(); lod++)
        {
            m_lod_node->setRange(lod-1, lod_ranges[lod-1], lod_ranges[lod]);
        }
    }
}

```

set_graphics_lod

Description:

Override of `diguyGraphicsShape::set_graphics_lod()`, called as needed.

The `set_graphics_lod()` function should select and show the object representing the passed LOD.

This implementation manually sets the `osg::Switch` created in `build()` to show the correct geode. If automatic LOD switching is enabled, the switch node will not exist and this function will have no effect.

```

void
diguyOsgExtGraphicsShape::set_graphics_lod(int lod)
{
    if (m_manual_lod_switch_node != NULL)
    {
        m_manual_lod_switch_node->setSingleChildOn(lod-1);
    }
}

```

Create function definition

Create and return an object of a subclass of `diguyGraphicsShape`. This function will be called whenever DI-Guy requires a shape object.

```

diguyGraphicsShape*
diguyOsgExtGraphicsShape_create_func(void* internal_data)
{
    return new diguyOsgExtGraphicsShape(internal_data);
}

```

5.62 DI-Guy Graphics API, diguyOsgGraphicsLink

diguyOsgGraphicsLink.cpp

DI-Guy SDK Version 12.5.2

This file was automatically generated from `diguyOsgGraphicsLink.cpp`. Do not edit this file directly; the

changes will be lost.

Introduction

DI-Guy 'links' are essentially the same as 'bones' in other APIs.

There are two primary purposes of links:

- ◆ to form a skeleton that can be animated using motion data, and
- ◆ to serve as attachment points for shapes.

The class **diguyOsgGraphicsLink** is an example of a [diguyGraphicsLink](#) subclass for a scene graph environment using the DI-Guy Loader. The example subclass shows how to use the accessors of the [diguyGraphicsLink](#) class and which virtual functions should be overridden for a scene graph environment such as OSG.

Transformation Coordinate Systems

This example shows two ways that link transform matrix data can be retrieved from the [diguyGraphicsLink](#) base class: transforms in character-local space, and transforms in link-local space.

Character Local

For the character-local approach the top-level position link is given the world space transform; this becomes the origin of the character local coordinate system. All child link transforms are in that coordinate system.

This approach will be faster than the link-local approach, as DI-Guy has already computed the character-local matrices, and they can be retrieved and applied as-is to OSG Matrixf nodes.

Link Local

For the link-local approach all links are given a transform relative to their parent link's coordinate system.

This approach will be slower than the character-local approach, for in each update call the translate, rotate, and scale values must be retrieved, and a link-local matrix generated from these values.

(This approach was used in DI-Guy versions prior to DI-Guy 11.)

Header files and forward declarations

```
#define USE_CHARACTER_LOCAL_TRANSFORMS 1

#include "diguyGraphicsMaterial.h"
#include "diguyGraphicsMesh.h"
#include "diguyGraphicsTexture.h"

#include "diguyOsgGraphicsLink.h"
#include "diguyOsgGraphicsShaderProgram.h"
#include "diguyOsgGraphicsShaderInstance.h"
#include "diguyOsgGraphicsShape.h"
#include "diguyOsgGraphicsState.h"
#include "diguyOsgUtil.h"

#include <diguyCharacter.h>
#include <diguyScenario.h>
#include <diguyGraphicsShaderProgram.h>
#include <libbdilog.h>

#include <stdlib.h>

#include <osg/Group>
#include <osg/MatrixTransform>
#include <osg/Matrixf>

#include <osg/Material>
#include <osg/PolygonMode>
#include <osg/Shape>
#include <osg/ShapeDrawable>
#include <osg/StateSet>

#include <osg/TexEnv>
#include <osg/Texture2D>

/*
 * Convenience constants.
 */
```

```

#define CONSTANT_PI ((float) 3.1415926535897932384626433832795028841971693993751058209749)
#define CONSTANT_DEG2RAD (CONSTANT_PI / 180.0f)

osg::ref_ptr<osg::Texture2D> diguyOsgGraphicsLink::sm_degenerate_texture = NULL;

/*
 * Top-level static variable that stores the osg group to which DI-Guy
 * objects should be added.
 */
osg::Group* diguyOsgGraphicsLink::s_diguy_attach_pt = NULL;

/*
 * Needed for looking up the current light. (There might be a cleaner way.)
 */
osgViewer::Viewer* diguyOsgGraphicsLink::s_viewer = NULL;

```

Constructor

Description:

Initialize data members of the class to a known state.

Base class data will not be fully initialized until during the Build Stage, so most base class accessors should **not** be called here.

```

diguyOsgGraphicsLink::diguyOsgGraphicsLink(void* internal_data)
:
    diguyGraphicsLink(internal_data),
    m_offset_node(NULL),
    m_dof_node(NULL),
    m_bounding_sphere_visual(NULL),
    m_diguy_osg_shader_instance(NULL),
    m_diguy_osg_shader_program(NULL),
    m_ufrm_link_matrix(NULL),
    m_ufrm_inverse_transform_matrix(NULL),
    m_ufrm_eye_point_os(NULL),
    m_ufrm_light_dir_os(NULL)
{
    /*
     * Nothing else to do after calling base class constructor and
     * initializing data members.
     */
}

```

Character Local Approach

```
#if USE_CHARACTER_LOCAL_TRANSFORMS
```

build

Build Stage

Description:

Override of `diguyGraphicsLink::build()`.

The `build()` function should create the objects necessary to apply offsets contained in this link to its shapes and child links.

There are two types of offsets:

- ◆ Static offsets that specify link lengths from the parent link (e.g., distance from shoulder to elbow).
- ◆ Runtime transformations, usually joint rotations, read from DI-Guy motion data.

This implementation creates `osg::MatrixTransform` that will allow offsets from the parent link to be applied, and transformations from DI-Guy motion data to be applied.

```
void
diguyOsgGraphicsLink::build()
{
    // Potential optimization
    //     if (get_num_shapes      () == 0 &!get_is_position_link())
    //         return;

    /*
     * If we're using the character-local transformation approach, the
     * offset nodes aren't needed, and the DOF nodes are attached to the
     * top-level position node.
     */
    if (get_is_position_link())
    {
        /*
         * The top-level position link needs an offset node. This node
         * is attached to the scene via s_diguy_attach_pt. It does not
         * have any actual offset values, however; it just serves as a
         * top-level handle to the node hierarchy.
         */
        m_offset_node = new osg::MatrixTransform;

        /*
         * If a scene graph mask has been specified for the character of
         * this link, set the node mask to it.
         */
        diguyCharacter* character = get_character();
        if (character &character->get_scene_graph_mask() != 0)
            m_offset_node->setNodeMask(character->get_scene_graph_mask());

        /*
         * Build a DOF (degree of freedom) matrix transform for this
         * link that will be updated each frame from motion data.
         *
         * The position link DOF node applies a transform from world
         * space to character-local space.
         */
        m_dof_node = new osg::MatrixTransform;

        m_offset_node->addChild(m_dof_node);
    }
    else
    {
        /*
         * Build a DOF (degree of freedom) matrix transform for this
         * link that will be updated each frame from motion data.
         */
    }
}
```

```

    * Non-top-level links are attached to the position link's DOF
    * node. They will contain character-local transforms.
    */

    // TODO for diguy 12 we could change this to a m_group_node
    // and avoid using a matrix transform since the dof data is going
    // into the uniform link matrix
    m_dof_node = new osg::MatrixTransform;

    /*
    * Attach the DOF node to the position link's DOF node.
    */
    diguyCharacter* character = get_character();
    if (character)
    {
        diguyGraphicsLink* position_link = character->get_position_link();
        diguyOsgGraphicsLink* osg_position_link = (diguyOsgGraphicsLink *) position_l

        osg_position_link->m_dof_node->addChild(m_dof_node);
    }
}

```

unbuild

Unbuild Stage

Description:

Override of `diguyGraphicsLink::unbuild()`.

The `unbuild()` function should destroy the objects created in the `build()` function.

This implementation dereference the nodes created in `build()`.

```

void
diguyOsgGraphicsLink::unbuild()
{
    if (!m_dof_node)
        return;

    if (get_is_position_link())
    {
        diguyOsgUtil::detach_and_null(m_dof_node);
        diguyOsgUtil::detach_and_null(m_offset_node);
    }
    else
    {
        diguyOsgUtil::detach_and_null(m_dof_node);
        diguyOsgUtil::detach_and_null(m_offset_node);
    }
}

```

update

Update Stage

Description:

Override of `diguyGraphicsLink::update()`.

The `update()` function should update objects created in `build()` with up-to-date values from motion data.

This implementation reads transformation data from the base class and loads it into the `osg::MatrixTransform` nodes created in `build()`.

```
void
diguyOsgGraphicsLink::update()
{
    if (!m_dof_node & !m_ufrm_link_matrix)
        return;

    /*
     * Update the scene graph nodes created during build() with new values.
     * The position link matrix is in world space, all others in
     * character-local space.
     */
    osg::Matrixf mat;
    int transpose = 1;

    if (!get_is_position_link())
    {
        /*
         * Set the link's matrix uniform.
         */
        get_local_transformation_matrix_4x4_ptr(mat.ptr(), transpose);
        m_ufrm_link_matrix->set(mat);

        return;
    }
    else
    {
        get_transformation_matrix_4x4_ptr(mat.ptr(), transpose);

        /*
         * We need to set the link's matrix uniform, but since the offset
         * from the origin will be applied as part of setting the
         * m_dof_node's matrix below, just set the uniform to contain the
         * identity matrix. Otherwise we'd end up with double offset.
         */
        if (m_ufrm_link_matrix)
        {
            osg::Matrixf identity;
            identity.makeIdentity();
            m_ufrm_link_matrix->set(identity);
        }

        /*
         * Apply scale for character.
         */
        osg::Vec3f scale;
        diguyCharacter* character = get_character();
        if (character)
            character->get_scale(
```

```

if (scale[0] != 1.0f || scale[1] != 1.0f || scale[2] != 1.0f)
{
    mat.preMultScale(scale);
}

/*
 * Set the m_dof_node matrix.
 */
m_dof_node->setMatrix(mat);

if (s_viewer)
{
    /*
     * This code is needed to get the bump map shader working. It
     * will need to be adapted to your code and it relies on
     * s_viewer and the diguy Camera position to be set. The two
     * uniforms below represent the light and the camera in the
     * local space of the character.
     *
     * Note I'm transposing this matrix for constancy but the
     * transformations below are the reverse of what I would expect.
     * Might be differences between diguy OGL and this.
     * Di-Guy 12.5 removed the transpose here, still a bit confused
     */
    osg::Matrixf inv_mat;
    get_inverse_rotation_matrix_4x4_ptr(inv_mat.ptr(), 0, 1);

    m_ufrm_inverse_transform_matrix->set(inv_mat);

    // FIXME shouldn't doing invert twice, but it's hard to access _mat in matrix
    // and short on time
    osg::Matrixf inv_mat_no_trans;
    get_inverse_rotation_matrix_4x4_ptr(inv_mat_no_trans.ptr(), 0, 0);

    // this needs clean up now that that inverse matrix is in the vid card
    osg::Light* light = s_viewer->getLight();
    if (light)
    {
        osg::Vec4 light_dir = light->getPosition();
        light_dir = inv_mat_no_trans * light_dir;
        m_ufrm_light_dir_os->set(light_dir); //uniform vec4
    }

    if (get_character())
    {
        diguyViewCamera* camera = get_character()->get_scenario()->get_scenario_camera();
        osg::Vec3 eye_point(camera->get_position_x(), camera->get_position_y(), camera->get_position_z());
        //float char_pos_x, char_pos_y, char_pos_z;
        //get_character()->get_position(
        //eye_point = eye_point - osg::Vec3(char_pos_x, char_pos_y, char_pos_z);
        //eye_point = inv_mat_no_trans*eye_point;
        eye_point = inv_mat*eye_point;
        m_ufrm_eye_point_os->set(eye_point); //uniform vec3
    }
}
}

```

Link Local Approach

```
#else
```

build

Build Stage

Description:

Override of `diguyGraphicsLink::build()`.

Look in the Character Local Approach `build()` comment above for what this function should do.

```
void
diguyOsgGraphicsLink::build()
{
    /*
     * Create a matrix that has the link's static offset transformations.
     * This matrix will not be changed during update() calls.
     */
    osg::Matrixf mat;
    osg::Matrixf mat_temp;

    mat.makeIdentity();

    /*
     * Add explicit offset translation.
     */
    float offset_t[3];
    get_offset_translation(

    mat_temp.makeTranslate(offset_t[0],
        offset_t[1],
        offset_t[2]);
    mat.preMult(mat_temp);

    /*
     * Add explicit offset rotation.
     */
    float offset_r[3];
    get_offset_rotation(

    if ((offset_r[0] != 0.0f) ||
        (offset_r[1] != 0.0f) ||
        (offset_r[2] != 0.0f))
    {
        mat_temp.makeRotate(offset_r[2] * CONSTANT_DEG2RAD,
            osg::Vec3f(0.0f, 1.0f, 0.0f),
            offset_r[1] * CONSTANT_DEG2RAD,
            osg::Vec3f(1.0f, 0.0f, 0.0f),
            offset_r[0] * CONSTANT_DEG2RAD,
            osg::Vec3f(0.0f, 0.0f, 1.0f));
        mat.preMult(mat_temp);
    }
}
```

```

/*
 * The runtime translation values of each link may be marked as always
 * static. If this is the case for this link, add the runtime
 * translation into the offset matrix.
 */
if (get_translation_is_static())
{
    float t[3];
    get_translation(

        mat_temp.makeTranslate(t[0],
            t[1],
            t[2]);
    mat.preMult(mat_temp);
}

/*
 * Runtime rotation values may also be marked as static.
 */
if (get_rotation_is_static())
{
    float r[3];
    get_rotation(

        if ((r[0] != 0.0f) ||
            (r[1] != 0.0f) ||
            (r[2] != 0.0f))
        {
            mat_temp.makeRotate(r[2] * CONSTANT_DEG2RAD,
                osg::Vec3f(0.0f, 1.0f, 0.0f),
                r[1] * CONSTANT_DEG2RAD,
                osg::Vec3f(1.0f, 0.0f, 0.0f),
                r[0] * CONSTANT_DEG2RAD,
                osg::Vec3f(0.0f, 0.0f, 1.0f));
            mat.preMult(mat_temp);
        }
    }

/*
 * Runtime scale values may also be marked as static.
 */
if (get_scale_is_static())
{
    float s[3];
    get_scale(

        if ((s[0] != 1.0f) ||
            (s[1] != 1.0f) ||
            (s[2] != 1.0f))
        {
            mat_temp.makeScale(osg::Vec3f(s[0], s[1], s[2]));
            mat.preMult(mat_temp);
        }
    }

/*
 * Build an offset node for this link based on the above computed offset
 * matrix.
 */
m_offset_node = new osg::MatrixTransform;
m_offset_node->setMatrix(mat);

```

```

/*
 * Build a DOF (degree of freedom) matrix transform for this link that
 * will be updated each frame from motion data.
 *
 * This DOF node will be updated with link-local transform data.
 */
m_dof_node = new osg::MatrixTransform;

m_offset_node->addChild(m_dof_node);

/*
 * See if we should attach to parent link. The top-level position link
 * should not start out attached. When necessary it will be attached to
 * the scene graph in the attach_to_scene() function.
 */
if (!get_is_position_link())
{
    diguyGraphicsLink* parent_link = get_parent_link();
    diguyOsgGraphicsLink* osg_parent_link =
        (diguyOsgGraphicsLink *) parent_link;
    osg_parent_link->m_dof_node->addChild(m_offset_node);
}
}

```

unbuild

Unbuild Stage

Description:

Override of `diguyGraphicsLink::unbuild()`.

Look in the Character Local Approach `unbuild()` comment above for what this function should do.

```

void
diguyOsgGraphicsLink::unbuild()
{
    m_offset_node->removeChild(m_dof_node);

    if (!get_is_position_link())
    {
        diguyGraphicsLink* parent_link = get_parent_link();
        diguyOsgGraphicsLink* osg_parent_link =
            (diguyOsgGraphicsLink *) parent_link;

        osg_parent_link->m_dof_node->removeChild(m_offset_node);
    }

    /*
     * Setting the m_offset_node and m_dof_node ref_ptrs to NULL
     * dereferences the objects they point to, possibly deleting them.
     */
    m_offset_node = NULL;
    m_dof_node = NULL;
}

```

update*Update Stage***Description:**

Override of `diguyGraphicsLink::update()`.

Look in the Character Local Approach `update()` comment above for what this function should do.

```
void
diguyOsgGraphicsLink::update()
{
    /*
     * Update the scene graph nodes created during build() with new values.
     *
     * These values should be obtained by calling get_translation(),
     * get_rotation(), and get_scale().
     */
    osg::Matrixf mat;
    osg::Matrixf mat_temp;

    /*
     * Start with identity.
     */
    mat.makeIdentity();

    if (!get_translation_is_static())
    {
        float t[3];
        get_translation(

            mat_temp.makeTranslate(t[0],
                                   t[1],
                                   t[2]);
        mat.preMult(mat_temp);
    }

    if (!get_rotation_is_static())
    {
        float r[3];
        get_rotation(

            if ((r[0] != 0.0f) ||
                (r[1] != 0.0f) ||
                (r[2] != 0.0f))
            {
                mat_temp.makeRotate(r[2] * CONSTANT_DEG2RAD,
                                     osg::Vec3f(0.0f, 1.0f, 0.0f),
                                     r[1] * CONSTANT_DEG2RAD,
                                     osg::Vec3f(1.0f, 0.0f, 0.0f),
                                     r[0] * CONSTANT_DEG2RAD,
                                     osg::Vec3f(0.0f, 0.0f, 1.0f));
                mat.preMult(mat_temp);
            }
    }

    if (!get_scale_is_static())
```

```

{
    float s[3];
    get_scale(

        if ((s[0] != 1.0f) ||
            (s[1] != 1.0f) ||
            (s[2] != 1.0f))
        {
            mat_temp.makeScale(osg::Vec3f(s[0], s[1], s[2]));
            mat.preMult(mat_temp);
        }
    }

    /*
     * Set the m_dof_node matrix.
     */
    m_dof_node->setMatrix(mat);
}

```

Common to Character Local and Link Local Approach

```
#endif
```

make_bounding_sphere_visual

Description:

Utility function to create a sphere visual.

```

osg::Geode* diguyOsgGraphicsLink::make_bounding_sphere_visual(osg::Vec3&center, float radius)
{
    osg::ref_ptr<osg::PolygonMode> polygon_mode = new osg::PolygonMode;
    polygon_mode->setMode(osg::PolygonMode::FRONT_AND_BACK, osg::PolygonMode::LINE);

    osg::ref_ptr<osg::ShapeDrawable> shape = new osg::ShapeDrawable;
    shape->setShape(new osg::Sphere(center, radius));
    osg::Uniform* uniform = new osg::Uniform(osg::Uniform::FLOAT_MAT4, "ufrn_link_matrix");
    osg::Matrixf mat;
    mat.makeIdentity();
    uniform->set(mat);

    osg::Geode* bounding_sphere_visual = new osg::Geode;
    bounding_sphere_visual->getOrCreateStateSet()->setAttribute(polygon_mode);
    bounding_sphere_visual->getOrCreateStateSet()->addUniform(uniform);
    bounding_sphere_visual->addDrawable(shape);

    return bounding_sphere_visual;
}

```

InitialBoundVisitor

Description:

DI-Guys are modeled with parts (hands, guns, etc...) at the origin. These parts are rendered using a uniform in the shader unavailable to the OSG cull traversal. The InitialBoundVisitor

is a NodeVisitor that adds a initial bound (osg::BoundingBox) to the drawable of geode nodes to ensure parts are not improperly culled.

```
class InitialBoundVisitor : public osg::NodeVisitor
{
public:
    InitialBoundVisitor(float edge_length) : m_edge_length(edge_length)
    {
        setTraversalMode(osg::NodeVisitor::TRAVERSE_ALL_CHILDREN);
    }

    virtual void apply(osg::Geode&geode)
    {
        const float len = m_edge_length;
        const osg::BoundingBoxf bbox(osg::Vec3f(-len, -len, -len), osg::Vec3f(len, len, len));

        const unsigned int num = geode.getNumDrawables();
        for (unsigned int i = 0; i < num; ++i)
        {
            osg::Drawable* drawable = geode.getDrawable(i);
            const osg::BoundingBox&curr_bbox = drawable->getInitialBound();
            if (!curr_bbox.valid())
            {
                /*
                 * Drawables are shared between character instances.
                 * Only set the initial bound of the drawable if it hasn't
                 * already been set. This avoids dirtying the bound and forcing
                 * recomputation of bounding info for other characters sharing
                 * this drawable.
                 */
                drawable->setInitialBound(bbox);
            }
        }

        traverse(geode);
    }

private:
    float m_edge_length;
};
```

attach_to_scene

Update Stage

Description:

Override of `diguyGraphicsLink::attach_to_scene()`.

The `attach_to_scene()` function should perform the actions necessary to attach this link to somewhere on the scene graph.

This implementation attaches this link's offset node as a child to the node specified by `s_diguy_attach_pt`.

This function will only be called on the position link.

```

void
diguyOsgGraphicsLink::attach_to_scene()
{
    /*
     * Need m_offset_node to do attach.
     */
    if (!m_offset_node)
        return;

    if (m_offset_node->getNumParents() > 0)
    {
        bdi_log_printf(BDI_LOG_WARN,
            "WARNING: Link '%s' already attached to something.  "
            "Call to attach_to_scene() failed.\n",
            get_name());
        return;
    }

    /*
     * Make the attachment.
     */
    if (s_diguy_attach_pt)
    {
        diguyCharacter* character = get_character();
        if (character)
        {
            /*
             * Add an initial bound (bounding box) to all drawables
             * of the character such that parts cull correctly.
             */
            const float radius = character->get_default_bounding_radius();
            const float len = sqrt((radius*radius)/3.0f);
            InitialBoundVisitor initialBoundVisitor(len);
            m_offset_node->accept(initialBoundVisitor);

#define SHOW_BOUNDING_SPHERE 0
#if SHOW_BOUNDING_SPHERE
            // Show bounding sphere around characters.
            osg::BoundingSphere bSphere = m_offset_node->getBound();
            m_bounding_sphere_visual = make_bounding_sphere_visual(bSphere.center(), bSphere.radius());

            diguyGraphicsLink* position_link = character->get_position_link();
            diguyOsgGraphicsLink* osg_position_link = (diguyOsgGraphicsLink *) position_link;

            osg_position_link->m_dof_node->addChild(m_bounding_sphere_visual);
#endif
        }

        s_diguy_attach_pt->addChild(m_offset_node);
    }
}

```

detach_from_scene

Update Stage

Description:

Override of `diguyGraphicsLink::detach_from_scene()`.

The `detach_to_scene()` function should perform the actions necessary to detach this link from the scene graph.

This implementation removes this link's offset node as a child from the node specified by `s_diguy_attach_pt`.

This function will only be called on the position link.

```
void
diguyOsgGraphicsLink::detach_from_scene()
{
    /*
     * Need m_offset_node to do detach.
     */
    if (!m_offset_node)
        return;

    /*
     * If m_offset_node has no parents, it is already detached.
     */
    if (m_offset_node->getNumParents() == 0)
    {
        bdi_log_printf(BDI_LOG_WARN,
            "WARNING: Link '%s' not attached to anything.  "
            "Call to detach_from_scene() failed.\n",
            get_name());
        return;
    }

    if (s_diguy_attach_pt)
    {
        s_diguy_attach_pt->removeChild(m_offset_node);
        m_dof_node->removeChild(m_bounding_sphere_visual);
        m_bounding_sphere_visual = NULL;
    }
}
```

attach_to_link

Update Stage

Description:

Override of `diguyGraphicsLink::attach_to_link()`.

This function is similar to `attach_to_scene()`, but the link's object should be attached to another link instead of the scene graph.

```
void
diguyOsgGraphicsLink::attach_to_link(diguyGraphicsLink* link)
{
    /*
     * Need m_offset_node to do attach.
     */
```

```

if (!m_offset_node)
    return;

/*
 * If m_offset_node already has parents, it is already attached.
 */
if (m_offset_node->getNumParents() > 0)
{
    bdi_log_printf(BDI_LOG_WARN,
        "WARNING: Link '%s' already attached to something.  "
        "Call to attach_to_link() failed.\n",
        get_name());
    return;
}

/*
 * Attach m_offset_node of this link to m_dof_node of the
 * attachment link.
 */
diguyOsgGraphicsLink* osg_link = (diguyOsgGraphicsLink*) link;
if (osg_link->m_dof_node)
{
    osg_link->m_dof_node->addChild(m_offset_node);
}
}

```

detach_from_link

Update Stage

Description:

Override of `diguyGraphicsLink::detach_from_link()`.

This function is similar to `detach_from_scene()`, but the link's object should be detached from another link instead of the scene graph.

```

void
diguyOsgGraphicsLink::detach_from_link()
{
    /*
     * Need m_offset_node to do detach.
     */
    if (!m_offset_node)
        return;

    /*
     * If m_offset_node has no parents, it is already detached.
     */
    if (m_offset_node->getNumParents() == 0)
    {
        bdi_log_printf(BDI_LOG_WARN,
            "WARNING: Link '%s' not attached to anything.  "
            "Call to detach_from_link() failed.\n",
            get_name());
        return;
    }
}

```

```

diguyGraphicsLink* link = get_attached_to_link();

diguyOsgGraphicsLink* osg_link = (diguyOsgGraphicsLink*) link;
if (osg_link->m_dof_node)
{
    osg_link->m_dof_node->removeChild(m_offset_node);
}
}

/*
 * Create a default variation object, and load it with a 1x1 place holder image
 */
osg::ref_ptr<osg::Texture2D>
diguyOsgGraphicsLink::get_default_variation_texture()
{
    if(sm_degenerate_texture){
        return sm_degenerate_texture;
    }
    /*
     * Theres a small chance that this might leak a 1x1 pixel texture at shut down.
     */

    sm_degenerate_texture = new osg::Texture2D();

    GLenum pixel_format = GL_RGBA;
    GLint internal_texture_format = GL_RGBA;

    /*
     * Make the zero change variation texture object. The osg image will take
     * ownership of it.
     */

    unsigned char* texture_map_data = new unsigned char [4];
    texture_map_data[0] = 128;
    texture_map_data[1] = 128;
    texture_map_data[2] = 128;
    texture_map_data[3] = 255;

    /*
     * Create an osg image.
     */
    osg::Image* osg_image = new osg::Image();

    osg_image->setImage(1,
        1,
        1,
        internal_texture_format,
        pixel_format,
        GL_UNSIGNED_BYTE,
        texture_map_data,
        osg::Image::USE_NEW_DELETE, // osg will deallocate texture_map_data_copy with c
        4); // pack/unpack alignment

    // not sure this is needed
    sm_degenerate_texture->setUnRefImageDataAfterApply(true);

    sm_degenerate_texture->setWrap(osg::Texture2D::WRAP_S, osg::Texture2D::CLAMP);

    sm_degenerate_texture->setWrap(osg::Texture2D::WRAP_T, osg::Texture2D::CLAMP);

    sm_degenerate_texture->setFilter(osg::Texture2D::MAG_FILTER, osg::Texture2D::NEAREST);

```

```

sm_degenerate_texture->setFilter(osg::Texture2D::MIN_FILTER, osg::Texture2D::NEAREST);

sm_degenerate_texture->setResizeNonPowerOfTwoHint(true);
sm_degenerate_texture->setImage(osg_image);

return sm_degenerate_texture;
}

```

set_shader_instance

Update Stage

Description:

Override of `diguyGraphicsLink::set_shader_instance()`.

The `set_shader_instance()` function should perform the actions necessary to associate the passed shader instance with this link's node object.

This implementation updates the offset node's state set to use the passed shader's state set.

```

void
diguyOsgGraphicsLink::set_shader_instance(diguyGraphicsShaderInstance* shader_instance)
{
    diguyCharacter* character = get_character();
    if (!character)
        return;

    /*
     * Convert the passed shader instance to an OSG-specific object.
     */
    m_diguy_osg_shader_instance = (diguyOsgGraphicsShaderInstance*) shader_instance;
    m_diguy_osg_shader_program = NULL;
    osg::ref_ptr<osg::StateSet> shader_state_set = NULL;
    if (m_diguy_osg_shader_instance)
    {
        m_diguy_osg_shader_program = (diguyOsgGraphicsShaderProgram*) m_diguy_osg_shader_instance;

        bdi_log_printf(BDI_LOG_INFO, "INFO: Setting character '%s' shader instance to '%s'",
            character->get_name(),
            m_diguy_osg_shader_program->get_name());

        shader_state_set = new osg::StateSet(*m_diguy_osg_shader_instance->get_state_set());

        /*
         * Add a default variation texture to the node, the shapes can override this
         * on a per shape basis.
         */
        osg::ref_ptr<osg::Texture2D> texture = get_default_variation_texture();
        shader_state_set->setTextureAttribute(DIGUY_TEXTURE_MAP_TYPE_COMPONENT_RAMP, texture);

        shader_state_set->setTextureMode(DIGUY_TEXTURE_MAP_TYPE_COMPONENT_RAMP,
            GL_TEXTURE_2D,
            osg::StateAttribute::PROTECTED | osg::StateAttribute::ON);

        m_offset_node->setStateSet(shader_state_set);
    }
    else

```

```

{
    bdi_log_printf(BDI_LOG_INFO, "INFO: Un-setting character '%s' shader instance.\n",
        character->get_name());

    m_offset_node->setStateSet(NULL);
}

/*
 * Ideally we'd only set this if m_ufrm_link_matrix is valid for the
 * shader. However it doesn't appear to be trivial to identify that in
 * OSG. Make a unique shader state per link that supports segmented
 * geometry.
 */
if (shader_state_set)
{
    int i;
    for (i = 0; i < character->get_num_links(); i++)
    {
        diguyOsgGraphicsLink* link = ((diguyOsgGraphicsLink *) character->get_link_at(i));

        if (!link->m_dof_node)
            continue;

        osg::StateSet* child_shader_state_set = new osg::StateSet(*shader_state_set);

        link->m_ufrm_link_matrix = new osg::Uniform(osg::Uniform::FLOAT_MAT4, "ufrm_link_matrix", 16);
        child_shader_state_set->addUniform(link->m_ufrm_link_matrix);

        if (link->get_is_position_link())
        {
            link->m_ufrm_light_dir_os = new osg::Uniform(osg::Uniform::FLOAT_VEC4, "ufrm_light_dir", 4);
            child_shader_state_set->addUniform(link->m_ufrm_light_dir_os);

            link->m_ufrm_eye_point_os = new osg::Uniform(osg::Uniform::FLOAT_VEC3, "ufrm_eye_point", 3);
            child_shader_state_set->addUniform(link->m_ufrm_eye_point_os);

            link->m_ufrm_inverse_transform_matrix = new osg::Uniform(osg::Uniform::FLOAT_MAT4, "ufrm_inverse_transform_matrix", 16);
            child_shader_state_set->addUniform(link->m_ufrm_inverse_transform_matrix);
        }
        link->m_dof_node->setStateSet(child_shader_state_set);
    }
}
}

```

Create function definition

Create and return an object of a subclass of [diguyGraphicsLink](#). This function will be called whenever DI-Guy requires a link object.

```

diguyGraphicsLink*
diguyOsgGraphicsLink_create_func(void* internal_data)
{
    return new diguyOsgGraphicsLink(internal_data);
}

```

5.63 DI-Guy Graphics API, diguyOsgGraphicsMaterial

diguyOsgGraphicsMaterial.cpp

DI-Guy SDK Version 12.5.2

This file was automatically generated from `diguyOsgGraphicsMaterial.cpp`. Do not edit this file directly; the changes will be lost.

Introduction

Materials modify the color and lighting characteristics of meshes. They work in concert with textures to determine the final appearance of visible geometry.

The class **diguyOsgGraphicsMaterial** is an example of a [diguyGraphicsMaterial](#) subclass for an **scene graph renderer** using the DI-Guy Loader. The example subclass shows how to use the accessors of the [diguyGraphicsMaterial](#) class and which virtual functions should be overridden for an scene graph renderer such as Open Scene Graph.

Header files and forward declarations

```
#include "diguyOsgGraphicsMaterial.h"
```

Constructor

Description:

Initialize data members of the class to a known state.

Most base class data will not be fully initialized until during the Build Stage, so most base class accessors should **not** be called here.

```

diguyOsgGraphicsMaterial::diguyOsgGraphicsMaterial(void* internal_data)
:
    diguyGraphicsMaterial(internal_data)
{
    /*
     * Nothing else to do after calling base class constructor and
     * initializing data members.
     */
}

```

build

Build Stage

Description:

Override of `diguyGraphicsMaterial::build()`.

The `build()` function should create renderer objects that will set the proper material state in the scene graph traversal.

This implementation creates an `osg::Material` object, and initializes it from material properties retrieved using `diguyGraphicsMaterial` accessors.

```

void
diguyOsgGraphicsMaterial::build()
{
    float ambient[4];
    float diffuse[4];
    float specular[4];
    float emission[4];
    float shininess;

    get_ambient(ambient);
    get_diffuse(diffuse);
    get_specular(specular);
    get_emission(emission);
    get_shininess(

    m_material = new osg::Material;

    m_material->setAmbient(osg::Material::FRONT_AND_BACK,
        osg::Vec4(ambient[0],
            ambient[1],
            ambient[2],
            ambient[3]));
    m_material->setDiffuse(osg::Material::FRONT_AND_BACK,
        osg::Vec4(diffuse[0],
            diffuse[1],
            diffuse[2],
            diffuse[3]));
    m_material->setSpecular(osg::Material::FRONT_AND_BACK,
        osg::Vec4(specular[0],
            specular[1],
            specular[2],
            specular[3]));
    m_material->setEmission(osg::Material::FRONT_AND_BACK,

```



```

        osg::Vec4(emission[0],
            emission[1],
            emission[2],
            emission[3]));
    m_material->setShininess(osg::Material::FRONT_AND_BACK,
        shininess);
}

```

unbuild

Unbuild Stage

Description:

Override of `diguyGraphicsMaterial::unbuild()`.

The `unbuild()` function should destroy the objects created in the `build()` function.

This implementation dereferences the `osg::Material` created in `build()`.

```

void
diguyOsgGraphicsMaterial::unbuild()
{
    /*
     * Setting the m_material ref_ptr to NULL dereferences the object it
     * points to, possibly deleting it.
     */
    m_material = NULL;
}

```

Create function definition

Create and return an object of a subclass of `diguyGraphicsMaterial`.

```

diguyGraphicsMaterial*
diguyOsgGraphicsMaterial_create_func(void* internal_data)
{
    return new diguyOsgGraphicsMaterial(internal_data);
}

```

5.64 DI-Guy Graphics API, diguyOsgGraphicsMesh

diguyOsgGraphicsMesh.cpp

DI-Guy SDK Version 12.5.2

This file was automatically generated from `diguyOsgGraphicsMesh.cpp`. Do not edit this file directly; the changes will be lost.

Introduction

Meshes contain the visual geometry of a DI-Guy character. They are attached to shapes, and are associated with materials and textures.

The class **diguyOsgGraphicsMesh** is an example of a **diguyGraphicsMesh** subclass for a scene graph renderer using the DI-Guy Loader. The example subclass shows how to use the accessors of the **diguyGraphicsMesh** class and which virtual functions should be overridden for a scene graph renderer such as OSG.

Header files and forward declarations

```
#include "diguyOsgGraphicsMesh.h"
#include "diguyOsgGraphicsShaderProgram.h"
#include "diguyOsgGraphicsState.h"

#include <osg/Geometry>
#include <osg/ShapeDrawable>

#include <stdlib.h>

#include <libbdilog.h>
```

Constructor

Description:

This constructor initializes data members of the class to a known state.

Base class data will not be fully initialized until during the Build Stage, so base class accessors should **not** be called here.

```
diguyOsgGraphicsMesh::diguyOsgGraphicsMesh(void* internal_data)
:
    diguyGraphicsMesh(internal_data)
{
    /*
     * Nothing else to do after calling base class constructor and
     * initializing data members.
     */
}
```

build

Build Stage

Description:

Override of **diguyGraphicsMesh::build()**.

The **build()** function should create objects necessary to render the visible geometry of this mesh in the Draw Stage.

This implementation creates `osg::Group` and `osg::Drawable` nodes that are attached to the scene graph.

```
void
diguyOsgGraphicsMesh::build()
{
    /*
     * Create the first geometry drawable object.
     */
    osg::Geometry* geometry = new osg::Geometry;
    osg::ref_ptr<osg::Drawable> drawable_ptr = geometry;

    m_drawables.push_back(drawable_ptr);
    m_has_graphics_state_set.push_back(false);

    /*
     * If not skinned or deformable, it's static.
     */
    if (get_is_skinned_mesh() || get_is_deformable())
        geometry->setDataVariance(osg::Object::DYNAMIC);
    else
        geometry->setDataVariance(osg::Object::STATIC);

    /*
     * Prefer VB0s.
     */
    geometry->setUseVertexBufferObjects(true);

    /*
     * Vertices.
     */
    int nv = get_vertex_count();

    osg::Vec3Array* vertices = new osg::Vec3Array;
    geometry->setVertexArray(vertices);

    int vstride = get_vertex_stride() / sizeof(float);
    float* vptr = get_first_vertex();

    /*
     * Long way for now, adding data vertex by vertex. I'd like to find
     * the OSG way for simply copying a binary block of data containing
     * float vertex data into the vertices array.
     */
    vertices->reserve(nv);

    int index;
    for (index=0; index<nv; index++)
    {
        vertices->push_back(osg::Vec3(*vptr, *(vptr+1), *(vptr+2)));

        // note that vstride is in bytes, so be careful with pointer arithmetic
        vptr += vstride;
    }

    /*
     * Normals.
     */
    osg::Vec3Array* normals = new osg::Vec3Array;
    geometry->setNormalArray(normals);
}
```

```

int nstride = get_normal_stride() / sizeof(float);
float* nptr = get_first_normal();

normals->reserve(nv);

for (index=0; index<nv; index++)
{
    normals->push_back(osg::Vec3(*nptr, *(nptr+1), *(nptr+2)));
    nptr += nstride;
}

geometry->setNormalBinding(osg::Geometry::BIND_PER_VERTEX);

/*
 * Texture indices.
 */
static const int texture_unit = 0;

osg::Vec2Array* UVs = new osg::Vec2Array;
geometry->setTexCoordArray(texture_unit, UVs);

int tstride = get_texture_indices_stride() / sizeof(float);
float* tptr = get_first_texture_indices();

UVs->reserve(nv);

for (index=0; index<nv; index++)
{
    UVs->push_back(osg::Vec2(*tptr, *(tptr+1)));
    tptr += tstride;
}

/*
 * Polygon indices.
 */
int ni = get_index_count();

unsigned int* uint_ptr = NULL;
unsigned short* ushort_ptr = NULL;

if (get_index_type() == DIGUY_GRAPHICS_INDEX_TYPE_UINT)
    uint_ptr = (unsigned int*) get_index_buffer_data();
else if (get_index_type() == DIGUY_GRAPHICS_INDEX_TYPE_USHORT)
    ushort_ptr = (unsigned short*) get_index_buffer_data();

osg::DrawElementsUInt* index_set =
    new osg::DrawElementsUInt(osg::PrimitiveSet::TRIANGLES, 0);
geometry->addPrimitiveSet(index_set);

index_set->reserve(ni);

for (index=0; index<ni; index++)
{
    if (uint_ptr)
    {
        index_set->push_back(*uint_ptr);
        uint_ptr++;
    }
    else if (ushort_ptr)
    {
        index_set->push_back((unsigned int) *ushort_ptr);
    }
}

```

```

        ushort_ptr++;
    }
}

/*
 * If the mesh is a skinned mesh, we need to add some GLSL attributes.
 */
if (get_is_skinned_mesh())
{
    geometry->setSupportsDisplayList(false);

    /*
     * attr_link_weight GLSL attribute.
     */
    osg::Vec4Array* link_weight_array = new osg::Vec4Array;
    geometry->setVertexAttribArray(osg::Drawable::ATTRIBUTE_6, link_weight_array);
    geometry->setVertexAttribBinding(osg::Drawable::ATTRIBUTE_6, osg::Geometry::BIND_F);

    int lw_stride = get_link_weight_stride() / sizeof(float);
    float* lw_ptr = get_first_link_weight();

    link_weight_array->reserve(nv);

    for (index=0; index<nv; index++)
    {
        link_weight_array->push_back(osg::Vec4(*lw_ptr, *(lw_ptr+1), *(lw_ptr+2), *(lw_ptr+3));
        lw_ptr += lw_stride;
    }

    /*
     * attr_link_index GLSL attribute.
     */
    osg::Vec4Array* link_index_array = new osg::Vec4Array;
    geometry->setVertexAttribArray(osg::Drawable::ATTRIBUTE_7, link_index_array);
    geometry->setVertexAttribBinding(osg::Drawable::ATTRIBUTE_7, osg::Geometry::BIND_F);

    float* li_ptr = get_first_link_index();
    int li_stride = get_link_index_stride() / sizeof(float);

    link_index_array->reserve(nv);

    for (index=0; index<nv; index++)
    {
        link_index_array->push_back(osg::Vec4(*li_ptr, *(li_ptr+1), *(li_ptr+2), *(li_ptr+3));
        li_ptr += li_stride;
    }
}

if ((get_vertex_format() == DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_T4_I4F_W4) ||
    (get_vertex_format() == DIGUY_GRAPHICS_VERTEX_FORMAT_V3_N3_T2_T4))
{
    /*
     * attr_tangents GLSL attribute. Using secondary colors as the attrib
     */
    osg::Vec4Array* vertex_tangent_array = new osg::Vec4Array;
    geometry->setVertexAttribArray(osg::Drawable::SECONDARY_COLORS, vertex_tangent_array);
    geometry->setVertexAttribBinding(osg::Drawable::SECONDARY_COLORS, osg::Geometry::BIND_F);

    float* li_ptr = get_first_tangent();
    int li_stride = get_tangent_stride() / sizeof(float);

```

```

vertex_tangent_array->reserve(nv);

for (index=0; index<nv; index++)
{
    vertex_tangent_array->push_back(osg::Vec4(*li_ptr, *(li_ptr+1), *(li_ptr+2),
    li_ptr += li_stride;
}

}

/*
 * Set state sets of drawables.
 *
 * If there is more than one state set for this mesh (e.g., for
 * switched diguy graphics states), we need to create separate
 * drawables for each switched graphics state. The meshes will be
 * substantially the same, except for which state set is attached to
 * them.
 */
int num_switched_graphics_states = get_num_switched_graphics_states();

osg::Geometry* base_geometry = geometry;
osg::Geometry* current_geometry = base_geometry;

int switch_state_index;

for (switch_state_index = 1;
    switch_state_index < num_switched_graphics_states;
    switch_state_index++)
{
    /*
     * Clone the geometry of the first, "base" mesh.
     * The mesh with switch state 0 uses the original base mesh.
     */
    current_geometry = new osg::Geometry(*base_geometry);

    osg::ref_ptr<osg::Drawable> drawable_ptr = current_geometry;
    m_drawables.push_back(drawable_ptr);
    m_has_graphics_state_set.push_back(false);
}

/*
 * Free memory allocated by the loader.
 */
//free_loader_memory();
}

```

unbuild

Unbuild Stage

Description:

Override of `diguyGraphicsMesh::unbuild()`.

The `unbuild()` function should delete the objects created in the `build()` function.

This implementation dereferences the nodes created in `build()`.

```

void
diguyOsgGraphicsMesh::unbuild()
{
    /*
     * Setting the m_drawable ref_ptr to NULL dereferences the
     * Geometry object it points to, possibly deleting it.
     */
    unsigned int i;
    for (i = 0; i < m_drawables.size(); i++)
        m_drawables[i] = NULL;

    m_drawables.clear();
}

```

get_num_drawables

Description:

Class-local accessor function for number of drawables objects.

```

int
diguyOsgGraphicsMesh::get_num_drawables()
{
    return m_drawables.size();
}

```

get_num_drawables

Description:

Class-local accessor function for pointers to drawables objects.

```

osg::Drawable*
diguyOsgGraphicsMesh::get_drawable(int drawable_index)
{
    if (m_drawables.size() == 0)
        return NULL;

    if ((drawable_index < 0) || (drawable_index >= (int) m_drawables.size()))
        drawable_index = 0;

    if (!m_has_graphics_state_set[drawable_index])
    {
        attach_state_set_to_drawable(drawable_index);
        m_has_graphics_state_set[drawable_index] = true;
    }

    osg::Drawable* drawable = m_drawables[drawable_index].get();

    return drawable;
}

```

attach_state_set_to_drawable

Description:

Looks up the state set for the drawable and attaches the state set to the drawable if not already attached.

Allows for just in time loading of state sets (specifically textures) associated with the drawable.

```
void
diguyOsgGraphicsMesh::attach_state_set_to_drawable(int drawable_index)
{
    /*
     * Get a pointer to the graphics state. Attach it to the created
     * geometry object. If this mesh has switched graphics states, ...
     */
    diguyGraphicsState* diguy_state = get_state(drawable_index);
    diguyOsgGraphicsState* diguy_osg_state = (diguyOsgGraphicsState*) diguy_state;

    if (diguy_osg_state)
    {
        /*
         * Get the osg StateSet of this mesh's diguyGraphicsState.
         */
        osg::StateSet* diguy_graphics_state_state_set = diguy_osg_state->get_state_set();

        if (diguy_graphics_state_state_set)
        {
            osg::ref_ptr<osg::Drawable> drawable_ptr = m_drawables[drawable_index];
            osg::Geometry* current_geometry = dynamic_cast<osg::Geometry*>(drawable_ptr.get());

            /*
             * If this is a skinned mesh, we need to make a copy of
             * the returned state set, and add a GLSL uniform to it.
             * Otherwise we can use it as-is.
             */
            if (get_is_skinned_mesh())
            {
                osg::StateSet* state_set_copy = new osg::StateSet(*diguy_graphics_state_state_set);
                osg::Uniform* ufrm_apply_matrices_flag = new osg::Uniform("ufrm_apply_matrices");
                state_set_copy->addUniform(ufrm_apply_matrices_flag);
                current_geometry->setStateSet(state_set_copy);
            }
            else
            {
                current_geometry->setStateSet(diguy_graphics_state_state_set);
            }
        }
        else
        {
            bdi_log_printf(BDI_LOG_WARN, "WARNING: DI-Guy Graphics state invalid.\n");
        }
    }
}
```

Create function definition

Create and return an object of a subclass of [diguyGraphicsMesh](#). This function will be called whenever DI-Guy requires a mesh object.


```
diguyGraphicsMesh*
diguyOsgGraphicsMesh_create_func(void* internal_data)
{
    return new diguyOsgGraphicsMesh(internal_data);
}
```

5.65 DI-Guy Graphics API, diguyOsgGraphicsShaderInstance

diguyOsgGraphicsShaderInstance.cpp

DI-Guy SDK Version 12.5.2

This file was automatically generated from diguyOsgGraphicsShaderInstance.cpp. Do not edit this file directly; the changes will be lost.

Introduction

Unlike *shader programs*, which are shared by many characters, shader instances allow non-shared, character-specific shader state to be maintained. An example is having different uniform variable values for different characters, such as a highlight color that is applied during shading.

See *oglGraphicsShaderProgram* for information on shader programs.

diguyOsgGraphicsShaderInstance is an example of a [diguyGraphicsShaderInstance](#) subclass for a scene graph renderer using the DI-Guy Loader. The example subclass shows how to use the accessors of the [diguyGraphicsShaderInstance](#) class and which virtual functions should be overridden for a scene graph renderer such as OSG.

Header files and forward declarations

```
#include <stdlib.h>
#include <stdio.h>

#include <diguyApp.h>
#include "diguyOsgGraphicsShaderInstance.h"
#include "diguyOsgGraphicsShaderProgram.h"
```

```
#include "diguyGraphicsTexture.h"
#include <diguyGraphicsShaderTechnique.h>
#include <diguyCharacter.h>
#include <libbdilog.h>

#include <osg/ShadeModel>
```

Constructor

Description:

Initialize data members of the class to a known state.

Most base class data will not be fully initialized until during the Build Stage, so most base class accessors should **not** be called here.

```
diguyOsgGraphicsShaderInstance::diguyOsgGraphicsShaderInstance(void* internal_data)
:
    diguyGraphicsShaderInstance(internal_data),
    m_shader_state_set(NULL),
    m_ufrm_apply_matrices_flag(NULL),
    m_ufrm_textures_enabled(NULL)
{
    /*
     * Delay creating state set and uniforms until the state set
     * is requested by get_state_set().
     */
}
```

Destructor

```
diguyOsgGraphicsShaderInstance::~~diguyOsgGraphicsShaderInstance()
{
    /*
     * Dereference state set and uniforms.
     */
    m_shader_state_set = NULL;

    m_ufrm_apply_matrices_flag = NULL;
    m_ufrm_textures_enabled = NULL;
}
```

get_state_set

Description:

Class-local accessor function for OSG state set of the shader instance. The OSG state set is created if it doesn't yet exist.

```
osg::StateSet*
diguyOsgGraphicsShaderInstance::get_state_set()
{
    /*
     * If m_shader_state_set has already been created, return it.
     */
    if (m_shader_state_set != NULL)
        return m_shader_state_set;
```

```

/*
 * Create a new osg::StateSet.
 */
m_shader_state_set = new osg::StateSet;

/*
 * Add some uniform variables.
 */
if (get_shader_program()->get_shader_type() == DIGUY_GRAPHICS_SHADER_PROGRAM_TYPE_SKIN
{
    /*
     * In order to use the same shader program for both skinned and
     * segmented meshes, we need a way to tell the shader whether or not
     * to apply the ufrm_link_matrices values. This provides a default
     * value that gets overridden on a per mesh basis.
     */
    m_ufrm_apply_matrices_flag = new osg::Uniform("ufrm_apply_matrices_flag", 0.0f);
    m_shader_state_set->addUniform(m_ufrm_apply_matrices_flag);
}

m_ufrm_textures_enabled = new osg::Uniform("ufrm_textures_enabled", 1);
m_shader_state_set->addUniform(m_ufrm_textures_enabled);

/*
 * Set up texture uniforms.
 */
osg::Uniform* diffuse_map = new osg::Uniform("ufrm_diffuse_map", DIGUY_TEXTURE_MAP_TY
m_shader_state_set->addUniform(diffuse_map);

osg::Uniform* bump_map = new osg::Uniform("ufrm_bump_map", DIGUY_TEXTURE_MAP_TYPE_BUM
m_shader_state_set->addUniform(bump_map);

osg::Uniform* specular_map = new osg::Uniform("ufrm_specular_map", DIGUY_TEXTURE_MAP_T
m_shader_state_set->addUniform(specular_map);

osg::Uniform* component_map = new osg::Uniform("ufrm_component_map", DIGUY_TEXTURE_MAP
m_shader_state_set->addUniform(component_map);

osg::Uniform* component_ramp_map = new osg::Uniform("ufrm_component_ramp", DIGUY_TEXTU
m_shader_state_set->addUniform(component_ramp_map);

/*
 * Attach shader_program to the state set.
 */
diguyGraphicsShaderTechnique* shader_technique = get_shader_technique();
if (!shader_technique)
{
    bdi_log_printf(BDI_LOG_WARN,
        "WARNING: Shader instance created without a shader technique.\n");
}
else
{
    diguyCharacter* character = get_character();

    diguyOsgGraphicsShaderProgram* shader_program = NULL;

    int render_flags = diguyApp::get_app()->get_render_pass_flags();
    if (render_flags & DIGUY_RENDER_PASS_INTERSECTION)
    {
        shader_program =

```

```

        static_cast<diguyOsgGraphicsShaderProgram*>(shader_technique->get_intersece

    osg::ref_ptr<osg::ShadeModel> shadeModel = new osg::ShadeModel(osg::ShadeModel
    m_shader_state_set->setAttributeAndModes(shadeModel.get(), osg::StateAttribute
}
else
{
    shader_program =
        static_cast<diguyOsgGraphicsShaderProgram*>(shader_technique->get_shader_

    if (character)
    {
        /*
         * TODO implement shader LODs for now just bump down to the
         * non-bump mapped shader.
         */
        if (!character->get_has_bump_maps() &
            (shader_program->get_quality_level() == DIGUY_GRAPHICS_SHADER_QUALITY_
            (shader_technique->get_num_shader_lod_programs() > 1))
        {
            shader_program = static_cast<diguyOsgGraphicsShaderProgram*>(shader_t
        }
    }
}

    m_shader_state_set->setAttributeAndModes(shader_program->get_program(), osg::Stat
}

    return m_shader_state_set.get();
}

```

Create function definition

Create and return an object of a subclass of [diguyGraphicsShaderInstance](#). This function will be called whenever DI-Guy requires a shader instance object.

```

diguyGraphicsShaderInstance*
diguyOsgGraphicsShaderInstance_create_func(void* internal_data)
{
    return new diguyOsgGraphicsShaderInstance(internal_data);
}

```

5.66 DI-Guy Graphics API, diguyOsgGraphicsShaderProgram

diguyOsgGraphicsShaderProgram.cpp

DI-Guy SDK Version 12.5.2

This file was automatically generated from diguyOsgGraphicsShaderProgram.cpp. Do not edit this file directly; the changes will be lost.

Introduction

Most rendering environments use some type of shader system that offloads rendering operations from the CPU to a GPU. Terminology varies, but DI-Guy uses the term "shader" to refer to a pair of sub-programs: a vertex shader and a pixel or fragment shader.

Shader programs are shared among characters; if two characters use the same shader, the same shader object is used for both. If character-specific data is needed it can be stored in a shader instance.

The class **diguyOsgGraphicsShaderProgram** is an example of a **diguyGraphicsShaderProgram** subclass for an scene graph renderer using the DI-Guy Loader. The example subclass shows how to use the accessors of the **diguyGraphicsShaderProgram** class and which virtual functions should be overridden for a scene graph renderer such as OSG.

Header files and forward declarations

```
#include <stdlib.h>
#include <stdio.h>

#include <fstream>

#include <osg/Drawable>
#include <osg/Version>

#include "diguyOsgGraphicsShaderProgram.h"
#include "diguyOsgGraphicsShaderInstance.h"
#include "diguyOsgGraphicsLink.h"
#include "diguyOsgGraphicsShape.h"
#include <diguyApp.h>
#include <libbdilog.h>

#include <string.h>
```

Constructor

Description:

Initialize data members of the class to a known state.

Most base class data is not fully initialized until during the Build Stage, so most base class accessors should **not** be called here.

```
diguyOsgGraphicsShaderProgram::diguyOsgGraphicsShaderProgram(void* internal_data)
:
    diguyGraphicsShaderProgram(internal_data),
    m_program(NULL),
    m_vertex_shader(NULL),
    m_fragment_shader(NULL)
{
    m_vertex_shader_filename[0] = NULL;
    m_fragment_shader_filename[0] = NULL;
```

```

/*
 * Create an OSG program.
 */
m_program = new osg::Program;
m_program->setName(get_name());

/*
 * Add vertex and fragment shaders to it. Note that source code is not
 * loaded here, but in load_vertex_shader_source() and
 * load_pixel_shader_source().
 */
m_vertex_shader = new osg::Shader(osg::Shader::VERTEX);
m_fragment_shader = new osg::Shader(osg::Shader::FRAGMENT);

m_program->addShader(m_vertex_shader);
m_program->addShader(m_fragment_shader);

/*
 * Add attributes (GLSL attributes, not OSG Attributes).
 *
 * I'm not yet sure about how to properly choose explicit indices, so
 * I'm going to use the ATTRIBUTE_6 and ATTRIBUTE_7 values
 * from osg::Drawable.
 *
 * From osg_gls_l_july2005.pdf:
 *
 * - GL supports both explicit and automatic attribute binding
 * - GLSL will dynamically assign attribute indices if not otherwise
 *   specified
 * - However, OSG currently supports only explicit binding, so app
 *   must assign indices
 * - Automatic binding makes display lists dependent on osg::Program,
 *   and has impact on DL sharing
 * - GLSL specifies much freedom in selecting attribute indices, but
 *   some current drivers impose restrictions
 */
m_program->addBindAttribLocation("attr_weight_array", osg::Drawable::ATTRIBUTE_6);
m_program->addBindAttribLocation("attr_index_array", osg::Drawable::ATTRIBUTE_7);
m_program->addBindAttribLocation("attr_tangent", osg::Drawable::SECONDARY_COLORS);
}

```

Destructor

Description:

Dereference OSG objects created in the constructor.

```

diguyOsgGraphicsShaderProgram::~diguyOsgGraphicsShaderProgram()
{
    m_vertex_shader_filename[0] = NULL;
    m_fragment_shader_filename[0] = NULL;

    m_program = NULL;
    m_vertex_shader = NULL;
    m_fragment_shader = NULL;
}

```

load_vertex_shader_source

*Build Stage***Description:**

Override of

`diguyGraphicsShaderProgram::load_vertex_shader_source()`.

The `load_vertex_shader_source()` function should load the source code of the shader's vertex shader part. The source code should be linked in the `link()` function instead of here, as usually the pixel shader source needs to be loaded too before the `link()` call.

This implementation uses OSG to load the shader source code.

```
int
diguyOsgGraphicsShaderProgram::load_vertex_shader_source(const char* filename_base)
{
    /*
     * Use the function
     *   diguyGraphicsShaderProgram::derive_diguy_shader_source_filename()
     * to derive default DI-Guy shader filenames.
     */
    derive_diguy_shader_source_filename(m_vertex_shader_filename,
        511,
        filename_base,
        "glsl",
        "vert",
        1);

    bool retval = m_vertex_shader->loadShaderSourceFromFile(m_vertex_shader_filename);

    /*
     * Prepend #define OSG_INTERSECTION to the shader to allow
     * the intersection shader to work with OSG.
     *
     * Prepend #version at the top to comply with
     * ATI shader strictness.
     */
    const std::string my_version = "#version 120\n";
    const std::string my_defines = "#define OSG_INTERSECTION\n";
    const std::string my_line_reset = "#line 0 0\n";
    const std::string&shader_source = m_vertex_shader->getShaderSource();

    std::string new_source = my_version + my_defines + my_line_reset + shader_source;

    m_vertex_shader->setShaderSource(new_source);

    if (retval == true)
        return 0;

    return -1;
}
```

load_pixel_shader_source*Build Stage*

Description:

Override of `diguyGraphicsShaderProgram::load_pixel_shader_source()`.

The `load_pixel_shader_source()` function should load the source code of the shader's pixel shader part. The source code should be linked in the `link()` function instead of here, as usually the vertex shader source needs to be loaded too before the `link()` call.

This implementation uses OSG to load the shader source code.

```
int
diguyOsgGraphicsShaderProgram::load_pixel_shader_source(const char* filename_base)
{
    /*
     * Use the function
     * diguyGraphicsShaderProgram::derive_diguy_shader_source_filename()
     * to derive default DI-Guy shader filenames.
     */
    derive_diguy_shader_source_filename(m_fragment_shader_filename,
        511,
        filename_base,
        "glsl",
        "frag",
        1);

    /*
     * Because sRGB textures aren't currently supported in OSG, we need to
     * disable them in the shader. To do so we're using a #define in the
     * shader source. Below is a modified version of the
     * osg::Shader::loadShaderSourceFromFile() function's internal code
     * that adds the #define where we want it.
     */

    /*
     * Old way:
     */
    //bool retval = m_fragment_shader->loadShaderSourceFromFile(m_fragment_shader_filename);

    std::ifstream source_file;

    source_file.open(m_fragment_shader_filename, std::ios::binary);
    if (!source_file)
    {
        bdi_log_printf(BDI_LOG_ERROR,
            "ERROR: Can't open shader source file '%s'.\n",
            m_fragment_shader_filename);
        return -1;
    }

    source_file.seekg(0, std::ios::end);
    int length = source_file.tellg();
    char* text = new char[length + 1];
    source_file.seekg(0, std::ios::beg);
    source_file.read(text, length);
    source_file.close();
    text[length] = '\0';

    std::string shader_source = text;
    std::string version = "#version 120\n";
}
```



```

// OSG Doesn't support SRGB texture targets at the moment so make the pow
// gamma no op.
std::string my_defines;
if(diguyApp::get_app()->get_use_sRGB_textures()){
    my_defines = "#define GAMMA_FACTOR 2.2 \n";
}
else{
    my_defines = "#define GAMMA_FACTOR 1.0 \n";
}

// No support for point lights in osg samples yet
//my_defines += "#define POINT_LIGHT_SUPPORT\n";

// No support for shadow maps in osg samples yet
//my_defines += "#define SHADOW_SUPPORT\n";

my_defines += "#define COMPONENT_MAP_SUPPORT\n";
// reset line numbers for error reporting
my_defines += "#line 0 0\n";

// catenate defines and source together
shader_source = my_defines + text;

m_fragment_shader->setShaderSource(shader_source.c_str());
delete[] text;

return 0;
}

```

link

Build Stage

Description:

Override of `diguyGraphicsShaderProgram::link()`.

```

int
diguyOsgGraphicsShaderProgram::link()
{
    // Options for the uniform array name.
    static char* ufrm = "ufrm_link_matrices";
    static char* ufrm0 = "ufrm_link_matrices[0]";

    static char* ufrm_color = "ufrm_link_colors";
    static char* ufrm0_color = "ufrm_link_colors[0]";

    /*
     * Work done internally by OSG.
     *
     * Work around for uniform array bug in OSG. Compile and link the
     * program to determine if the the graphics driver will return
     * "ufrm_link_matrices" or "ufrm_link_matrices[0]" for the array of
     * uniforms in the shader. Give the string to the diguyOsgGraphicsShape
     * so it can properly set and use the uniform name.
     */
    if (!diguyOsgGraphicsShape::s_ufrm_link_matrices_string)
    {
        static bool warn_once = true;
    }
}

```

```

osgViewer::Viewer::Contexts contexts;
osgViewer::Viewer* viewer = diguyOsgGraphicsLink::get_viewer();
if (!viewer)
{
    if (warn_once)
    {
        warn_once = false;
        bdi_log_printf(BDI_LOG_WARN,
            "WARNING: Can't detect shader uniform array name "
            "(diguyGraphicsLink::s_viewer is NULL). Set viewer using "
            "diguyOsgGraphicsLink::set_viewer().\n");

        // Just guess at one...
        diguyOsgGraphicsShape::s_ufrm_link_matrices_string = ufrm0;
    }

    return 0;
}

viewer->getContexts(contexts);

m_program->compileGLObjects(*(contexts[0]->getState()));

osg::Program::PerContextProgram* pcp_program = m_program->getPCP(contexts[0]->getS

#if OPENSCENEGAPH_MAJOR_VERSION == 2
    pcp_program->linkProgram();
#else
    pcp_program->linkProgram(*(contexts[0]->getState()));
#endif

    if (!diguyOsgGraphicsShape::s_ufrm_link_matrices_string)
    {
        if (pcp_program->getUniformLocation("ufrm_link_matrices") != -1)
        {
            diguyOsgGraphicsShape::s_ufrm_link_matrices_string = ufrm;
        }
        else
        {
            diguyOsgGraphicsShape::s_ufrm_link_matrices_string = ufrm0;
        }
    }
}

if (!diguyOsgGraphicsShape::s_ufrm_link_colors_string)
{
    static bool warn_once = true;
    osgViewer::Viewer::Contexts contexts;
    osgViewer::Viewer* viewer = diguyOsgGraphicsLink::get_viewer();
    if (!viewer)
    {
        if (warn_once)
        {
            warn_once = false;
            bdi_log_printf(BDI_LOG_WARN,
                "WARNING: Can't detect shader uniform array name "
                "(diguyGraphicsLink::s_viewer is NULL). Set viewer using "
                "diguyOsgGraphicsLink::set_viewer().\n");

            // Just guess at one...
            diguyOsgGraphicsShape::s_ufrm_link_colors_string = ufrm0;
        }
    }
}

```

```

    }

    return 0;
}

viewer->getContexts(contexts);

m_program->compileGLObjects(*(contexts[0]->getState()));

osg::Program::PerContextProgram* pcp_program = m_program->getPCP(contexts[0]->getState());

#if OPENSCENEGGRAPH_MAJOR_VERSION == 2
    pcp_program->linkProgram();
#else
    pcp_program->linkProgram(*(contexts[0]->getState()));
#endif

    if (!diguyOsgGraphicsShape::s_ufrm_link_colors_string)
    {
        if (pcp_program->getUniformLocation("ufrm_link_colors") != -1)
        {
            diguyOsgGraphicsShape::s_ufrm_link_colors_string = ufrm_color;
        }
        else if (pcp_program->getUniformLocation("ufrm_link_colors[0]") != -1)
        {
            diguyOsgGraphicsShape::s_ufrm_link_colors_string = ufrm0_color;
        }
    }

    return 0;
}

```

reload

Description:

Override of `diguyGraphicsShaderProgram::reload()`.

The `reload()` function should reload previously loaded shader source code, and relink the program. It can be called during any stage, but is usually called during the Update Stage.

```

int
diguyOsgGraphicsShaderProgram::reload()
{
    /*
     * The shader will internally be set to out-of-date, and will
     * automatically be relinked when needed.
     */
    m_vertex_shader->loadShaderSourceFromFile(m_vertex_shader_filename);
    m_fragment_shader->loadShaderSourceFromFile(m_fragment_shader_filename);

    return 0;
}

```

Create function definition

Create and return an object of a subclass of [diguyGraphicsShaderProgram](#). This function will be called whenever DI-Guy requires a shader program object.

```
diguyGraphicsShaderProgram*
diguyOsgGraphicsShaderProgram_create_func(void* internal_data)
{
    return new diguyOsgGraphicsShaderProgram(internal_data);
}
```

5.67 DI-Guy Graphics API, diguyOsgGraphicsShape

diguyOsgGraphicsShape.cpp

DI-Guy SDK Version 12.5.2

This file was automatically generated from diguyOsgGraphicsShape.cpp. Do not edit this file directly; the changes will be lost.

Introduction

Shapes are the high-level containers of visible geometry. They are attached to links, and contain zero or more mesh objects.

The class **diguyOsgGraphicsShape** is an example of a [diguyGraphicsShape](#) subclass for a scene graph environment using the DI-Guy Loader. The example subclass shows how to use the accessors of the [diguyGraphicsShape](#) class and which virtual functions should be overridden for a scene graph environment such as OSG.

Header files and forward declarations

```
#include "diguyOsgGraphicsShape.h"
#include "diguyOsgGraphicsLink.h"
#include "diguyOsgGraphicsMesh.h"

#include <osg/Drawable>
#include <osg/StateSet>
#include <osg/TexEnv>
#include <osg/Texture2D>

#include <libbdiolog.h>
#include <diguyGraphicsTexture.h>
#include <diguyCharacter.h>
```

```

#include <diguyScenario.h>
#include <cstring>
#include <assert.h>

/*
 * Top-level static variable that determines whether LOD calculations are
 * based strictly on distance of object from camera, or on size of object,
 * in pixels, in frame.
 */
bool diguyOsgGraphicsShape::s_use_pixel_lods = false;

/*
 * String to use for the uniform array for bones.
 * The value of the string ("ufrm_link_matrices" or "ufrm_link_matrices[0]")
 * is determined in the diguyOsgGraphicsShaderProgram to work around
 * a bug with name of uniform arrays in OSG.
 */
char* diguyOsgGraphicsShape::s_ufrm_link_matrices_string = NULL;
char* diguyOsgGraphicsShape::s_ufrm_link_colors_string = NULL;

```

Constructor

Description:

Initialize data members of the class to a known state.

Base class data will not be fully initialized until the Build Stage is done, so most base class accessors should **not** be called here.

```

diguyOsgGraphicsShape::diguyOsgGraphicsShape(void* internal_data)
:
    diguyGraphicsShape(internal_data),
    m_switch_node(NULL),
    m_lod_node(NULL),
    m_manual_lod_switch_node(NULL),
    m_ufrm_link_matrices(NULL),
    m_ufrm_link_colors(NULL)
{
}

```

build

Build Stage

Description:

Override of `diguyGraphicsShape::build()`.

The `build()` function should create the necessary objects that will apply offset data of this shape from its parent link during scene graph traversal. It may also need to create objects for level-of-detail switching.

This implementation creates OSG nodes that will serve as attachment points between the meshes that contain visible geometry and the links to which they will be attached.

```

void
diguyOsgGraphicsShape::build()
{
    int lod;

    diguyGraphicsLink* diguy_link = get_link();
    diguyOsgGraphicsLink* osg_link = (diguyOsgGraphicsLink *) diguy_link;

    osg::MatrixTransform* link_dof_node = osg_link->get_dof_node();

    if (!link_dof_node)
        return;

    /*
     * By default we'll attach meshes to the link's dof node. If this
     * shape is switched, however (e.g. for switchable hands), we'll create
     * a switch node and attach them to the switch.
     */
    osg::ref_ptr<osg::Group> lod_attach_node = link_dof_node;

    /*
     * Check if there's a per shape variation texture to bind. This texture
     * modifies the color of various pieces of the character.
     */
    unsigned char * variation_texture = get_variation_texture();
    if (variation_texture)
    {
        /*
         * Add a new node to the hierarchy that is the child of the link
         * and holds the unique ramp texture for this mesh.
         */
        osg::ref_ptr<osg::Group> shape_node = new osg::Group();
        lod_attach_node->addChild(shape_node);

        /*
         * Change the LOD attach node to this new group so that having the texture
         * propagates down the render graph.
         */
        lod_attach_node = shape_node;

        int width = get_variation_texture_size();
        int height = 1;
        GLenum pixel_format = GL_RGBA;

        GLint internal_texture_format = GL_RGBA;

        /*
         * Make a copy of the texture map data. The osg image will take
         * ownership of it.
         */

        unsigned char* texture_map_data_copy = new unsigned char [width*4];
        memcpy(texture_map_data_copy, variation_texture, sizeof(unsigned char)*width*4);

        /*
         * Create an osg image.
         */
        osg::ref_ptr<osg::Image> osg_image = new osg::Image();

        osg_image->setImage(width,
            height,

```

```

        1,
        internal_texture_format,
        pixel_format,
        GL_UNSIGNED_BYTE,
        texture_map_data_copy,
        osg::Image::USE_NEW_DELETE, // osg will deallocate texture_map_data_copy wi
        4); // pack/unpack alignment

    /*
     * Create a texture object, and load it with the osg::Image created
     * above.
     */
    osg::ref_ptr<osg::Texture2D> m_texture;
    m_texture = new osg::Texture2D();

    // not sure this is needed
    m_texture->setUnRefImageDataAfterApply(true);

    m_texture->setWrap(osg::Texture2D::WRAP_S, osg::Texture2D::CLAMP);
    m_texture->setWrap(osg::Texture2D::WRAP_T, osg::Texture2D::CLAMP);

    m_texture->setFilter(osg::Texture2D::MAG_FILTER, osg::Texture2D::NEAREST);
    m_texture->setFilter(osg::Texture2D::MIN_FILTER, osg::Texture2D::NEAREST);

    m_texture->setResizeNonPowerOfTwoHint(true);
    m_texture->setImage(osg_image);

    /*
     * Setup the texture unit and turn the texture on.
     */
    osg::ref_ptr<osg::StateSet> texture_state_set = new osg::StateSet;

    texture_state_set->setTextureAttribute(DIGUY_TEXTURE_MAP_TYPE_COMPONENT_RAMP, m_t

    texture_state_set->setTextureMode(DIGUY_TEXTURE_MAP_TYPE_COMPONENT_RAMP,
        GL_TEXTURE_2D,
        osg::StateAttribute::PROTECTED | osg::StateAttribute::ON);
    lod_attach_node->setStateSet(texture_state_set);
}

/*
 * See if we should create a switch.
 */
if (get_switch_var_name() != NULL)
{
    m_switch_node = new osg::Switch();
    m_switch_node->setNewChildDefaultValue(true);
    link_dof_node->addChild(m_switch_node);

    lod_attach_node = m_switch_node;
}

/*
 * Determine whether automatic graphics LOD switching is enabled. This
 * will determine which kind of node we create for LOD switching.
 */
int automatic_graphics_lod_switching = 1;
diguyCharacter* character = get_character();
if (character)
{
    diguyScenario* scenario = character->get_scenario();

```

```

        automatic_graphics_lod_switching = scenario->get_automatic_graphics_lod_switching
    }

    /*
     * If automatic switching is enabled, create an osg::LOD() node.
     */
    if (automatic_graphics_lod_switching)
    {
        /*
         * Determine the lod_ranges.
         */
        float* lod_ranges = get_lod_ranges();

        /*
         * Create an lod node.
         */
        m_lod_node = new osg::LOD();
        m_lod_node->setCenter(osg::Vec3f(0.0f, 0.0f, 0.0f));

        if (s_use_pixel_lods)
            m_lod_node->setRangeMode(osg::LOD::PIXEL_SIZE_ON_SCREEN);
        else
            m_lod_node->setRangeMode(osg::LOD::DISTANCE_FROM_EYE_POINT);

        lod_attach_node->addChild(m_lod_node);

        /*
         * Create a geode for each lod. Attach mesh drawables to geodes.
         */
        for (lod=1; lod<=get_num_lods(); lod++)
        {
            osg::Geode* geode = new osg::Geode();

            if (s_use_pixel_lods)
            {
                /*
                 * Because DI-Guy LOD ranges must increase at each index,
                 * but unlike with distances the earlier numbers represent
                 * low LODs, we have to reverse the order of lod ranges.
                 *
                 * LOD 1: lod_ranges[6] to lod_ranges[7]
                 * LOD 2: lod_ranges[5] to lod_ranges[6]
                 * etc.
                 */
                m_lod_node->addChild(geode, lod_ranges[7 - lod], lod_ranges[8 - lod]);
            }
            else
            {
                /*
                 * LOD 1: lod_ranges[0] to lod_ranges[1]
                 * LOD 2: lod_ranges[1] to lod_ranges[2]
                 * etc.
                 */
                m_lod_node->addChild(geode, lod_ranges[lod-1], lod_ranges[lod]);
            }

            int num_meshes = get_num_meshes(lod);
            int mesh_index;

            for (mesh_index=0; mesh_index<num_meshes; mesh_index++)
            {

```



```

        diguyGraphicsMesh* diguy_mesh;
        diguyOsgGraphicsMesh* diguy_osg_mesh;

        diguy_mesh = get_mesh_at_index_for_lod(mesh_index, lod);
        diguy_osg_mesh = (diguyOsgGraphicsMesh*) diguy_mesh;

        if (diguy_osg_mesh)
        {
            osg::Drawable* drawable = (osg::Drawable*) diguy_osg_mesh->get_drawable();
            geode->addDrawable(drawable);
        }
    }
}

/*
 * Create a geode for each lod. Attach mesh drawables to geodes.
 */
else
{
    m_manual_lod_switch_node = new osg::Switch();
    m_manual_lod_switch_node->setNewChildDefaultValue(false);

    lod_attach_node->addChild(m_manual_lod_switch_node);

    /*
     * Create a geode for each lod. Attach mesh drawables to geodes.
     */
    for (lod=1; lod<=get_num_lods(); lod++)
    {
        osg::Geode* geode = new osg::Geode();

        m_manual_lod_switch_node->addChild(geode);

        int num_meshes = get_num_meshes(lod);
        int mesh_index;

        for (mesh_index=0; mesh_index<num_meshes; mesh_index++)
        {
            diguyGraphicsMesh* diguy_mesh;
            diguyOsgGraphicsMesh* diguy_osg_mesh;

            diguy_mesh = get_mesh_at_index_for_lod(mesh_index, lod);
            diguy_osg_mesh = (diguyOsgGraphicsMesh*) diguy_mesh;

            if (diguy_osg_mesh)
            {
                /*
                 * Some meshes can have multiple graphics states
                 * associated with them (e.g., everything in the mesh
                 * is that same except that it may have multiple texture
                 * options).
                 *
                 * Determine which graphics state we should use for
                 * meshes attached to this shape.
                 */
                int graphics_state_switch_index = get_graphics_state_switch_index();
                osg::Drawable* drawable = (osg::Drawable*) diguy_osg_mesh->get_drawable();
                geode->addDrawable(drawable);
            }
        }
    }
}

```

```

        m_manual_lod_switch_node->setSingleChildOn(get_graphics_lod() - 1);
    }
}
}

```

unbuild

Unbuild Stage

Description:

Override of `diguyGraphicsShape::unbuild()`.

The `unbuild()` function should destroy the objects created in the `build()` function.

Deref the nodes created in `build()`.

```

void
diguyOsgGraphicsShape::unbuild()
{
    const char* name = get_name();

    diguyGraphicsLink* diguy_link = get_link();
    diguyOsgGraphicsLink* osg_link = (diguyOsgGraphicsLink *) diguy_link;

    osg::MatrixTransform* link_dof_node = osg_link->get_dof_node();

    if (!link_dof_node)
        return;

    osg::Group* lod_detach_node = link_dof_node;

    /*
     * Remove all of the group nodes we created and added in the build()
     * call; this should reduce the ref count of the added nodes by 1, to
     * 1. Then set the pointers to NULL; this should reduce the ref count
     * to 0, resulting in the group nodes being deleted.
     *
     * Setting the m_switch_node ref_ptr to NULL dereferences the Switch
     * object it points to, possibly deleting it. Ditto for m_lod_node and
     * m_manual_lod_switch_node.
     */
    if (m_switch_node != NULL)
        lod_detach_node = m_switch_node;

    if (m_lod_node != NULL)
    {
        lod_detach_node->removeChild(m_lod_node);
        m_lod_node = NULL;
    }
    else if (m_manual_lod_switch_node != NULL)
    {
        lod_detach_node->removeChild(m_manual_lod_switch_node);
        m_manual_lod_switch_node = NULL;
    }
}

```

```

    }

    if (m_switch_node != NULL)
    {
        link_dof_node->removeChild(m_switch_node);
        m_switch_node = NULL;
    }
}

```

show

Update Stage

Description:

Override of `diguyGraphicsShape::show()`.

Change switch node state to show attached children.

```

void
diguyOsgGraphicsShape::show()
{
    if (m_switch_node)
        m_switch_node->setAllChildrenOn();
}

```

hide

Update Stage

Description:

Override of `diguyGraphicsShape::hide()`.

Change switch node state to hide attached children.

```

void
diguyOsgGraphicsShape::hide()
{
    if (m_switch_node)
        m_switch_node->setAllChildrenOff();
}

```

update

Update Stage

Description:

Override of `diguyGraphicsShape::update()`.

Send skinned uniforms to shader.

```

void
diguyOsgGraphicsShape::update()
{
    int num_matrices = 0;
    const float* matrix_data = NULL;

    get_shader_matrix_data(

    osg::StateSet* shader_state_set = NULL;

    if ((num_matrices > 0) &matrix_data)
    {
        /*
         * Ideally this would happen at the build stage, but we don't know
         * the num of matrices yet at that point and we want to avoid
         * allocating the max num every time.
         */
        if (!m_ufrm_link_matrices &s_ufrm_link_matrices_string)
        {
            shader_state_set = new osg::StateSet();

            /*
             * Adds a per shape set of uniforms that feed skinned data to the
             * underlying shader.
             */
            m_ufrm_link_matrices = new osg::Uniform(osg::Uniform::FLOAT_MAT4,
                s_ufrm_link_matrices_string,
                num_matrices);

            shader_state_set->addUniform(m_ufrm_link_matrices);

            /*
             * Propagate state change to children.
             */
            if (m_lod_node)
                m_lod_node->setStateSet(shader_state_set);

            if (m_manual_lod_switch_node)
                m_manual_lod_switch_node->setStateSet(shader_state_set);

            if (m_switch_node)
                m_switch_node->setStateSet(shader_state_set);
        }

        /*
         * Update the matrices.
         */
        if (m_ufrm_link_matrices != NULL)
        {
            int i;
            for (i=0; i<num_matrices; i++)
            {
                m_ufrm_link_matrices->setElement(i, osg::Matrixf(matrix_data));
                matrix_data += 16;    // 16 floats per matrix
            }
        }
    }

    int num_colors = 1;
    const float* color_data = NULL;

```

```

const int isSkinned = get_contains_skinned_file();
if (isSkinned)
{
    if (get_bind_color_array(== -1)
    {
        return;
    }
}

if (!m_ufrm_link_colors &s_ufrm_link_colors_string)
{
    /*
    * Adds a per shape set of uniforms that feed skinned data to the
    * underlying shader.
    */
    if (!shader_state_set)
    {
        shader_state_set = new osg::StateSet();
    }

    m_ufrm_link_colors = new osg::Uniform(osg::Uniform::FLOAT_VEC4,
        s_ufrm_link_colors_string,
        num_colors);

    shader_state_set->addUniform(m_ufrm_link_colors);

    /*
    * Propagate state change to children.
    */
    if (m_lod_node)
        m_lod_node->setStateSet(shader_state_set);

    if (m_manual_lod_switch_node)
        m_manual_lod_switch_node->setStateSet(shader_state_set);

    if (m_switch_node)
        m_switch_node->setStateSet(shader_state_set);
}

if (m_ufrm_link_colors)
{
    if (isSkinned)
    {
        for (int i = 0; i < num_colors; ++i)
        {
            m_ufrm_link_colors->setElement(i,
                osg::Vec4(color_data[0], color_data[1], color_data[2], color_data[3]),
                color_data += 4;
            }
        }
    }
    else
    {
        unsigned char r, g, b;

        diguyGraphicsLink* link = get_link();
        link->get_unique_color(
            m_ufrm_link_colors->setElement(0, osg::Vec4(r/255.0f, g/255.0f, b/255.0f, 1))
        )
    }
}
}

```

set_graphics_lod_ranges**Description:**

Override of `diguyGraphicsShape::set_graphics_lod_ranges()`, called as needed.

The `set_graphics_lod_ranges()` function should update the ranges of lod objects created in the `build()` function with the passed ranges.

In this implementation the ranges of the `osg::LOD` created in `build()` are updated. If automatic LOD switching is enabled, the LOD node will not exist and this function will have no effect.

```
void
diguyOsgGraphicsShape::set_graphics_lod_ranges(float* lod_ranges)
{
    if (m_lod_node != NULL)
    {
        int lod;
        for (lod=1; lod<=get_num_lods(); lod++)
        {
            m_lod_node->setRange(lod-1, lod_ranges[lod-1], lod_ranges[lod]);
        }
    }
}
```

set_graphics_lod**Description:**

Override of `diguyGraphicsShape::set_graphics_lod()`, called as needed.

The `set_graphics_lod()` function should select and show the object representing the passed LOD.

This implementation manually sets the `osg::Switch` created in `build()` to show the correct geode. If automatic LOD switching is enabled, the switch node will not exist and this function will have no effect.

```
void
diguyOsgGraphicsShape::set_graphics_lod(int lod)
{
    if (m_manual_lod_switch_node != NULL)
    {
        m_manual_lod_switch_node->setSingleChildOn(lod-1);
    }
}
```

Create function definition

Create and return an object of a subclass of `diguyGraphicsShape`. This function will be called whenever DI-Guy requires a shape object.

```
diguyGraphicsShape*
diguyOsgGraphicsShape_create_func(void* internal_data)
{
    return new diguyOsgGraphicsShape(internal_data);
}
```

5.68 DI-Guy Graphics API, diguyOsgGraphicsState

diguyOsgGraphicsState.cpp

DI-Guy SDK Version 12.5.2

This file was automatically generated from diguyOsgGraphicsState.cpp. Do not edit this file directly; the changes will be lost.

Graphics state objects mostly serve as containers for other sub-objects, such as materials and textures, that control the visual state of geometry. At times renderer-specific objects need to be created in this level, as well.

The class **diguyOsgGraphicsState** is an example of a [diguyGraphicsState](#) subclass for a scene graph renderer using the DI-Guy Loader. The example subclass shows how to use the accessors of the [diguyGraphicsState](#) class and which virtual functions should be overridden for a scene graph renderer such as OSG.

Header files and forward declarations

```
#include "diguyOsgGraphicsState.h"
#include "diguyOsgGraphicsMaterial.h"
#include "diguyOsgGraphicsTexture.h"

#include <libbdilog.h>
```

Constructor

Description:

This constructor initializes data members of the class to a known state.

Most base class data will not be fully initialized until during the Build Stage, so most base class accessors should **not** be called here.

```
diguyOsgGraphicsState::diguyOsgGraphicsState(void* internal_data)
:
    diguyGraphicsState(internal_data),
    m_state_set(NULL)
{
}
```

build

Build Stage

Description:

Override of `diguyGraphicsState::build()`.

The `build()` function should create the necessary objects that will combine the state's material and texture data.

This implementation creates an `osg::StateSet` from the data in this `diguyGraphicsState`. This will entail combining the state data from this state's material and texture objects.

```
void
diguyOsgGraphicsState::build()
{
    m_state_set = new osg::StateSet;

    /*
     * Get a pointer to the diguyGraphicsMaterial, and cast it
     * to a diguyOsgGraphicsMaterial.
     */
    diguyGraphicsMaterial* diguy_material = get_material();
    diguyOsgGraphicsMaterial* diguy_osg_material = (diguyOsgGraphicsMaterial*) diguy_material;

    if (diguy_osg_material)
    {
        osg::Material* mtl = diguy_osg_material->get_material();

        if (mtl)
        {
            m_state_set->setAttribute(mtl);

            /*
             * Note: This doesn't work yet for skinned meshes; not sure why.
             */
            int is_unlit = get_is_unlit();

            if (is_unlit)
                m_state_set->setMode(GL_LIGHTING, osg::StateAttribute::PROTECTED | osg::StateAttribute::UNLIT);
            else
                m_state_set->setMode(GL_LIGHTING, osg::StateAttribute::PROTECTED | osg::StateAttribute::DEFAULT);
        }
    }

    /*
     * Get a pointer to the list of diguyGraphicsTexture, and cast it to a
     * diguyOsgGraphicsTexture.
     */
    int i;
    for (i = 0; i < get_num_textures(); i++)
    {
        diguyGraphicsTexture* diguy_texture = get_texture(i);
        diguyOsgGraphicsTexture* diguy_osg_texture = (diguyOsgGraphicsTexture*) diguy_texture;

        if (diguy_osg_texture)
        {
            osg::Texture2D* texture = diguy_osg_texture->get_texture();
            /*
             * Setup the texture unit and turn the texture on.
             */
        }
    }
}
```



```

        osg::StateSet* texture_state_set = new osg::StateSet;

        texture_state_set->setTextureAttribute(get_texture_type(i), diguy_osg_texture);

        texture_state_set->setTextureMode(get_texture_type(i),
            GL_TEXTURE_2D,
            osg::StateAttribute::PROTECTED | osg::StateAttribute::ON);

        if (texture_state_set & texture)
        {
            /*
             * Combine the texture state set with the material state set.
             */
            m_state_set->merge(*texture_state_set);
        }
    }

    int textures_available_mask = get_textures_available();
    /*
     * Set the associated uniform that turns on bump mapping part of the
     * shader.
     */
    osg::Uniform* ufrm_bump_map_enabled = new osg::Uniform("ufrm_bump_map_enabled",
        !!(textures_available_mask & DIGUY_TEXTURE_MAP_MASK_BUMP));
    m_state_set->addUniform(ufrm_bump_map_enabled);

    /*
     * Set the associated uniform that turns on specular mapping part of the
     * shader.
     */
    osg::Uniform* ufrm_spec_map_enabled = new osg::Uniform("ufrm_specular_map_enabled",
        !!(textures_available_mask & DIGUY_TEXTURE_MAP_MASK_SPECULAR));
    m_state_set->addUniform(ufrm_spec_map_enabled);

    /*
     * Set the associated uniform that turns on texture variation mapping part of the
     * shader.
     */
    osg::Uniform* ufrm_comp_map_enabled = new osg::Uniform("ufrm_component_map_enabled",
        !!(textures_available_mask & DIGUY_TEXTURE_MAP_MASK_COMPONENTS));
    m_state_set->addUniform(ufrm_comp_map_enabled);

    /*
     * Set the default rendering bin for objects with this state. States
     * that contain a material or texture with an alpha component, which
     * controls transparency, should be placed in the TRANSPARENT_BIN,
     * those without in the OPAQUE_BIN.
     */
    if (get_contains_alpha_component())
        m_state_set->setRenderingHint(osg::StateSet::TRANSPARENT_BIN);
    else
        m_state_set->setRenderingHint(osg::StateSet::OPAQUE_BIN);
}

```

unbuild

Unbuild Stage

Description:

Override of `diguyGraphicsState::unbuild()`.

The `unbuild()` function should destroy the objects created in the `build()` function.

This implementation dereferences the `osg::StateSet` created in `build()`.

```
void
diguyOsgGraphicsState::unbuild()
{
    /*
     * Setting the m_state_set ref_ptr to NULL dereferences the
     * StateSet object it points to, possibly deleting it.
     */
    m_state_set = NULL;
}
```

Create function definition

Create and return an object of a subclass of `diguyGraphicsState`. This function will be called whenever DI-Guy requires a state object.

```
diguyGraphicsState*
diguyOsgGraphicsState_create_func(void* internal_data)
{
    return new diguyOsgGraphicsState(internal_data);
}
```

5.69 DI-Guy Graphics API, diguyOsgGraphicsTexture**diguyOsgGraphicsTexture.cpp****DI-Guy SDK Version 12.5.2**

This file was automatically generated from `diguyOsgGraphicsTexture.cpp`. Do not edit this file directly; the changes will be lost.

Introduction

Textures modify the visible appearance of meshes. They work in concert with materials to determine the final appearance of visible geometry.

The class **diguyOsgGraphicsTexture** is an example of a **diguyGraphicsTexture** subclass for a scene graph renderer using the DI-Guy Loader. The example subclass shows how to use the accessors of the **diguyGraphicsTexture** class and which virtual functions should be overridden for a scene graph renderer such as OSG.

Header files and forward declarations

```
#include "diguyOsgGraphicsTexture.h"

#include <osg/TexEnv>
#include <osgDB/ReadFile>

#include <libbdiolog.h>
#include <stdlib.h>
#include <string.h>
```

Constructor

This constructor initializes data members of the class to a known state.

Most base class data will not be fully initialized until during the Build Stage, so most base class accessors should **not** be called here.

```
diguyOsgGraphicsTexture::diguyOsgGraphicsTexture(void* internal_data)
:
    diguyGraphicsTexture(internal_data)
{
    /*
     * Nothing else to do after calling base class constructor and
     * initializing data members.
     */
}
```

build

Build Stage

Description:

Override of **diguyGraphicsTexture::build()**.

The **build()** function should create renderer objects that will set the proper texture state in the scene graph traversal.

This implementation creates an **osg::Texture2D** object and a companion **osg::StateSet**. The state set of this texture will be combined with the state set of a material in the

diguyOsgGraphicsState class.

```

void
diguyOsgGraphicsTexture::build()
{
    /*
     * Get the texture map data buffer.
     */
    unsigned char* texture_map_data = get_texture_map_data();

    /*
     * If there is texture data, create a mip-mapped texture object.
     */
    if (texture_map_data)
    {
        int width = get_width();
        int height = get_height();
        GLenum pixel_format;

        int pixel_format_as_int = get_gl_format();
        if (pixel_format_as_int == -1)
        {
            bdi_log_printf(BDI_LOG_WARN, "WARNING: Failed to obtain legal "
                "pixel format for texture '%s'.\n",
                get_name());
            return;
        }

        pixel_format = (GLenum) pixel_format_as_int;

        GLint internal_texture_format = pixel_format;

        /*
         * Make a copy of the texture map data. The osg image will take
         * ownership of it.
         */
        unsigned char* texture_map_data_copy = get_texture_map_data();
        int texture_size = get_texture_map_data_size();

        unsigned char* texture_map_data_copy = new unsigned char [texture_size];
        memcpy(texture_map_data_copy, texture_map_data, texture_size);

        /*
         * Create an osg image.
         */
        osg::Image* osg_image = new osg::Image();

        osg_image->setImage(width,
            height,
            1,
            internal_texture_format,
            pixel_format,
            GL_UNSIGNED_BYTE,
            texture_map_data_copy,
            osg::Image::USE_NEW_DELETE, // osg will deallocate texture_map_data_copy wi
            4); // pack/unpack alignment

        osg_image->setFileName(get_filename());

        /*
         * Mipmaps are generated automatically by OSG for non-compressed

```

```

*   textures.
*
*   Compressed textures might already have them; if so, we have
*   to set the mipmap offsets manually.
*/
int num_included_mipmaps = get_num_included_mipmaps();

if (num_included_mipmaps > 1)
{
    /*
    *   Get mipmap offsets.
    *
    *   The number of mipmap offsets in osg is one less than the
    *   number of mipmaps, as the offset of "mipmap 0" (the
    *   original, base image) is assumed to be 0.
    */
    osg::Image::MipmapDataType mipmaps;
    mipmaps.resize(num_included_mipmaps - 1);

    int k;
    for (k=1; k<num_included_mipmaps; k++)
        mipmaps[k-1] = get_mipmap_data_offset(k);

    osg_image->setMipmapLevels(mipmaps);
}

/*
*   Create a texture object, and load it with the osg::Image created
*   above.
*/
m_texture = new osg::Texture2D();

// not sure this is needed
m_texture->setUnRefImageDataAfterApply(true);

if (get_clamp_s())
    m_texture->setWrap(osg::Texture2D::WRAP_S, osg::Texture2D::CLAMP);
else
    m_texture->setWrap(osg::Texture2D::WRAP_S, osg::Texture2D::REPEAT);

if (get_clamp_t())
    m_texture->setWrap(osg::Texture2D::WRAP_T, osg::Texture2D::CLAMP);
else
    m_texture->setWrap(osg::Texture2D::WRAP_T, osg::Texture2D::REPEAT);

m_texture->setFilter(osg::Texture2D::MAG_FILTER, osg::Texture2D::LINEAR);
m_texture->setFilter(osg::Texture2D::MIN_FILTER, osg::Texture2D::NEAREST_MIPMAP_L);

m_texture->setResizeNonPowerOfTwoHint(true);
m_texture->setImage(osg_image);

/*
*   Create a state set for the texture.
*/
/*
*   THIS MIGHT Not make sense with multi-texturing now and is unused..
*   this state is created now in the diguyOsgState
*/
m_texture_state_set = new osg::StateSet;

m_texture_state_set->setTextureAttribute(0, m_texture);

```

```

        m_texture_state_set->setTextureMode(0,
            GL_TEXTURE_2D,
            osg::StateAttribute::PROTECTED | osg::StateAttribute::ON);
    }
    else
    {
        /*
         * This should probably do more to create a "null" texture
         * that is, say, white.
         */
        m_texture = NULL;
    }

    /*
     * Free memory allocated by the loader.
     */
    free_loader_memory();
}

```

unbuild

Unbuild Stage

Description:

Override of `diguyGraphicsTexture::unbuild()`.

The `unbuild()` function should destroy the objects created in the `build()` function.

This implementation dereference the objects created in `build()`.

```

void
diguyOsgGraphicsTexture::unbuild()
{
    /*
     * Setting the m_texture ref_ptr to NULL dereferences the Texture2D
     * object it points to, possibly deleting it.
     */
    m_texture = NULL;
    m_texture_state_set = NULL;
}

```

Create function definition

Create and return an object of a subclass of `diguyGraphicsTexture`. This function will be called whenever DI-Guy requires a texture object.

```

diguyGraphicsTexture*
diguyOsgGraphicsTexture_create_func(void* internal_data)
{
    return new diguyOsgGraphicsTexture(internal_data);
}

```

diguyOsgUtil.cpp

DI-Guy SDK Version 12.5.2

This file was automatically generated from diguyOsgUtil.cpp. Do not edit this file directly; the changes will be lost.

```

/*
 * Copyright (C) 1992-2011 Boston Dynamics
 * ALL RIGHTS RESERVED.
 *
 * These coded instructions, statements, and computer programs
 * contain unpublished proprietary information of Boston Dynamics
 * and are protected by Copyright Laws of the United States.
 * They may not be used, duplicated, or disclosed in any form, in
 * whole or in part, without the prior written consent from Boston
 * Dynamics.
 *
 * RESTRICTED RIGHTS LEGEND
 * Use, duplication, or disclosure by the government is subject
 * to restrictions as set forth in FAR 52.227.19(c)(2) or
 * subparagraph (c)(1)(ii) of the Rights in Technical Data and
 * Computer Software clause at DFARS 252.227-7013 and/or in
 * similar or successor clauses in the FAR, or the DOD or NASA
 * FAR Supplement, or to subparagraphs (c)(1) and (c)(2) of the
 * Commercial Computer Software--Restricted Rights at 48 CFR
 * 52.227-19, as applicable. Unpublished-rights reserved under
 * the Copyright Laws of the United States.
 * Contractor/Manufacturer is:
 * Boston Dynamics/78 Fourth Avenue/Waltham MA 02451.
 */

#include <stdio.h>

#include <libbdiolog.h>
#include <diguyAuthorInterface.h>
#include <diguy_vector_classes.h>

#include <osg/BlendFunc>
#include <osg/CullFace>
#include <osg/Geode>
#include <osg/Geometry>
#include <osg/LineWidth>
#include <osg/Material>
#include <osg/MatrixTransform>
#include <osg/ShapeDrawable>
#include <osg/StateSet>

#include "diguyOsgUtil.h"

```

```

// static
void
diguyOsgUtil::detach_and_null(osg::ref_ptr<osg::MatrixTransform>&child)
{
    if (child == NULL)
        return;

    while (child->getNumParents() > 0)
        child->getParent(0)->removeChild(child);
}

```

```

    child = NULL;
}

```

```

// static
void
diguyOsgUtil::detach_and_null(osg::ref_ptr<osg::Geode>&child)
{
    if (child == NULL)
        return;

    while (child->getNumParents() > 0)
        child->getParent(0)->removeChild(child);

    child = NULL;
}

```

```

// static
void
diguyOsgUtil::detach_and_null(osg::Geode* geode, osg::ref_ptr<osg::Geometry>&drawable)
{
    if (drawable == NULL)
        return;

    if (geode == NULL)
        return;

    geode->removeDrawable(drawable);
    drawable = NULL;
}

```

```

// static
osg::Geometry*
diguyOsgUtil::create_sphere_solid(float radius,
    int num_slices,
    int num_layers,
    const diguyVec4f &color)
{
    int index;
    int num_vertices = 0;

    /*
     * Generate vertices.
     */
    diguyVec3f* vertices;
    diguyVec3f* normals;
    diguyVec2f* texture_indices;

    int num_indices;
    unsigned int* indices;

    diguyAuthorInterface::generate_sphere_vertices(radius,
        num_slices,
        num_layers,
        diguyVec3f(0.0f),
        diguyVec3f(0.0f),

```



```

    0,

    /*
     * Create a geometry drawable object.
     */
    osg::Geometry* geometry = new osg::Geometry;

    geometry->setDataVariance(osg::Object::STATIC);
    geometry->setUseVertexBufferObjects(true);

    /*
     * Create osg vertices array.
     *
     * Long way for now, adding data vertex by vertex. I'd like to find
     * the OSG way for simply copying a binary block of data containing
     * float vertex data into the vertices array.
     */
    osg::Vec3Array* osg_vertices = new osg::Vec3Array;
    osg_vertices->reserve(num_vertices);

    float* vptr =

    for (index=0; index<num_vertices; index++)
    {
        osg_vertices->push_back(osg::Vec3(*vptr, *(vptr+1), *(vptr+2)));
        vptr += 3;
    }

    geometry->setVertexArray(osg_vertices);

    /*
     * Create osg normals array.
     */
    osg::Vec3Array* osg_normals = new osg::Vec3Array;
    geometry->setNormalArray(osg_normals);
    geometry->setNormalBinding(osg::Geometry::BIND_PER_VERTEX);

    float* nptr =

    osg_normals->reserve(num_vertices);

    for (index=0; index<num_vertices; index++)
    {
        osg_normals->push_back(osg::Vec3(*nptr, *(nptr+1), *(nptr+2)));
        nptr += 3;
    }

    /*
     * Create osg texture indices array.
     */
    // not done yet

    /*
     * Triangle vertex indices.
     */
    osg::DrawElementsUInt* index_set = new osg::DrawElementsUInt(osg::PrimitiveSet::TRIANG
    index_set->reserve(num_indices);

```

```

geometry->addPrimitiveSet(index_set);

int i;
for (i=0; i<num_indices; i++)
    index_set->push_back(indices[i]);

diguyAuthorInterface::delete_generated_vertices(vertices,
    normals,
    texture_indices,
    indices);

/*
 * State set.
 */
osg::StateSet* state_set = create_material_state_set(color, true);
geometry->setStateSet(state_set);

return geometry;
}

```

```

// static
osg::Geometry*
diguyOsgUtil::create_sphere_wireframe(float radius,
    int num_slices,
    int num_layers,
    const diguyVec4f &color,
    float line_width,
    bool lit)
{
    int index;
    int num_vertices = 0;

    /*
     * Generate vertices.
     */
    diguyVec3f* vertices;
    diguyVec3f* normals;
    diguyVec2f* texture_indices;

    int num_indices;
    unsigned int* indices;

    diguyAuthorInterface::generate_sphere_vertices(radius,
        num_slices,
        num_layers,
        diguyVec3f(0.0f),
        diguyVec3f(0.0f),

        1,

        /*
         * Create a geometry drawable object.
         */
        osg::Geometry* geometry = new osg::Geometry;

```

```

geometry->setDataVariance(osg::Object::STATIC);

/*
 * Create osg vertices array.
 *
 * Long way for now, adding data vertex by vertex. I'd like to find
 * the OSG way for simply copying a binary block of data containing
 * float vertex data into the vertices array.
 */
osg::Vec3Array* osg_vertices = new osg::Vec3Array(num_indices);
for (index=0; index<num_indices; index++)
{
    int vertex_index = indices[index];

    float* vptr = _index].x);
    (*osg_vertices)[index].set(vptr[0], vptr[1], vptr[2]);
}
geometry->setVertexArray(osg_vertices);

if (lit)
{
    osg::Vec3Array* osg_normals = new osg::Vec3Array(num_indices);
    for (index=0; index<num_indices; index++)
    {
        int vertex_index = indices[index];

        float* nptr = index].x);
        (*osg_normals)[index].set(nptr[0], nptr[1], nptr[2]);
    }
    geometry->setNormalArray(osg_normals);
    geometry->setNormalBinding(osg::Geometry::BIND_PER_VERTEX);
}

geometry->addPrimitiveSet(new osg::DrawArrays(osg::PrimitiveSet::LINES, 0, num_indices);

diguyAuthorInterface::delete_generated_vertices(vertices,
    normals,
    texture_indices,
    indices);

/*
 * State set.
 */
osg::StateSet* state_set = create_material_state_set(color, false);

osg::LineWidth* osg_line_width = new osg::LineWidth;
osg_line_width->setWidth(line_width);

state_set->setAttributeAndModes(osg_line_width, osg::StateAttribute::ON);
state_set->setMode(GL_BLEND, osg::StateAttribute::OFF);
state_set->setTextureMode(0, GL_TEXTURE_1D, osg::StateAttribute::OFF);
state_set->setTextureMode(0, GL_TEXTURE_2D, osg::StateAttribute::OFF);
if (!lit)
    state_set->setMode(GL_LIGHTING, osg::StateAttribute::OVERRIDE|osg::StateAttribute

geometry->setStateSet(state_set);

return geometry;
}

```

```

// static
osg::Geometry*
diguyOsgUtil::create_cone_solid(float radius,
    float height,
    int num_slices,
    int num_layers,
    const diguyVec3f &position_offset,
    const diguyVec3f &orientation,
    const diguyVec4f &color)
{
    int index;
    int num_vertices = 0;

    /*
     * Generate vertices.
     */
    diguyVec3f* vertices;
    diguyVec3f* normals;
    diguyVec2f* texture_indices;

    int num_indices;
    unsigned int* indices;

    diguyAuthorInterface::generate_cone_vertices(radius,
        height,
        num_slices,
        num_layers,
        position_offset,
        orientation,

        0,

        /*
         * Create a geometry drawable object.
         */
        osg::Geometry* geometry = new osg::Geometry;

        geometry->setDataVariance(osg::Object::STATIC);
        geometry->setUseVertexBufferObjects(true);

        /*
         * Create osg vertices array.
         */
        osg::Vec3Array* osg_vertices = new osg::Vec3Array;
        osg_vertices->reserve(num_vertices);

        osg::Vec3Array* osg_normals = new osg::Vec3Array;
        osg_normals->reserve(num_vertices);

        float* vptr =
        float* nptr =

        for (index=0; index<num_vertices; index++)
        {
            osg_vertices->push_back(osg::Vec3(*vptr, *(vptr+1), *(vptr+2)));
            vptr += 3;
        }
    }
}

```

```

        osg_normals->push_back(osg::Vec3(*nptr, *(nptr+1), *(nptr+2)));
        nptr += 3;
    }

    geometry->setVertexArray(osg_vertices);
    geometry->setNormalArray(osg_normals);
    geometry->setNormalBinding(osg::Geometry::BIND_PER_VERTEX);

    /*
     * Create osg texture indices array.
     */
    // not done yet

    /*
     * Triangle vertex indices.
     */
    osg::DrawElementsUInt* index_set = new osg::DrawElementsUInt(osg::PrimitiveSet::TRIANGULAR_FAN);
    index_set->reserve(num_indices);
    geometry->addPrimitiveSet(index_set);

    int i;
    for (i=0; i<num_indices; i++)
        index_set->push_back(indices[i]);

    diguyAuthorInterface::delete_generated_vertices(vertices,
        normals,
        texture_indices,
        indices);

    /*
     * State set.
     */
    osg::StateSet* state_set = create_material_state_set(color, true);
    geometry->setStateSet(state_set);

    return geometry;
}

```

```

// static
osg::Geometry*
diguyOsgUtil::create_cone_wireframe(float radius,
    float height,
    int num_slices,
    int num_layers,
    const diguyVec3f &position_offset,
    const diguyVec3f &orientation,
    const diguyVec4f &color,
    float line_width,
    bool lit)
{
    int index;
    int num_vertices = 0;

    /*
     * Generate vertices.
     */
    diguyVec3f* vertices;
    diguyVec3f* normals;
    diguyVec2f* texture_indices;

```

```

int num_indices;
unsigned int* indices;

diguyAuthorInterface::generate_cone_vertices(radius,
    height,
    num_slices,
    num_layers,
    position_offset,
    orientation,

    1,

    /*
     * Create a geometry drawable object.
     */
    osg::Geometry* geometry = new osg::Geometry;

    geometry->setDataVariance(osg::Object::STATIC);
    geometry->setUseVertexBufferObjects(true);

    /*
     * Create osg vertices array.
     */
    osg::Vec3Array* osg_vertices = new osg::Vec3Array(num_indices);
    for (index=0; index<num_indices; index++)
    {
        int vertex_index = indices[index];

        float* vptr = _index].x);
        (*osg_vertices)[index].set(vptr[0], vptr[1], vptr[2]);
    }
    geometry->setVertexArray(osg_vertices);

    if (lit)
    {
        osg::Vec3Array* osg_normals = new osg::Vec3Array(num_indices);
        for (index=0; index<num_indices; index++)
        {
            int vertex_index = indices[index];

            float* nptr = index].x);
            (*osg_normals)[index].set(nptr[0], nptr[1], nptr[2]);
        }
        geometry->setNormalArray(osg_normals);
        geometry->setNormalBinding(osg::Geometry::BIND_PER_VERTEX);
    }

    geometry->addPrimitiveSet(new osg::DrawArrays(osg::PrimitiveSet::LINES, 0, num_indices);

    diguyAuthorInterface::delete_generated_vertices(vertices,
        normals,
        texture_indices,
        indices);

    /*

```

```

    * State set.
    */
    osg::StateSet* state_set = create_material_state_set(color, false);

    osg::LineWidth* osg_line_width = new osg::LineWidth;
    osg_line_width->setWidth(line_width);

    state_set->setAttributeAndModes(osg_line_width, osg::StateAttribute::ON);
    state_set->setMode(GL_BLEND, osg::StateAttribute::OFF);
    state_set->setTextureMode(0, GL_TEXTURE_1D, osg::StateAttribute::OFF);
    state_set->setTextureMode(0, GL_TEXTURE_2D, osg::StateAttribute::OFF);
    if (!lit)
        state_set->setMode(GL_LIGHTING, osg::StateAttribute::OVERRIDE|osg::StateAttribute

    geometry->setStateSet(state_set);

    return geometry;
}

```

```

// static
osg::Geometry*
diguyOsgUtil::create_cylinder_solid(float radius,
    float height,
    int num_slices,
    int num_layers,
    const diguyVec3f &position_offset,
    const diguyVec3f &orientation,
    const diguyVec4f &color)
{
    int index;
    int num_vertices = 0;

    /*
    * Generate vertices.
    */
    diguyVec3f* vertices;
    diguyVec3f* normals;
    diguyVec2f* texture_indices;

    int num_indices;
    unsigned int* indices;

    diguyAuthorInterface::generate_cylinder_vertices(radius,
        height,
        num_slices,
        num_layers,
        position_offset,
        orientation,

        0,

    /*
    * Create a geometry drawable object.
    */
    osg::Geometry* geometry = new osg::Geometry;

```

```

geometry->setDataVariance(osg::Object::STATIC);
geometry->setUseVertexBufferObjects(true);

/*
 * Create osg vertices array.
 */
osg::Vec3Array* osg_vertices = new osg::Vec3Array;
osg_vertices->reserve(num_vertices);

osg::Vec3Array* osg_normals = new osg::Vec3Array;
osg_normals->reserve(num_vertices);

float* vptr =
float* nptr =

for (index=0; index<num_vertices; index++)
{
    osg_vertices->push_back(osg::Vec3(*vptr, *(vptr+1), *(vptr+2)));
    vptr += 3;

    osg_normals->push_back(osg::Vec3(*nptr, *(nptr+1), *(nptr+2)));
    nptr += 3;
}

geometry->setVertexArray(osg_vertices);
geometry->setNormalArray(osg_normals);
geometry->setNormalBinding(osg::Geometry::BIND_PER_VERTEX);

/*
 * Create osg texture indices array.
 */
// not done yet

/*
 * Triangle vertex indices.
 */
osg::DrawElementsUInt* index_set = new osg::DrawElementsUInt(osg::PrimitiveSet::TRIANGULAR_FAN);
index_set->reserve(num_indices);
geometry->addPrimitiveSet(index_set);

int i;
for (i=0; i<num_indices; i++)
    index_set->push_back(indices[i]);

diguyAuthorInterface::delete_generated_vertices(vertices,
    normals,
    texture_indices,
    indices);

/*
 * State set.
 */
osg::StateSet* state_set = create_material_state_set(color, true);
geometry->setStateSet(state_set);

return geometry;
}

// static

```



```

osg::Geometry*
diguyOsgUtil::create_cylinder_wireframe(float radius,
    float height,
    int num_slices,
    int num_layers,
    const diguyVec3f &position_offset,
    const diguyVec3f &orientation,
    const diguyVec4f &color,
    float line_width,
    bool lit)
{
    int index;
    int num_vertices = 0;

    /*
     * Generate vertices.
     */
    diguyVec3f* vertices;
    diguyVec3f* normals;
    diguyVec2f* texture_indices;

    int num_indices;
    unsigned int* indices;

    diguyAuthorInterface::generate_cylinder_vertices(radius,
        height,
        num_slices,
        num_layers,
        position_offset,
        orientation,

        1,

        /*
         * Create a geometry drawable object.
         */
        osg::Geometry* geometry = new osg::Geometry;

        geometry->setDataVariance(osg::Object::STATIC);
        geometry->setUseVertexBufferObjects(true);

        /*
         * Create osg vertices array.
         */
        osg::Vec3Array* osg_vertices = new osg::Vec3Array(num_indices);
        for (index=0; index<num_indices; index++)
        {
            int vertex_index = indices[index];

            float* vptr = _index].x);
            (*osg_vertices)[index].set(vptr[0], vptr[1], vptr[2]);
        }
        geometry->setVertexArray(osg_vertices);

        if (lit)
        {

```

```

        osg::Vec3Array* osg_normals = new osg::Vec3Array(num_indices);
        for (index=0; index<num_indices; index++)
        {
            int vertex_index = indices[index];

            float* nptr = index].x);
            (*osg_normals)[index].set(nptr[0], nptr[1], nptr[2]);
        }
        geometry->setNormalArray(osg_normals);
        geometry->setNormalBinding(osg::Geometry::BIND_PER_VERTEX);
    }

    geometry->addPrimitiveSet(new osg::DrawArrays(osg::PrimitiveSet::LINES, 0, num_indices);

    diguyAuthorInterface::delete_generated_vertices(vertices,
        normals,
        texture_indices,
        indices);

    /*
     * State set.
     */
    osg::StateSet* state_set = create_material_state_set(color, false);

    osg::LineWidth* osg_line_width = new osg::LineWidth;
    osg_line_width->setWidth(line_width);

    state_set->setAttributeAndModes(osg_line_width, osg::StateAttribute::ON);
    state_set->setMode(GL_BLEND, osg::StateAttribute::OFF);
    state_set->setTextureMode(0, GL_TEXTURE_1D, osg::StateAttribute::OFF);
    state_set->setTextureMode(0, GL_TEXTURE_2D, osg::StateAttribute::OFF);
    if (!lit)
        state_set->setMode(GL_LIGHTING, osg::StateAttribute::OVERRIDE|osg::StateAttribute

    geometry->setStateSet(state_set);

    return geometry;
}

```

```

// static
osg::Geometry*
diguyOsgUtil::create_box_solid(const diguyVec3f &min,
    const diguyVec3f &max,
    const diguyVec3f &position_offset,
    const diguyVec3f &orientation,
    const diguyVec4f &color)
{
    int index;
    int num_vertices = 0;

    /*
     * Generate vertices.
     */
    diguyVec3f* vertices;
    diguyVec3f* normals;
    diguyVec2f* texture_indices;

    int num_indices;
    unsigned int* indices;

```

```

diguyAuthorInterface::generate_box_vertices(min,
    max,
    position_offset,
    orientation,

    0,

    /*
     * Create a geometry drawable object.
     */
    osg::Geometry* geometry = new osg::Geometry;

    geometry->setDataVariance(osg::Object::STATIC);
    geometry->setUseVertexBufferObjects(true);

    /*
     * Create osg vertices array.
     */
    osg::Vec3Array* osg_vertices = new osg::Vec3Array;
    osg_vertices->reserve(num_vertices);

    osg::Vec3Array* osg_normals = new osg::Vec3Array;
    osg_normals->reserve(num_vertices);

    float* vptr =
    float* nptr =

    for (index=0; index<num_vertices; index++)
    {
        osg_vertices->push_back(osg::Vec3(*vptr, *(vptr+1), *(vptr+2)));
        vptr += 3;

        osg_normals->push_back(osg::Vec3(*nptr, *(nptr+1), *(nptr+2)));
        nptr += 3;
    }

    geometry->setVertexArray(osg_vertices);
    geometry->setNormalArray(osg_normals);
    geometry->setNormalBinding(osg::Geometry::BIND_PER_VERTEX);

    /*
     * Create osg texture indices array.
     */
    // not done yet

    /*
     * Triangle vertex indices.
     */
    osg::DrawElementsUInt* index_set = new osg::DrawElementsUInt(osg::PrimitiveSet::TRIANG
    index_set->reserve(num_indices);
    geometry->addPrimitiveSet(index_set);

    int i;
    for (i=0; i<num_indices; i++)
        index_set->push_back(indices[i]);

```

```

    diguyAuthorInterface::delete_generated_vertices(vertices,
        normals,
        texture_indices,
        indices);

    /*
     * State set.
     */
    osg::StateSet* state_set = create_material_state_set(color, true);
    geometry->setStateSet(state_set);

    return geometry;
}

```

```

// static
osg::Geometry*
diguyOsgUtil::create_box_wireframe(const diguyVec3f &min,
    const diguyVec3f &max,
    const diguyVec3f &position_offset,
    const diguyVec3f &orientation,
    const diguyVec4f &color,
    float line_width,
    bool lit)
{
    int index;
    int num_vertices = 0;

    /*
     * Generate vertices.
     */
    diguyVec3f* vertices;
    diguyVec3f* normals;
    diguyVec2f* texture_indices;

    int num_indices;
    unsigned int* indices;

    diguyAuthorInterface::generate_box_vertices(min,
        max,
        position_offset,
        orientation,

        1,                // for_wireframe

    /*
     * Create a geometry drawable object.
     */
    osg::Geometry* geometry = new osg::Geometry;

    geometry->setDataVariance(osg::Object::STATIC);
    geometry->setUseVertexBufferObjects(true);

    /*
     * Create osg vertices array.
     */
}

```

```

osg::Vec3Array* osg_vertices = new osg::Vec3Array(num_indices);
for (index=0; index<num_indices; index++)
{
    int vertex_index = indices[index];

    float* vptr = _index].x);
    (*osg_vertices)[index].set(vptr[0], vptr[1], vptr[2]);
}
geometry->setVertexArray(osg_vertices);

if (lit)
{
    osg::Vec3Array* osg_normals = new osg::Vec3Array(num_indices);
    for (index=0; index<num_indices; index++)
    {
        int vertex_index = indices[index];

        float* nptr = index].x);
        (*osg_normals)[index].set(nptr[0], nptr[1], nptr[2]);
    }
    geometry->setNormalArray(osg_normals);
    geometry->setNormalBinding(osg::Geometry::BIND_PER_VERTEX);
}

geometry->addPrimitiveSet(new osg::DrawArrays(osg::PrimitiveSet::LINES, 0, num_indices);

diguyAuthorInterface::delete_generated_vertices(vertices,
    normals,
    texture_indices,
    indices);

/*
 * State set.
 */
osg::StateSet* state_set = create_material_state_set(color, false);

osg::LineWidth* osg_line_width = new osg::LineWidth;
osg_line_width->setWidth(line_width);

state_set->setAttributeAndModes(osg_line_width, osg::StateAttribute::ON);
state_set->setMode(GL_BLEND, osg::StateAttribute::OFF);
state_set->setTextureMode(0, GL_TEXTURE_1D, osg::StateAttribute::OFF);
state_set->setTextureMode(0, GL_TEXTURE_2D, osg::StateAttribute::OFF);
if (!lit)
    state_set->setMode(GL_LIGHTING, osg::StateAttribute::OVERRIDE|osg::StateAttribute::ON);

geometry->setStateSet(state_set);

return geometry;
}

```

```

// static
osg::StateSet*
diguyOsgUtil::create_material_state_set(const diguyVec4f &color, bool lit)
{
    osg::Material* material = new osg::Material;

    material->setAmbient(osg::Material::FRONT, osg::Vec4(color.r, color.g, color.b, color.a));
    material->setDiffuse(osg::Material::FRONT, osg::Vec4(color.r, color.g, color.b, color.a));
    material->setSpecular(osg::Material::FRONT, osg::Vec4(0.0f, 0.0f, 0.0f, 1.0f));
}

```

```

material->setEmission(osg::Material::FRONT, osg::Vec4(0.0f, 0.0f, 0.0f, 1.0f));
material->setShininess(osg::Material::FRONT, 10.0f);

osg::StateSet* state_set = new osg::StateSet;
state_set->setAttribute(material);

if (color.w < 1.0f)
{
    osg::BlendFunc* bf = new osg::BlendFunc(osg::BlendFunc::SRC_ALPHA,
        osg::BlendFunc::ONE_MINUS_SRC_ALPHA);
    state_set->setAttributeAndModes(bf);

    state_set->setRenderingHint(osg::StateSet::TRANSPARENT_BIN);
}
else
{
    state_set->setRenderingHint(osg::StateSet::OPAQUE_BIN);
}

osg::CullFace* cull = new osg::CullFace;
cull->setMode(osg::CullFace::BACK);
state_set->setAttributeAndModes(cull, osg::StateAttribute::ON|osg::StateAttribute::OVERWRITE_REPLACE);

return state_set;
}

```

```

// static
osg::StateSet*
diguyOsgUtil::create_material_state_set(diguyAuthorVisualMaterial* visual_material)
{
    osg::Material* material = new osg::Material;

    diguyVec4f brush_color = visual_material->get_brush_color();

    material->setAmbient(osg::Material::FRONT, diguyVec4_to_osgVec4(brush_color));
    material->setDiffuse(osg::Material::FRONT, diguyVec4_to_osgVec4(brush_color));
    material->setSpecular(osg::Material::FRONT, osg::Vec4(0.0f, 0.0f, 0.0f, 1.0f));
    material->setEmission(osg::Material::FRONT, osg::Vec4(0.0f, 0.0f, 0.0f, 1.0f));
    material->setShininess(osg::Material::FRONT, 10.0f);

    osg::StateSet* state_set = new osg::StateSet;
    state_set->setAttribute(material);

    if (brush_color.w < 1.0f)
    {
        osg::BlendFunc* bf = new osg::BlendFunc(osg::BlendFunc::SRC_ALPHA,
            osg::BlendFunc::ONE_MINUS_SRC_ALPHA);
        state_set->setAttributeAndModes(bf);

        state_set->setRenderingHint(osg::StateSet::TRANSPARENT_BIN);
    }
    else
    {
        state_set->setRenderingHint(osg::StateSet::OPAQUE_BIN);
    }

    osg::CullFace* cull = new osg::CullFace;
    cull->setMode(osg::CullFace::BACK);
    state_set->setAttributeAndModes(cull, osg::StateAttribute::ON|osg::StateAttribute::OVERWRITE_REPLACE);
}

```

```

    return state_set;
}

```

```

// static
osg::Vec3
diguyOsgUtil::diguyVec3_to_osgVec3(const diguyVec3f &diguy_vec3)
{
    return osg::Vec3(diguy_vec3.x, diguy_vec3.y, diguy_vec3.z);
}

```

```

// static
osg::Vec4
diguyOsgUtil::diguyVec4_to_osgVec4(const diguyVec4f &diguy_vec4)
{
    return osg::Vec4(diguy_vec4.x, diguy_vec4.y, diguy_vec4.z, diguy_vec4.w);
}

```

```

// static
osg::Quat
diguyOsgUtil::diguy_euler_zxy_radians_to_osgQuat(const diguyVec3f &diguy_euler)
{
    diguyVec4f diguy_quat = diguy_euler.diguy_euler_angles_to_quaternion();
    return osg::Quat(diguy_quat.qx, diguy_quat.qy, diguy_quat.qz, diguy_quat.qw);
}

```

```

// static
osg::Quat
diguyOsgUtil::diguy_euler_zxy_degrees_to_osgQuat(const diguyVec3f &diguy_euler)
{
    diguyVec3f diguy_euler_radians(diguy_euler.rz * DIGUY_DEG2RAD, // yaw
    diguy_euler.rx * DIGUY_DEG2RAD, // roll
    diguy_euler.ry * DIGUY_DEG2RAD); // pitch

    diguyVec4f diguy_quat = diguy_euler_radians.diguy_euler_angles_to_quaternion();
    return osg::Quat(diguy_quat.qx, diguy_quat.qy, diguy_quat.qz, diguy_quat.qw);
}

```

```

// static
diguyVec3f
diguyOsgUtil::osgVec3_to_diguyVec3(const osg::Vec3&osg_vec3)
{
    return diguyVec3f(osg_vec3.x(), osg_vec3.y(), osg_vec3.z());
}

```

```

// static
diguyVec4f
diguyOsgUtil::osgVec4_to_diguyVec4(const osg::Vec4&osg_vec4)
{
    return diguyVec4f(osg_vec4.x(), osg_vec4.y(), osg_vec4.z(), osg_vec4.w());
}

```

```

// static
void
diguyOsgUtil::set_name_from_uid(osg::Object* object, long uid)

```

```
{  
    char sprintf_temp[128];  
    sprintf(sprintf_temp, "%ld", uid);  
    object->setName(sprintf_temp);  
}
```


5.70 diguyPathShape

diguyPathShape.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

- ◆ [Waypoint Functions](#)
- ◆ [Navmesh Functions](#)
- ◆ [Experimental Functions](#)

Link against: libdiguy

5.70.1 class diguyPathShape

```
class BDI_DECLSPEC_diguy diguyPathShape
{
public:
    const char* get_name();
    int set_name(const char* name);
    long get_uid();
    int set_clamp_at_end_flag(int value);
    int get_clamp_at_end_flag();
    int update();
    float get_length();
    int get_point_at_distance(float distance_into_path_shape,
        float* x, float* y, float* z,
        float* rz, float* rx, float* ry);
    int get_num_waypoints();
    diguyWaypoint* get_waypoint_at_index(int index);
    diguyWaypoint* create_waypoint(float x = 0.0f, float y = 0.0f, float z = 0.0f,
        float yaw = 0.0f, float roll = 0.0f, float pitch = 0.0f,
        float weight = 1.0f,
        int index = -1);
    int add_waypoint(diguyWaypoint* waypoint);
    void set_use_as_navmesh_portal(int use_as_portal);
    int get_use_as_navmesh_portal();
    void set_one_way_portal(int one_way);
    int get_one_way_portal();
    int translate(float tx, float ty, float tz);
    int rotate_about_point(float rz, float rx, float ry,
        float rotation_pt_x, float rotation_pt_y, float rotation_pt_z);
    int set_ground_clamp_when_on_path(int ground_clamp_when_on_path);
    int get_ground_clamp_when_on_path();
};
```

5.70.2 function diguyPathShape::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the path shape. This pointer will never be NULL.

Returns:

name of the path shape

Callable From:

- ◇ C++
- ◇ Script

5.70.3 function diguyPathShape::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.70.4 function diguyPathShape::get_uid

Prototype:

```
long get_uid();
```

Description:

All path shapes are assigned a unique identifier, or uid. This function returns this path shape's uid.

Note: unique identifiers will change between DI-Guy runs!

Returns:

unique identifier of object

Callable From:

- ◇ C++
- ◇ Script

5.70.5 function diguyPathShape::set_clamp_at_end_flag

Prototype:

```
int set_clamp_at_end_flag(int value);
```

Description:

This function determines what happens to objects following this path when they reach the end.

If 1, the object is restricted to stay at the end of the path.

If 0, the object will wrap to the beginning of the path.

Arguments:

value 1 to clamp, 0 to wrap

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.70.6 function diguyPathShape::get_clamp_at_end_flag

Prototype:

```
int get_clamp_at_end_flag();
```

Returns:

most recent setting of `set_clamp_at_end_flag()`.

5.70.7 function diguyPathShape::update

Prototype:

```
int update();
```

Description:

This function should be called after all modifications have been made to the path shape and its waypoints and it is time to re-derive the overall path shape.

This update is deferred for reasons of performance; if there are a lot of modifications to be made to the path shape they can all be done in a group with only a single final update.

A call to `update( )` is typically required after certain calls are made on objects owned by the path, such as a `diguyWaypoint`. The documentation for function calls of these objects will state whether a subsequent `update( )` call is necessary.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.70.8 function diguyPathShape::get_length

Prototype:

```
float get_length();
```

Description:

This function returns the total length of the path shape, in meters.

Returns:

length of path shape in meters

Callable From:

- ◇ C++
- ◇ Script

5.70.9 function diguyPathShape::get_point_at_distance

Prototype:

```
int get_point_at_distance(float distance_into_path_shape,
    float* x, float* y, float* z,
    float* rz, float* rx, float* ry);
```

Description:

This function returns, via the passed variable pointers, the position and orientation of the point on the path shape at the specified distance.

Arguments:

distance_into_path_shape	distance in meters into path shape of the desired point
x, y, z	position of point in meters from the origin
rz, rx, ry	orientation of point in degrees counter-clockwise from the positive X axis

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Waypoint Functions

5.70.10 function diguyPathShape::get_num_waypoints**Prototype:**

```
int get_num_waypoints();
```

Returns:

the number of waypoints belonging to this path shape

Callable From:

- ◇ C++
- ◇ Script

5.70.11 function diguyPathShape::get_waypoint_at_index**Prototype:**

```
diguyWaypoint* get_waypoint_at_index(int index);
```

Description:

This function gets the waypoint at the specified index and returns a pointer to it. The returned waypoint can then be queried for information or modified.

Note that if the waypoint is modified, the [update\(\)](#) function must be called to update the path shape.

Arguments:

index	index of the waypoint; indices start at 0

Returns:

pointer of type [diguyWaypoint](#); NULL if no waypoint at the specified index

Callable From:

- ◇ C++
- ◇ Script

5.70.12 function diguyPathShape::create_waypoint

Prototype:

```
diguyWaypoint* create_waypoint(float x = 0.0f, float y = 0.0f, float z = 0.0f,
                                float yaw = 0.0f, float roll = 0.0f, float pitch = 0.0f,
                                float weight = 1.0f,
                                int index = -1);
```

Description:

This function creates a new waypoint that is added to the path shape. Do not call `diguyScenario::destroy_waypoint( )` on the returned waypoint; it will be destroyed automatically with the path shape.

Arguments:

tx, ty, tz	position in meters from the origin
rz, rx, ry	orientations in degrees counter-clockwise from the positive X axis
weight	

how much influence this waypoint exerts over the path curve

index where the waypoint should be inserted in the path shape; indices start at 0; pass -1 to add to end of path shape

Returns:

pointer to type [diguyWaypoint](#)

Callable From:

- ◇ C++
- ◇ Script

5.70.13 function [diguyPathShape::add_waypoint](#)

Prototype:

```
int add_waypoint(diguyWaypoint* waypoint);
```

Description:

This function adds a waypoint to the path shape.

The waypoint should be created using the function [diguyScenario::create_waypoint\(\)](#), and destroyed using the function [diguyScenario::destroy_waypoint\(\)](#).

An internal copy of the waypoint is made, so it can be destroyed right after this function call or used for other path shapes.

Arguments:

waypoint waypoint to be appended to end of path

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Navmesh Functions

5.70.14 function diguyPathShape::set_use_as_navmesh_portal**Prototype:**

```
void set_use_as_navmesh_portal(int use_as_portal);
```

Description:

This function tells the Navmesh regions to use this path as a connection between two regions. The Navmesh should automatically choose the regions closest to the start and end points of the path. By default this functionality is off, but once activated it's assumed to be a two way connection.

Arguments:

`use_as_portal` 1 to use as a connection, 0 to not

Callable From:

- ◇ C++
- ◇ Script

5.70.15 function diguyPathShape::get_use_as_navmesh_portal**Prototype:**

```
int get_use_as_navmesh_portal();
```

Description:

This function returns if the path is being used as a connection between two regions.

1 if is a connection, 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.70.16 function diguyPathShape::set_one_way_portal**Prototype:**

```
void set_one_way_portal(int one_way);
```

Description:

This function tells the Navmesh regions to use this path as a one way connection between two regions. Defaults to 0 (two way connection)

Arguments:

`one_way` 1 to set as one way connection, 0 to set as bidirectional

Callable From:

- ◇ C++
- ◇ Script

5.70.17 function diguyPathShape::get_one_way_portal**Prototype:**

```
int get_one_way_portal();
```

Description:

This function returns if the path is being used as a one way connection between two regions.

1 if is one way connection, 0 if two way

Callable From:

- ◇ C++
- ◇ Script

Experimental Functions

The following functions are experimental and may change or disappear with no warning.

5.70.18 function diguyPathShape::translate

Prototype:

```
int translate(float tx, float ty, float tz);
```

5.70.19 function diguyPathShape::rotate_about_point**Prototype:**

```
int rotate_about_point(float rz, float rx, float ry,  
    float rotation_pt_x, float rotation_pt_y, float rotation_pt_z);
```

**5.70.20 function
diguyPathShape::set_ground_clamp_when_on_path****Prototype:**

```
int set_ground_clamp_when_on_path(int ground_clamp_when_on_path);
```

5.70.21 function diguyPathShape::get_ground_clamp_when_on_path

Prototype:

```
int get_ground_clamp_when_on_path();
```


5.71 diguyPluginKeyboardEvent

diguyPluginKeyboardEvent.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

Objects of this type are passed into the plugin keyboard functions `key_press( )` and `key_release( )`.

5.71.1 class diguyPluginKeyboardEvent

```
class BDI_DECLSPEC_diguy diguyPluginKeyboardEvent
{
public:
    diguyView* get_view();
    int get_key();
    int get_x();
    int get_y();
}
```

5.71.2 function diguyPluginKeyboardEvent::get_view

Prototype:

```
diguyView* get_view();
```

Description:

This function returns the view in which the keyboard event took place.

Returns:

object of type `diguyView`

Callable From:

◇ C++

5.71.3 function diguyPluginKeyboardEvent::get_key

Prototype:

```
int get_key();
```

Description:

This function returns the key that was pressed or released.

Returns:

code of key

Callable From:

◇ C++

5.71.4 function diguyPluginKeyboardEvent::get_x

Prototype:

```
int get_x();
```

Description:

This function returns the screen x coordinate at which the keyboard event took place.

Returns:

screen x coordinate

Callable From:

◇ C++

5.71.5 function diguyPluginKeyboardEvent::get_y**Prototype:**

```
int get_y();
```

Description:

This function returns the screen y coordinate at which the keyboard event took place.

Returns:

screen y coordinate

Callable From:

◇ C++

5.72 diguyPluginMouseEvent

diguyPluginMouseEvent.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

Objects of this type are passed into the plugin mouse functions `mouse_press( )`, `mouse_drag( )`, `mouse_release( )`, and `mouse_move( )`.

5.72.1 class diguyPluginMouseEvent

```
class BDI_DECLSPEC_diguy diguyPluginMouseEvent
{
public:
    diguyView* get_view();
    int get_x();
    int get_y();
    enum {
        BUTTON_STATE_LEFT    = 0x0001,
        BUTTON_STATE_RIGHT   = 0x0002,
        BUTTON_STATE_MIDDLE  = 0x0004,
        BUTTON_STATE_SHIFT   = 0x0008,
        BUTTON_STATE_CTRL    = 0x0010,
        BUTTON_STATE_ALT     = 0x0020
    };
    int get_button_state();
};
```

5.72.2 function diguyPluginMouseEvent::get_view

Prototype:

```
diguyView* get_view();
```

Description:

This function returns the view in which the mouse event took place.

Returns:

object of type `diguyView`

Callable From:

◇ C++

5.72.3 function diguyPluginMouseEvent::get_x

Prototype:

```
int get_x();
```

Description:

This function returns the screen x coordinate at which the mouse event took place.

Returns:

screen x coordinate

Callable From:

◇ C++

5.72.4 function diguyPluginMouseEvent::get_y

Prototype:

```
int get_y();
```

Description:

This function returns the screen y coordinate at which the mouse event took place.

Returns:

screen y coordinate

Callable From:

◇ C++

5.72.5 enumeration

```
enum {
    BUTTON_STATE_LEFT    = 0x0001,
    BUTTON_STATE_RIGHT   = 0x0002,
    BUTTON_STATE_MIDDLE  = 0x0004,
    BUTTON_STATE_SHIFT   = 0x0008,
    BUTTON_STATE_CTRL    = 0x0010,
    BUTTON_STATE_ALT     = 0x0020
};
```

Description:

This is an enumeration of the different button states reported by [get_button_state\(\)](#). The values may be OR-ed together. e.g., the value 0x21 would indicate that the left mouse button was clicked while the ALT was down.

Usable From:

◇ C++
◇ Script

5.72.6 function diguyPluginMouseEvent::get_button_state

Prototype:

```
int get_button_state();
```

Description:

This function returns state of the mouse buttons when the mouse event took place. Refer to the enumeration above for possible values.

Returns:

OR-ed together state of mouse buttons

Callable From:

◇ C++

5.73 diguyRegion

diguyRegion.h

DI-Guy SDK Version 12.5.2

Includes: [declspec_diguy.h](#) | [diguy_constants.h](#)

Contents:

- ◇ [Query Functions](#)
- ◇ [Editing Functions](#)

Link against: **libdiguy**

Summary

API for accessing [diguyRegion](#) objects generated by the crowd and region painter.

For information on subregions, see the documentation for the [diguySubregionIndex](#) and [diguySubregionMask](#) enumerations in [diguy_constants.h](#).

5.73.1 class diguyRegion

```
class BDI_DECLSPEC_diguy diguyRegion
{
public:
    const char* get_name();
    int is_point_in_region(diguySubregionMask subregion_mask,
        float x, float y, float z);
    int get_point_in_region(diguySubregionIndex subregion_index,
        float* res_x, float* res_y, float* res_z);
    int get_point_in_region_near_point(diguySubregionIndex subregion_index,
        float seed_x, float seed_y, float seed_z,
        float radius,
        float* res_x, float* res_y, float* res_z);
    int get_nearest_point_in_region(diguySubregionMask subregion_mask,
        float seed_x, float seed_y, float seed_z,
        float* res_x, float* res_y, float* res_z);
    int get_nearest_reachable_point(diguySubregionMask subregion_mask,
        float seed_x, float seed_y, float seed_z,
        float* res_x, float* res_y, float* res_z);
    int get_num_rows();
    int get_num_columns();
    float get_resolution();
    float get_random_factor();
    int shrink();
    int clear_region();
    int clear_subregion(diguySubregionIndex subregion_index);
    int resample_mesh(float resolution);
    int paint(diguySubregionIndex subregion_index,
        float x, float y, float z,
        float radius,
```



```

        int force_visual_update = 0);
int paint_rectangle(diguySubregionIndex subregion_index,
    float min_x, float min_y, float min_z,
    float max_x, float max_y, float max_z,
    int force_visual_update = 0);
int erase(diguySubregionIndex subregion_index,
    float x, float y, float z,
    float radius,
    int force_visual_update = 0);
int erase_rectangle(diguySubregionIndex subregion_index,
    float min_x, float min_y, float min_z,
    float max_x, float max_y, float max_z,
    int force_visual_update = 0);
int generate_navmesh(float min_x, float min_y, float min_z,
    float max_x, float max_y, float max_z);

```

Query Functions

5.73.2 function diguyRegion::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the region. This pointer will never be NULL.

Returns:

name of the region

Callable From:

- ◇ C++
- ◇ Script

5.73.3 function diguyRegion::is_point_in_region

Prototype:

```
int is_point_in_region(diguySubregionMask subregion_mask,
    float x, float y, float z);
```

Description:

Returns 1 if a point is approximately inside the region. Supports multiple subregions via or'ing masks together.

Use the base mask ([DIGUY_SUBREGION_MASK_BASE](#)) to test against the base of the region.

Returns:

1 if the point is in the region, 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.73.4 function diguyRegion::get_point_in_region

Prototype:

```
int get_point_in_region(diguySubregionIndex subregion_index,
    float* res_x, float* res_y, float* res_z);
```

Description:

Returns a random x, y, z point in the region specified. This function takes a region index (e.g. [DIGUY_SUBREGION_BASE](#)).

Arguments:

subregion_index subregion in which to find the point
 res_x, res_y, return point
 res_z

Returns:

0 if a point was successfully generated, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.73.5 function diguyRegion::get_point_in_region_near_point

Prototype:

```
int get_point_in_region_near_point(diguySubregionIndex subregion_index,
    float seed_x, float seed_y, float seed_z,
    float radius,
    float* res_x, float* res_y, float* res_z);
```

Description:

Returns a random x, y, z point, near the seed location, in the subregion specified.

Arguments:

subregion_index	subregion in which to find the point
seed_x, seed_y, seed_z	location to use as the seed point
radius	radius around the seed point within which the result point should be located
	return point

res_x, res_y,
res_z

Returns:

0 if a point was successfully generated, -1 on failure

Callable From:

◇ C++
◇ Script

5.73.6 function diguyRegion::get_nearest_point_in_region

Prototype:

```
int get_nearest_point_in_region(diguySubregionMask subregion_mask,
    float seed_x, float seed_y, float seed_z,
    float* res_x, float* res_y, float* res_z);
```

Description:

Returns the nearest point in the region near the seed location, in the subregion(s) specified.

Arguments:

subregion_mask subregion(s) to search
seed_x, location to use as the search seed point
seed_y, seed_z
res_x, res_y, return point
res_z

Returns:

0 if a point was successfully generated, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.73.7 function diguyRegion::get_nearest_reachable_point

Prototype:

```
int get_nearest_reachable_point(diguySubregionMask subregion_mask,
    float seed_x, float seed_y, float seed_z,
    float* res_x, float* res_y, float* res_z);
```

Description:

Returns the nearest reachable point to the seed location, inside the subregion(s) specified by the mask.

Determining reachability involves guaranteeing that there is a contiguous section of the base region between the seed point and the resulting point.

Arguments:

subregion_mask	the subregion(s) to search
seed_x,	the location to use as the search seed point
seed_y, seed_z	
	return point

res_x, res_y,
res_z

Returns:

0 if a point was successfully found, -1 on failure

Callable From:

◇ C++
◇ Script

5.73.8 function diguyRegion::get_num_rows

Prototype:

int get_num_rows();

Returns:

the number of rows in the region

Callable From:

◇ C++
◇ Script

5.73.9 function diguyRegion::get_num_columns

Prototype:

```
int get_num_columns();
```

Returns:

the number of columns in the region

Callable From:

- ◇ C++
- ◇ Script

5.73.10 function diguyRegion::get_resolution

Prototype:

```
float get_resolution();
```

Returns:

the resolution of the region in meters per point

Callable From:

- ◇ C++
- ◇ Script

5.73.11 function diguyRegion::get_random_factor

Prototype:

```
float get_random_factor();
```

Description:

This function returns a random factor between 0 and 1 computed for this region. This number can be used, for example, to modify colors or depths of region visuals.

The number will remain the same for the lifetime of the region.

Returns:

a random factor between 0 and 1

Callable From:

- ◇ C++
- ◇ Script

Editing Functions

5.73.12 function diguyRegion::shrink

Prototype:

```
int shrink();
```

Description:

Shrinks the region's bounds to just the area with tagged data in it. Can help conserve memory. Currently always returns 0.

Callable From:

- ◇ C++
- ◇ Script

5.73.13 function diguyRegion::clear_region

Prototype:

```
int clear_region();
```

Description:

Erases all region data and then shrinks the region.

Callable From:

- ◇ C++
- ◇ Script

5.73.14 function diguyRegion::clear_subregion

Prototype:

```
int clear_subregion(diguySubregionIndex subregion_index);
```

Description:

Erases all subregion data and then shrinks the region.

Callable From:

- ◇ C++
- ◇ Script

5.73.15 function diguyRegion::resample_mesh

Prototype:

```
int resample_mesh(float resolution);
```

Description:

Resamples the mesh and sets the resolution of the mesh to the new resolution.

Note that the resampled data is currently rough and will probably need to be touched up by hand. The system is currently limited to creating meshes with a resolution of 0.25 meters per point.

Callable From:

- ◇ C++
- ◇ Script

5.73.16 function diguyRegion::paint

Prototype:

```
int paint(diguySubregionIndex subregion_index,
          float x, float y, float z,
          float radius,
          int force_visual_update = 0);
```

Description:

This function adds to the specified subregion with a spherical brush centered at the specified point and with the specified radius.

If you are doing a number of edits to the mesh at once, it's preferable to wait until the last one before setting force_visual_update to 1.

Callable From:

- ◇ C++
- ◇ Script

5.73.17 function diguyRegion::paint_rectangle

Prototype:

```
int paint_rectangle(diguySubregionIndex subregion_index,
                    float min_x, float min_y, float min_z,
                    float max_x, float max_y, float max_z,
                    int force_visual_update = 0);
```

Description:

This function adds to the specified subregion with a rectangular brush stretching from the min point to the max point specified.

If you are doing a number of edits to the mesh at once, it's preferable to wait until the last one before setting force_visual_update to 1.

Callable From:

- ◇ C++
- ◇ Script

5.73.18 function diguyRegion::erase**Prototype:**

```
int erase(diguySubregionIndex subregion_index,
         float x, float y, float z,
         float radius,
         int force_visual_update = 0);
```

Description:

This function removes from the specified subregion with a spherical brush centered at the specified point and with the specified radius.

If you are doing a number of edits to the mesh at once, it's preferable to wait until the last one before setting force_visual_update to 1.

Returns:

1 if any points were erased, 0 if not

Callable From:

◇ C++
◇ Script

5.73.19 function diguyRegion::erase_rectangle**Prototype:**

```
int erase_rectangle(diguySubregionIndex subregion_index,
                  float min_x, float min_y, float min_z,
                  float max_x, float max_y, float max_z,
                  int force_visual_update = 0);
```

Description:

This function removes from the specified subregion with a rectangular brush stretching from min point to the max point specified.

If you are doing a number of edits to the mesh at once, it's preferable to wait until the last one before setting force_visual_update to 1.

Returns:

1 if any points were erased, 0 if not

Callable From:

◇ C++
 ◇ Script

5.73.20 function diguyRegion::generate_navmesh

Prototype:

```
int generate_navmesh(float min_x, float min_y, float min_z,
    float max_x, float max_y, float max_z);
```

Description:

This function attempts to grow the region to encompass the min and max points specified as well as determine the reachability of all of those points at once. Note: the navmesh reachability algorithm is not perfect and the results will probably need to be touched up by hand.

Callable From:

◇ C++
 ◇ Script

5.74 diguyScenario

diguyScenario.h

DI-Guy SDK Version 12.5.2

Includes: [declspec_diguy.h](#) | [diguyViewCamera.h](#) | [diguyViewFog.h](#) | [diguyViewLight.h](#) | [diguyMotionDirection.h](#) | [diguyMotionPosture.h](#) | [diguyMotionVariant.h](#) | [diguyCharacterTypeMap.h](#) | [diguyCharacterFaceExpression.h](#) | [diguy_vector_classes.h](#) | [diguy_constants.h](#) | [diguy_typedefs.h](#) | [diguyCharacterPath.h](#) | [diguyInteractionMachine.h](#) | [diguySensorRegion.h](#) | [diguySignal.h](#) | [diguyVariable.h](#) | [diguyView.h](#)

Contents:

- ◆ General Functions
- ◆ Time Control and Playback Functions
- ◆ Load and Save Functions
- ◆ Draw Functions
- ◆ Merge Functions
- ◆ Time and Date Functions
 - Datetime Functions
 - Simple Time Functions
- ◆ Appearance Functions
- ◆ Preload Functions
- ◆ Culling Functions
- ◆ Query Functions
- ◆ Character Functions
- ◆ Path Shape Functions
- ◆ Waypoint Functions
- ◆ Character Type Map Functions
- ◆ Signal Functions
- ◆ Sound Functions
- ◆ Group Functions
- ◆ Sensor Region Functions
- ◆ Scene Object Functions
- ◆ View Functions
- ◆ Camera Functions
- ◆ Fog Functions
- ◆ Light Functions
- ◆ Info Popup Functions
- ◆ Variable Functions
- ◆ Face Expression Functions
- ◆ Chain Settings Functions
- ◆ Callback Functions
 - Callback Enums:
- ◆ Event Handler Functions
- ◆ Review Functions
- ◆ LOD Functions
- ◆ Script Functions
- ◆ Meta-Action Functions

- ◆ Interaction Machine Functions
- ◆ I-Guy Functions
- ◆ Label Functions
- ◆ Button Panel Functions
- ◆ Misc. DI-Guy AI Functions
- ◆ Crowd Functions
- ◆ Crowd Profile Functions
- ◆ Load Manager Functions
- ◆ Octtree Functions
- ◆ Altitude Functions
- ◆ Impact and Hit Detection Functions
 - Impact Callback Functions:
- ◆ AI Line of Sight, Intersection Callback Functions
- ◆ Path Planning and Regions
- ◆ Formation Functions
- ◆ DI-Guy Scenario User Interface Functions
- ◆ DI-Guy Weapon Fire Effects Functions
 - ◇ DI-Guy Scenario selection functions
- ◆ Particle System Functions
- ◆ Lua Scripting Module Functions
- ◆ Experimental Functions
- ◆ Experimental Performance Functions
- ◆ Experimental Checkpoint Functions
- ◆ Experimental Shader Functions
- ◆ Deprecated Functions
- ◆ VegaPrime Helper Functions

Link against: libdiguy

5.74.1 class diguyScenario

```
class BDI_DECLSPEC diguy diguyScenario
{
public:
    static diguyScenario* get_current_scenario();
    const char* get_name();
    const char* get_type_name();
    const char* get_description();
#ifdef CPLUSPLUS_ONLY
    int update(float t);
    int update2(float t, float realtime_t);
    int set_playback_loop(int playback_loop);
    int set_tick_dt(float tick_dt);
#endif
    float get_t();
    float get_realtime_t();
    float get_desired_t();
    int set_tout(float tout);
    float get_tout();
    void set_tout_enabled(int tout_enabled);
    int get_tout_enabled();
    int get_playback_loop();
    float get_tick_dt();
};
```

```

int get_ticks_can_be_dropped();
void set_ticks_can_be_dropped(int ticks_can_be_dropped);
long get_max_ticks_behind_until_drop();
void set_max_ticks_behind_until_drop(long max_ticks_behind_until_drop);
int set_playback_mode(diguyScenarioPlaybackMode playback_mode);
int set_playback_mode_stop();
int set_playback_mode_play();
diguyScenarioPlaybackMode get_playback_mode();
float derive_playback_mode_dt(float dt_in);
int reset();
#ifdef CPLUSPLUS_ONLY
int load(const char* filename);
int save();
int save_as(const char* filename);
const char* get_filename();
const char* get_filename_without_directory();
int get_unsaved();
void set_unsaved(int unsaved);
#endif
#ifdef CPLUSPLUS_ONLY
int draw();
int draw_pass1();
int draw_pass2(int also_call_post_draw = 1);
#endif // CPLUSPLUS_ONLY
int post_draw(int call_plugins_post_draw = 1,
int call_lua_packages_draw = 1,
int call_post_draw_callback = 1);
int set_graphics_attach_ptr(void* graphics_attach_ptr);
#endif // CPLUSPLUS_ONLY
diguyScenarioMergeSettings* create_merge_settings(diguyScenarioMergeInitialSettings in
void destroy_merge_settings(diguyScenarioMergeSettings* merge_settings);
int merge(const char* filename,
diguyScenarioMergeSettings* merge_settings = NULL);
int merge_object(const char* string);
int include_object_library(const char* library_name);
int uninclude_object_library(const char* library_name);
void set_base_datetime(int year = -1,
int month = -1,
int day = -1,
int hour = -1,
int minute = -1,
int second = -1,
int msec = -1);
void set_base_datetime(const char* datetime_string,
const char* format);
void get_base_datetime(int* year = NULL,
int* month = NULL,
int* day = NULL,
int* hour = NULL,
int* minute = NULL,
int* second = NULL,
int* msec = NULL);
void set_base_datetime_default_to_now(int to_now);
int get_base_datetime_default_to_now();
void set_base_datetime_default_to_today(int to_today);
int get_base_datetime_default_to_today();
void set_current_datetime(int year = -1,
int month = -1,
int day = -1,
int hour = -1,
int minute = -1,
int second = -1,

```



```

    int msec    = -1);
void set_current_datetime(const char* datetime_string,
    const char* format);
void get_current_datetime(int* year = NULL,
    int* month   = NULL,
    int* day     = NULL,
    int* hour    = NULL,
    int* minute  = NULL,
    int* second  = NULL,
    int* msec    = NULL);
const char* get_current_datetime_string(const char* format);
void set_datetimes_are_utc(int are_utc);
int get_datetimes_are_utc();
void set_utc_offset_hours(float utc_offset_hours);
float get_utc_offset_hours();
void set_use_local_tz_utc_offset_hours(int use_local_tz);
int get_use_local_tz_utc_offset_hours();
void set_current_datetime_advance_method(diguyDatetimeAdvanceMethod advance_method);
diguyDatetimeAdvanceMethod get_current_datetime_advance_method();
void set_current_datetime_advance_factor(float advance_factor);
float get_current_datetime_advance_factor();
void get_network_datetime(int* year = NULL,
    int* month   = NULL,
    int* day     = NULL,
    int* hour    = NULL,
    int* minute  = NULL,
    int* second  = NULL,
    int* msec    = NULL);
int set_tin_time_of_day(float t);
float get_tin_time_of_day();
float get_time_of_day();
#ifdef CPLUSPLUS_ONLY
    int add_appearance(const char* appearance,
        char** equipment_list,
        char** character_types,
        const char* actor);
    int add_appearance_based_on_existing_appearance(const char* appearance,
        const char* existing_appearance,
        char** equipment_list);
    int remove_appearance(const char* appearance);
#endif // CPLUSPLUS_ONLY
int get_num_character_types_with_appearance(const char* appearance_name);
const char *get_character_type_from_appearance(const char* appearance_name, int index);
int preload_character_type(const char* character_type);
int preload_appearance(const char* appearance);
int preload_gesture(const char* character_type, const char* gesture_name);
void set_character_culling_enabled(int character_culling);
int get_character_culling_enabled();
void set_scene_object_culling_enabled(int object_culling);
int get_scene_object_culling_enabled();
void set_scene_object_sub_culling_enabled(int object_culling);
int get_scene_object_sub_culling_enabled();
void set_character_visualize_bounds_enabled(int visualize_bounds);
int get_character_visualize_bounds_enabled();
void set_scene_object_visualize_bounds_enabled(int val);
int get_scene_object_visualize_bounds_enabled();
int get_num_character_types();
const char* get_character_type_at_index(int index);
const char* get_character_type_abbreviation(const char* character_type);
const char* get_character_type_deprecated_at_version(const char* character_type);
const char* get_character_type_deprecated_suggested_alternatives(const char* character

```

```

int get_character_type_quality_bias(const char* character_type);
diguyCharacterTypeMap* get_character_type_map(const char* character_type);
const char* get_character_type_default_appearance(const char* character_type);
const char* get_character_type_default_skinned_appearance(const char* character_type);
int get_character_type_num_appearances(const char* character_type);
const char* get_character_type_appearance_at_index(const char* character_type,
    int index);
int get_character_type_num_head_appearances(const char* character_type);
const char* get_character_type_head_appearance_at_index(const char* character_type,
    int index);
int get_character_type_num_actions(const char* character_type);
const char* get_character_type_action_at_index(const char* character_type,
    int index);
float get_character_type_action_speed(const char* character_type,
    const char* action_name);
float get_character_type_action_duration(const char* character_type,
    const char* action_name);
float get_character_type_action_distance(const char* character_type,
    const char* action_name);
diguyMotionDirection get_character_type_action_direction(
    const char* character_type,
    const char* action_name);
float get_character_type_action_facing_angle_in(const char* character_type,
    const char* action_name);
float get_character_type_action_facing_angle_out(const char* character_type,
    const char* action_name);
int get_character_type_num_gestures(const char* character_type);
const char* get_character_type_gesture_at_index(const char* character_type,
    int index);
diguyCharacter* create_character(const char* name,
    const char* character_type,
    const char* appearance = NULL);
diguyCharacter* create_temporary_character(const char* name,
    const char* character_type,
    const char* appearance = NULL);
int destroy_character(diguyCharacter* character);
int send_character_to_recycle_bin(diguyCharacter* character);
diguyCharacter* retrieve_character_from_recycle_bin(const char* name,
    const char* character_type,
    const char* appearance,
    int make_character_temporary = 0);
int get_num_characters();
diguyCharacter* get_character_at_index(int index);
diguyCharacter* find_character(const char* name);
void look_at_character(diguyCharacter* character);
diguyCharacter* this_character();
diguyCharacterPathEvent* this_event_bead();
int get_num_path_shapes();
diguyPathShape* get_path_shape_at_index(int index);
diguyPathShape* get_current_path_shape();
diguyPathShape* find_path_shape(const char* name);
diguyPathShape* create_path_shape(const char* name);
diguyPathShape* copy_path_shape(diguyPathShape* existing_path_shape,
    const char* name);
int destroy_path_shape(diguyPathShape* path_shape);
diguyWaypoint* create_waypoint(float x = 0.0f, float y = 0.0f, float z = 0.0f,
    float yaw = 0.0f, float roll = 0.0f, float pitch = 0.0f,
    float weight_in = DIGUY_DEFAULT_FLOAT,
    float weight_out = DIGUY_DEFAULT_FLOAT);
int destroy_waypoint(diguyWaypoint* waypoint);
diguyCharacterTypeMap* get_nearest_character_type_map(

```

```

    diguyCharacterTypeMapField field1, const char* field1_value, const char* field1_a
    diguyCharacterTypeMapField field2 = DIGUY_CHARACTER_TYPE_MAP_FIELD_UNKNOWN, const
    diguyCharacterTypeMapField field3 = DIGUY_CHARACTER_TYPE_MAP_FIELD_UNKNOWN, const
    diguyCharacterTypeMapField field4 = DIGUY_CHARACTER_TYPE_MAP_FIELD_UNKNOWN, const
    diguyCharacterTypeMapField field5 = DIGUY_CHARACTER_TYPE_MAP_FIELD_UNKNOWN, const
    diguyCharacterTypeMapField field6 = DIGUY_CHARACTER_TYPE_MAP_FIELD_UNKNOWN, const
    diguyCharacterTypeMapField field7 = DIGUY_CHARACTER_TYPE_MAP_FIELD_UNKNOWN, const
    diguyCharacterTypeMapField field8 = DIGUY_CHARACTER_TYPE_MAP_FIELD_UNKNOWN, const
diguyCharacterTypeMap* get_nearest_character_type_map_using_field_strings(
    const char* field1_string, const char* field1_value, const char* field1_alterate
    const char* field2_string = NULL, const char* field2_value = NULL, const char* fi
    const char* field3_string = NULL, const char* field3_value = NULL, const char* fi
    const char* field4_string = NULL, const char* field4_value = NULL, const char* fi
    const char* field5_string = NULL, const char* field5_value = NULL, const char* fi
    const char* field6_string = NULL, const char* field6_value = NULL, const char* fi
    const char* field7_string = NULL, const char* field7_value = NULL, const char* fi
    const char* field8_string = NULL, const char* field8_value = NULL, const char* fi
diguyCharacterTypeMap* get_nearest_character_type_map_id_match(const char* type_map_id
int get_num_character_type_map_entries();
diguyCharacterTypeMap* get_character_type_map_entry_at_index(int index);
int get_num_type_map_character_classes();
const char* get_type_map_character_class_at_index(int index);
int get_num_type_map_categories(const char* character_class);
const char* get_type_map_category_at_index(const char* character_class,
    int index);
int get_num_type_map_subcategories(const char* character_class,
    const char* category);
const char* get_type_map_subcategory_at_index(const char* character_class,
    const char* category,
    int index);
int get_num_signals();
diguySignal* get_signal_at_index(int index);
diguySignal* find_signal(const char* name);
diguySignal* create_signal(const char* name);
int destroy_signal(diguySignal* signal);
void reset_signals();
int signal_palette_hide_all(const char* filter = NULL);
int signal_palette_show_all(const char* filter = NULL);
int get_num_sounds();
diguySound* get_sound_at_index(int index);
diguySound* find_sound(const char* name);
diguySound* create_sound(const char* name,
    const char* sound_filename = NULL,
    const char* phoneme_filename = NULL);
int destroy_sound(diguySound* sound);
int preload_sound(diguySound* sound);
int stop_all_sounds(float rampdown_time = 0.0f);
int play_3d_sound(const char* name,
    float x, float y, float z,
    float gain,
    int save_event);
int get_num_groups();
diguyCharacterGroup* get_group_at_index(int index);
diguyCharacterGroup* find_group(const char* name);
diguyCharacterGroup* create_group(const char* name);
diguyCharacterGroup* find_or_create_group(const char* name);
int destroy_group(diguyCharacterGroup* group);
int get_num_sensor_regions();
diguySensorRegion* get_sensor_region_at_index(int index);
diguySensorRegion* find_sensor_region(const char* name);
int get_num_scene_objects();

```

```

diguySceneObject* get_scene_object_at_index(int index);
diguySceneObject* find_scene_object(const char* name);
void set_scene_objects_enabled(int scene_objects_enabled);
int get_scene_objects_enabled();
diguySceneObject* create_scene_object(const char* name,
    const char* filename);
diguySceneObject* create_user_defined_scene_object(const char* name);
int destroy_scene_object(diguySceneObject* scene_object);
diguyView* get_primary_view();
int get_num_secondary_views();
diguyView* get_secondary_view_at_index(int index);
diguyView* find_view(const char* name);
diguyViewCamera* get_scenario_camera();
diguyViewCamera* find_camera(const char* name);
void set_scenario_camera_affected_by_script_events(int flag);
int get_scenario_camera_affected_by_script_events();
void set_scenario_camera_affected_by_script_events_when_iguy_active(int flag);
int get_scenario_camera_affected_by_script_events_when_iguy_active();
int get_num_camera_settings();
diguyViewCameraSettings* get_camera_settings_at_index(int index);
diguyViewCameraSettings* find_camera_settings(const char* settings_name);
int load_camera_settings(const char* settings_name, int update_current_camera = 1);
void set_render_camera(diguyViewCamera* camera);
diguyViewFog* get_scenario_fog();
diguyViewFog* find_fog(const char* name);
diguyViewFog* find_or_create_fog(const char* name);
void load_current_fog(const char* name);
int get_num_fog_settings();
diguyViewFogSettings* get_fog_settings_at_index(int index);
diguyViewFogSettings* find_fog_settings(const char* settings_name);
int load_fog_settings(const char* settings_name);
diguyViewLight* get_scenario_light(int i = 0);
diguyViewLight* find_light(const char* name);
int get_num_light_settings();
diguyViewLightSettings* get_light_settings_at_index(int index);
diguyViewLightSettings* find_light_settings(const char* settings_name);
int load_light_settings(const char* settings_name, int light_num = 0);
int get_num_info_popups();
diguyInfoPopup* get_info_popup_at_index(int index);
diguyInfoPopup* find_info_popup(const char* name);
int set_info_popup_default_encoding(const char* info_popup_default_encoding);
const char* get_info_popup_default_encoding();
int get_num_variables();
diguyVariable* get_variable_at_index(int index);
diguyVariable* find_variable(const char* name);
diguyVariable* find_or_create_variable(const char* name);
int destroy_variable(diguyVariable* variable);
int get_num_face_expressions();
diguyCharacterFaceExpression* get_face_expression_at_index(int index);
diguyCharacterFaceExpression* find_face_expression(const char* name);
diguyCharacterFaceExpression* create_face_expression(const char* name,
    const char* target_actor = "exface");
int destroy_face_expression(diguyCharacterFaceExpression* face_expression);
int get_num_chain_settings();
diguyChainSettings* get_chain_settings_at_index(int index);
diguyChainSettings* find_chain_settings(const char* name);
diguyChainSettings* create_chain_settings(const char* name);
int destroy_chain_settings(diguyChainSettings* chain_settings);
enum {
    CALLBACK_ID_CREATE = 1,
    CALLBACK_ID_DESTROY,

```

```

    CALLBACK_ID_RESET,
    CALLBACK_ID_WAIT_CURSOR_SHOW,
    CALLBACK_ID_WAIT_CURSOR_HIDE,
    CALLBACK_ID_LOAD,
    CALLBACK_ID_SAVE,
    CALLBACK_ID_LOAD_SCENARIO_FILE,
    CALLBACK_ID_SAVE_SCENARIO_FILE,
    CALLBACK_ID_SCENE_OBJECT_IMPACT,
    CALLBACK_ID_TIMED_EVENT,
    CALLBACK_ID_MANUALLY_INVOKED,
    CALLBACK_ID_POST_DRAW,
    CALLBACK_ID_PLAYBACK_MODE_CHANGED,
    CALLBACK_ID_INPUT_MODE_CHANGED,
    CALLBACK_ID_RENDER_MODE_CHANGED,
    CALLBACK_ID_PRE_REINITIALIZE,
    CALLBACK_ID_POST_REINITIALIZE,
    CALLBACK_ID_POST_LOAD_CHECKPOINT
};
#ifdef CPLUSPLUS_ONLY
int add_callback(int callback_id,
    diguyScenarioCallback* callback,
    void* callback_params = 0,
    void* callback_user_data = 0,
    int remove_on_scenario_load = 1);
int remove_callback(int callback_id,
    diguyScenarioCallback* callback);
int remove_callback_with_user_data(int callback_id,
    void* callback_user_data);
int add_default_character_callback(int callback_id,
    diguyCharacterCallback* callback,
    void* callback_params,
    void* callback_user_data,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
int remove_default_character_callback(int callback_id,
    diguyCharacterCallback* callback);
int remove_default_character_callback_with_user_data(int callback_id,
    void* callback_user_data);
int add_default_path_callback(int callback_id,
    diguyCharacterPathCallback* callback,
    void* callback_params,
    void* callback_user_data,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
int remove_default_path_callback(int callback_id,
    diguyCharacterPathCallback* callback);
int remove_default_path_callback_with_user_data(int callback_id,
    void* callback_user_data);
int add_default_sensor_region_callback(int callback_id,
    diguySensorRegionCallback* callback,
    void* callback_params,
    void* callback_user_data,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
int remove_default_sensor_region_callback(int callback_id,
    diguySensorRegionCallback* callback);
int remove_default_sensor_region_callback_with_user_data(int callback_id,
    void* callback_user_data);
int add_default_signal_callback(int callback_id,
    diguySignalCallback* callback,
    void* callback_params,

```

```

        void* callback_user_data,
        int add_to_existing_objects_flag = 0,
        int remove_on_scenario_load = 1);
int remove_default_signal_callback(int callback_id,
    diguySignalCallback* callback);
int remove_default_signal_callback_with_user_data(int callback_id,
    void* callback_user_data);
int add_default_variable_callback(int callback_id,
    diguyVariableCallback* callback,
    void* callback_params,
    void* callback_user_data,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
int remove_default_variable_callback(int callback_id,
    diguyVariableCallback* callback);
int remove_default_variable_callback_with_user_data(int callback_id,
    void* callback_user_data);
int add_default_view_callback(int callback_id,
    diguyViewCallback* callback,
    void* callback_params,
    void* callback_user_data,
    int add_to_existing_objects_flag = 1,
    int remove_on_scenario_load = 1);
int remove_default_view_callback(int callback_id,
    diguyViewCallback* callback);
int remove_default_view_callback_with_user_data(int callback_id,
    void* callback_user_data);
int add_default_camera_callback(int callback_id,
    diguyViewCameraCallback* callback,
    void* callback_params,
    void* callback_user_data,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
int remove_default_camera_callback(int callback_id,
    diguyViewCameraCallback* callback);
int remove_default_camera_callback_with_user_data(int callback_id,
    void* callback_user_data);
int add_default_fog_callback(int callback_id,
    diguyViewFogCallback* callback,
    void* callback_params,
    void* callback_user_data,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
int remove_default_fog_callback(int callback_id,
    diguyViewFogCallback* callback);
int remove_default_fog_callback_with_user_data(int callback_id,
    void* callback_user_data);
int add_default_light_callback(int callback_id,
    diguyViewLightCallback* callback,
    void* callback_params,
    void* callback_user_data,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
int remove_default_light_callback(int callback_id,
    diguyViewLightCallback* callback);
int remove_default_light_callback_with_user_data(int callback_id,
    void* callback_user_data);
#endif
void manually_invoke_callbacks_now(int callback_id);
int add_callback_script(int callback_id,
    const char* callback_script,

```



```

    const char* callback_script_type = NULL,
    int remove_on_scenario_load = 1);
int remove_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int add_default_character_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
int remove_default_character_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int add_default_sensor_region_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
int remove_default_sensor_region_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int add_default_signal_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
int remove_default_signal_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int add_default_variable_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
int remove_default_variable_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int add_default_view_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
int remove_default_view_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int add_default_camera_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
int remove_default_camera_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int add_default_fog_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
int remove_default_fog_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);

```

```

int add_default_light_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
int remove_default_light_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int map_event_handler_to_callback_id(int callback_id,
    const char* handler_name);
int unmap_event_handler_from_callback_id(int callback_id,
    const char* handler_name,
    int unmap_all_matches = 0);
diguyCallbackReturn manually_invoke_event_handler(const char* handler_name);
#ifdef CPLUSPLUS_ONLY
int register_character_event_handler(const char* handler_name,
    diguyCharacterCallback* callback,
    void* callback_user_data = 0);
#endif
int register_character_event_handler_from_library(const char* handler_name,
    const char* library_name,
    const char* function_name,
    void* callback_user_data = 0);
int register_character_event_handler_script(const char* handler_name,
    const char* handler_script,
    const char* handler_script_type = 0);
int has_registered_character_event_handler(const char* handler_name);
int unregister_character_event_handler(const char* handler_name);
#ifdef CPLUSPLUS_ONLY
int register_signal_event_handler(const char* handler_name,
    diguySignalCallback* callback,
    void* callback_user_data = 0);
#endif
int register_signal_event_handler_from_library(const char* handler_name,
    const char* library_name,
    const char* function_name,
    void* callback_user_data = 0);
int register_signal_event_handler_script(const char* handler_name,
    const char* handler_script,
    const char* handler_script_type = 0);
int has_registered_signal_event_handler(const char* handler_name);
int unregister_signal_event_handler(const char* handler_name);
#ifdef CPLUSPLUS_ONLY
int register_sensor_region_event_handler(const char* handler_name,
    diguySensorRegionCallback* callback,
    void* callback_user_data = 0);
#endif
int register_sensor_region_event_handler_from_library(const char* handler_name,
    const char* library_name,
    const char* function_name,
    void* callback_user_data = 0);
int register_sensor_region_event_handler_script(const char* handler_name,
    const char* handler_script,
    const char* handler_script_type = 0);
int has_registered_sensor_region_event_handler(const char* handler_name);
int unregister_sensor_region_event_handler(const char* handler_name);
#ifdef CPLUSPLUS_ONLY
int register_variable_event_handler(const char* handler_name,
    diguyVariableCallback* callback,
    void* callback_user_data = 0);
#endif

```



```

int register_variable_event_handler_from_library(const char* handler_name,
const char* library_name,
const char* function_name,
void* callback_user_data = 0);
int register_variable_event_handler_script(const char* handler_name,
const char* handler_script,
const char* handler_script_type = 0);
int has_registered_variable_event_handler(const char* handler_name);
int unregister_variable_event_handler(const char* handler_name);
int save_review_data(const char* filename);
int get_loaded_from_review_data();
int set_history_type(diguyHistoryType history_type);
diguyHistoryType get_history_type();
void set_overall_history_type(diguyHistoryType overall_history_type);
diguyHistoryType get_overall_history_type();
int get_replaying_history();
void set_history_max_sound_instances(int num_sound_instances);
void set_automatic_graphics_lod_switching(int enable_graphics_lod_switching);
int get_automatic_graphics_lod_switching();
#ifdef CPLUSPLUS_ONLY
void set_default_lod_ranges(const char* character_type, float* ranges);
#endif
void set_scale_graphics_lod_ranges_from_view_settings(int do_scale);
int get_scale_graphics_lod_ranges_from_view_settings();
void set_automatic_motion_lod_switching(int enable_motion_lod_switching);
int get_automatic_motion_lod_switching();
int eval_script(const char* script_text,
int save_as_transient = 0,
const char* script_type = NULL,
const char* script_source = NULL);
int eval_script_file(const char* filename,
const char* script_type = NULL);
int trigger_script(const char* script_name);
void trigger_script_delayed(const char* script_name, float t_delay);
int trigger_decision(const char* decision_name);
void trigger_decision_delayed(const char* decision_name, float t_delay);
void push_transient_script_event(float t, const char* script_text, const char* script_type);
void clear_transient_script_events();
int save_transient_script_events(const char* filename);
int load_transient_script_events(const char* filename);
int load_package(const char* filename,
const char* language = NULL,
int warn_if_not_found = 1);
int load_package_with_dependancy(const char* filename,
const char* depends_on_package,
const char* language = NULL,
int warn_if_not_found = 1);
int get_character_type_num_postures(const char* character_type);
diguyMotionPosture get_character_type_posture_at_index(const char* character_type,
int index);
diguyMotionPosture get_character_type_action_posture(const char* character_type,
const char* action_name);
int get_character_type_num_variants(const char* character_type);
diguyMotionVariant get_character_type_variant_at_index(const char* character_type,
int index);
diguyMotionVariant get_character_type_action_primary_variant(const char* character_type,
const char* action_name);
int get_character_type_action_num_variants(const char* character_type,
const char* action_name);
diguyMotionVariant get_character_type_action_variant_at_index(const char* character_type,
const char* action_name,

```

```

    int index);
int get_character_type_num_directions(const char* character_type);
diguyMotionDirection get_character_type_direction_at_index(const char* character_type,
    int index);
int get_num_interaction_machines();
diguyInteractionMachine* create_interaction_machine(const char* name);
void destroy_interaction_machine(diguyInteractionMachine* interaction_machine);
diguyInteractionMachine* get_interaction_machine_at_index(int index);
diguyInteractionMachine* find_interaction_machine(const char* name);
diguyInteractionMachine* get_active_interaction_machine();
diguyInteractionMachine* get_shared_interaction_machine();
int show_notification_with_shared_interaction_machine(const char* heading,
    const char* info,
    int pause_scenario = 0,
    const char* input_text = NULL,
    int ui_appearance = diguyInteractionMachine::UI_APPEARANCE_NOTIFICATION);
diguyIGuyController* get_iguy_controller();
diguyViewLabel* find_or_create_label(const char* name);
diguyViewLabel* find_label(const char* name);
int destroy_label(diguyViewLabel* label);
int get_num_labels();
diguyViewLabel* get_label_at_index(int index);
void reset_labels();
void send_label_to_front(diguyViewLabel* label);
void send_label_to_back(diguyViewLabel* label);
diguyViewButtonPanel* find_or_create_panel(const char* name,
    int horizontal = 0,
    int title = 1);
diguyViewButtonPanel* find_panel(const char* name);
int destroy_panel(diguyViewButtonPanel* panel);
int get_num_panels();
diguyViewButtonPanel* get_panel_at_index(int index);
int get_num_minds();
const char* get_mind_name_at_index(int index);
int is_valid_character_pointer(diguyCharacter* character);
int is_valid_character_group_pointer(diguyCharacterGroup* character_group);
diguyCrowd* create_crowd(const char* name,
    int make_companion_of_all = 1);
int destroy_crowd(diguyCrowd* crowd,
    int also_destroy_crowd_members = 0,
    int also_destroy_path_shapes = 0,
    int delay_destruction = 0);
int get_num_crowds();
diguyCrowd* get_crowd_at_index(int index);
const char* get_unique_crowd_name(const char* base_name);
diguyCrowd* find_crowd(const char* crowd_name);
diguyCrowd* find_or_create_crowd(const char* crowd_name,
    int make_companion_of_all = 1);
void set_create_network_crowds(int do_create);
int get_create_network_crowds();
diguyCrowdProfile* create_crowd_profile(const char* name);
int destroy_crowd_profile(diguyCrowdProfile* crowd_profile);
int get_num_crowd_profiles();
diguyCrowdProfile* get_crowd_profile_at_index(int index);
diguyCrowdProfile* find_crowd_profile(const char* name);
diguyLoadManager* get_character_load_manager();
int force_octtree_generation();
void add_character_to_octtree(diguyCharacter* character,
    int use_bounding_box_only = 0);
void remove_character_from_octtree(diguyCharacter* character);
int preload_octtree(const char* character_type, const char* appearance);

```

```

    int check_visibility(float x1, float y1, float z1,
        float x2, float y2, float z2);
#ifdef CPLUSPLUS_ONLY
    int set_default_character_altitude_function(diguyAltitudeFunction* altitude_function);
    void set_altitude_function(diguyScenarioAltitudeFunction* altitude_function);
    diguyScenarioAltitudeFunction* get_altitude_function();
#endif
    void set_altitude_max_climb(float altitude_max_climb);
    float get_altitude_max_climb();
    void set_altitude_max_drop(float altitude_max_drop);
    float get_altitude_max_drop();
    static void set_default_intersection_function(diguyScenarioIntersectionFunction* intersection_function);
    void set_intersection_function(diguyScenarioIntersectionFunction* intersection_function);
    int trigger_detonation(const char* munition_name,
        float x, float y, float z,
        const char* attacker_name = NULL,
        float radius_override = -1.0f,
        int broadcast_on_network = 1);
    diguyImpact* find_character_at_screen_coords(diguyView* view,
        float sx,
        float sy);
    diguyImpact* get_impact_at_screen_coords(diguyView* view,
        float sx,
        float sy,
        int use_octtree = 0);
    diguyImpact* get_intersection_at_screen_coords(diguyView* view,
        float sx,
        float sy,
        int use_octtree = 0);
    int intersect_static_geometry(
        float origin_x, float origin_y, float origin_z,
        float dir_x, float dir_y, float dir_z,
        float max_distance_to_test,
        float* intersection_x, float* intersection_y, float* intersection_z,
        float* normal_x = NULL, float* normal_y = NULL, float* normal_z = NULL,
        int use_callback = 1);
    float ground_clamp(float x, float y, float z,
        int* valid = NULL,
        int use_altitude_function = 1,
        int use_octtree = 0);
#ifdef CPLUSPLUS_ONLY
    int set_fire_weapon_intersection_function(diguyScenarioFindImpactOnLineFunction* find_impact_on_line_function);
    static void set_default_fire_weapon_intersection_function(diguyScenarioFindImpactOnLineFunction* find_impact_on_line_function);
#endif
    diguyImpact* map_color_to_impact(char r, char g, char b);
    diguyImpact* get_last_env_impact_info();
    void set_last_impact_been_processed();
    void set_last_env_impact_info(const diguyImpact&impact);
    void set_max_unprocessed_impacts(int max_unprocessed_impacts);
#ifdef CPLUSPLUS_ONLY
    int set_default_point_los_function(diguyPointLOSFunction* point_los_function);
    int set_default_character_los_function(diguyCharacterLOSFunction* character_los_function);
    int set_default_agent_feeler_function(diguyFeelerFunction* feeler_function);
#endif
    diguyIntersectionResult exp_intersect_geometry(
        float origin_x, float origin_y, float origin_z,
        float dir_x, float dir_y, float dir_z,
        float max_distance_to_test,
        int use_callback = 1);
    diguyRegion* find_or_create_region(const char* name);
    diguyRegion* find_region(const char* name);

```

```

diguyRegion* create_region(const char* name);
void destroy_region(diguyRegion* region);
int get_num_regions();
diguyRegion* get_region_at_index(int index);
int find_navigation_path(float x1, float y1, float z1,
    float x2, float y2, float z2,
    const char* via_region,
    diguyPathShape* path_shape,
    float cost_bias_for_preferred_regions = 0.25f,
    int preferred_subregions_mask = DIGUY_SUBREGION_MASK_NONE,
    float cost_bias_for_neutral_regions = 1.1f,
    int repulsed_regions_mask = DIGUY_SUBREGION_MASK_NONE,
    float cost_bias_for_repulsive_regions = 100.0f);
void set_max_distance_planning_multiplier(float mul);
float get_max_distance_planning_multiplier();
float get_time_spent_path_planning_this_frame();
void set_multithreaded_path_planning_enabled(int value);
int get_multithreaded_path_planning_enabled();
int get_num_formation();
diguyFormation* get_formation_at_index(int index);
diguyFormation* find_formation(const char* name);
diguyFormation* find_formation_of_size(int size);
diguyFormation* create_formation(const char* name);
diguyFormation* find_or_create_formation(const char* name);
int destroy_formation(diguyFormation* formation);
void set_action_bead_labels_visible(diguyVisibleFlag vflag);
void set_aim_trajectories_visible(diguyVisibleFlag vflag);
void set_author_selection_handles_visible(diguyVisibleFlag vflag);
void set_bead_array_visible(diguyVisibleFlag vflag);
void set_character_labels_visible(diguyVisibleFlag vflag);
void set_character_visible(diguyVisibleFlag vflag);
void set_crowd_behavior_visible(diguyVisibleFlag flag);
void set_crowd_feelers_visible(diguyVisibleFlag flag);
void set_crowd_influence_visible(diguyVisibleFlag flag);
void set_crowd_regions_visible(diguyVisibleFlag flag);
void set_light_array_visible(diguyVisibleFlag vflag);
void set_lua_objects_visible(diguyVisibleFlag flag);
void set_mesh_region_array_visible(diguyVisibleFlag vflag);
void set_sensor_region_array_visible(diguyVisibleFlag vflag);
void set_spath_visible(diguyVisibleFlag vflag);
void set_waypoint_array_visible(diguyVisibleFlag vflag);
diguyVisibleFlag get_action_bead_labels_visible();
diguyVisibleFlag get_aim_trajectories_visible();
diguyVisibleFlag get_author_selection_handles_visible();
diguyVisibleFlag get_bead_array_visible();
diguyVisibleFlag get_character_labels_visible();
diguyVisibleFlag get_character_visible();
diguyVisibleFlag get_crowd_behavior_visible();
diguyVisibleFlag get_crowd_feelers_visible();
diguyVisibleFlag get_crowd_influence_visible();
diguyVisibleFlag get_crowd_regions_visible();
diguyVisibleFlag get_light_array_visible();
diguyVisibleFlag get_lua_objects_visible();
diguyVisibleFlag get_mesh_region_array_visible();
diguyVisibleFlag get_sensor_region_array_visible();
diguyVisibleFlag get_spath_visible();
diguyVisibleFlag get_waypoint_array_visible();
void set_hide_author_selection_handles_during_play(int visible);
int get_hide_author_selection_handles_during_play();
int get_weapon_fire_effects_enabled();

```

```

int set_weapon_fire_effects_enabled(int enabled);

int get_num_active_fire_effects();

int get_weapon_fire_effect_data(int index, float * radius,
    float * position_x, float * position_y, float * position_z,
    float * color_r, float * color_g, float * color_b,
    float * falloff_r,
    float * falloff_rsq);
int set_current_character(diguyCharacter* current_character);
diguyCharacter* get_current_character();
int set_current_crowd(diguyCrowd* current_crowd);
diguyCrowd* get_current_crowd();
int set_current_crowd_profile(diguyCrowdProfile* current_profile);
int set_current_crowd_profile_by_name(const char* crowd_profile_name);
diguyCrowdProfile* get_current_crowd_profile();
int set_current_region(diguyRegion* current_region);
diguyRegion* get_current_region();
void set_particle_module_disabled(int disable_particle_module);
int create_particle_system(const char* description_name,
    float x, float y, float z,
    int record_transient_event = 1,
    float duration = -1.0f);
int create_particle_system_with_orientation(const char* description_name,
    float x, float y, float z,
    float rz, float rx, float ry,
    int record_transient_event = 1,
    float duration = -1.0f);
int has_particle_description(const char* description_name);
void set_global_wind(float x, float y, float z);
const char* lua_get_object_field_as_string(const char* lua_object,
    const char* field_name);
const char* lua_evaluate_object_function(const char* lua_object,
    const char* function_name,
    const char* argument = NULL,
    int has_return_string = 0);
const char* lua_evaluate_object_function_2a(const char* lua_object,
    const char* function_name,
    const char* argument,
    const char* argument2,
    int has_return_string = 0);
const char* lua_evaluate_object_function_3a(const char* lua_object,
    const char* function_name,
    const char* argument,
    const char* argument2,
    const char* argument3,
    int has_return_string = 0);
const char* lua_evaluate_object_function_4a(const char* lua_object,
    const char* function_name,
    const char* argument,
    const char* argument2,
    const char* argument3,
    const char* argument4,
    int has_return_string = 0);
int lua_send_message_to_object(const char* lua_object,
    const char* sender,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);
#ifdef CPLUSPLUS_ONLY
void* get_lua_state();

```

```

int push_qt_pointer_to_lua(void* lua_state,
    const char* class_name,
    void* ptr);
const char* lua_evaluate_object_function_4a_flex(const char* lua_object,
    const char* function_name,
    const char* arg1_type, void* argument1,
    const char* arg2_type, void* argument2,
    const char* arg3_type, void* argument3,
    const char* arg4_type, void* argument4,
    int has_return_string = 0);
const char* lua_evaluate_global_function_4a_flex(const char* function_name,
    const char* arg_type, void* argument,
    const char* arg_type2, void* argument2,
    const char* arg_type3, void* argument3,
    const char* arg_type4, void* argument4,
    int has_return_string = 0);
#endif
int lua_log_printf(int notify_level, const char* string);
const char* dump_package_info(const char* package_name);
int get_eval_decisions_as_scripts();
void set_eval_decisions_as_scripts(int eval_decisions_as_scripts);
diguyImpact* get_impact_on_line(float from_x,
    float from_y,
    float from_z,
    float to_x,
    float to_y,
    float to_z);
void wait_cursor_push();
void wait_cursor_pop();
#ifdef CPLUSPLUS_ONLY
void set_graphics_api_node_ptr(void* node_ptr);
void* get_graphics_api_node_ptr();
#endif
void set_draw_character_shadow_disks(int draw_character_shadow_disks);
int get_draw_character_shadow_disks();
int get_shadow_map_size();
int set_shadow_map_size(int shadow_size);
void set_default_character_scene_graph_mask(unsigned long mask);
unsigned long get_default_character_scene_graph_mask();
void set_default_scene_object_scene_graph_mask(unsigned long mask);
unsigned long get_default_scene_object_scene_graph_mask();
void set_internal_int(int var, int val);
void set_internal_float(int var, float val);
int set_num_active_point_lights(int active_lights);
int get_num_active_point_lights();
int optimized_update(float t);
int set_world_bounds(float x_min, float y_min, float z_min,
    float x_max, float y_max, float z_max);
int get_world_bounds(float* x_min, float* y_min, float* z_min,
    float* x_max, float* y_max, float* z_max);
void set_checkpoint_frequency(float freq);
float get_checkpoint_frequency();
void set_checkpointing_enabled(int enable_checkpointing);
int get_checkpointing_enabled();
void set_initial_render_mode(const char* render_mode);
const char* get_initial_render_mode();
void set_current_render_mode(const char* render_mode);
const char* derive_render_mode_from_light_settings(int set_current_to_derived = 1);
const char* get_current_render_mode();
void set_default_character_shader(const char* render_mode,
    const char* shader_name,

```

```

    int update_existing_characters = 1);
const char* get_default_character_shader(const char* render_mode);
void set_default_scene_object_shader(const char* render_mode,
    const char* shader_name,
    int update_existing_scene_objects = 1);
const char* get_default_scene_object_shader(const char* render_mode);
void set_default_particle_system_shader(const char* render_mode,
    const char* shader_name,
    int update_existing_particle_systems = 1);
const char* get_default_particle_system_shader(const char* render_mode);
int facefx_mount_animset(const char* actor_name, const char* file_name);
int set_vehicle_smoothing_enabled(int enable_smoothing);
int get_vehicle_smoothing_enabled();
int set_character_culling_distance(float dist);
float get_character_culling_distance();
int set_vehicle_culling_distance(float dist);
float get_vehicle_culling_distance();
int set_prop_culling_distance(float dist);
float get_prop_culling_distance();
#ifdef CPLUSPLUS_ONLY
    static int add_default_callback(int callback_id,
        diguyScenarioCallback* callback,
        void* callback_params = 0,
        void* callback_user_data = 0);
    static int remove_default_callback(int callback_id,
        diguyScenarioCallback* callback);
#endif
static int remove_default_callback_with_user_data(int callback_id,
    void* callback_user_data);
static int add_default_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type);
static int remove_default_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type);
const char* get_character_type_class(const char* character_type);
int merge_asset(const char* string);
void print_to_log(int notify_level, const char* string);
#ifdef CPLUSPLUS_ONLY
    diguyCharacter* create_pending_reflected_character(const char* name,
        const char* character_type,
        const char* appearance = NULL);
int set_network_translation(float x, float y, float z);
int get_network_translation(float* x, float* y, float* z);
void set_project_filename(const char* project_filename);
const char* get_project_filename();

```


General Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.2 function diguyScenario::get_current_scenario

Prototype:

```
static diguyScenario* get_current_scenario();
```

Description:

This static function returns a pointer to the current scenario object. All functions below can then be called through this pointer. This is the same pointer returned by the call `diguy_create_scenario()`.

lua scripts get the current scenario through the variable `this_scenario`.

Returns:

a pointer of type `diguyScenario`

C++ Example:

```
diguyScenario* sc = diguyScenario::get_current_scenario();
sc->reset();
```


Lua Example:

```
this_scenario:reset();
```

5.74.3 function diguyScenario::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the object

5.74.4 function diguyScenario::get_type_name

Prototype:

```
const char* get_type_name();
```

Description:

Returns the type name of the object. This pointer will never be NULL.

Returns:

type name of the object

5.74.5 function diguyScenario::get_description

Prototype:

```
const char* get_description();
```

Returns:

the scenario description as specified in the scenario info page

Time Control and Playback Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.6 function diguyScenario::update

Prototype:

```
int update(float t);
```

Description:

Sets the desired simulation time of the scenario.

Some update operations will use the simulation desired time directly.

Many other scenario operations -- notably character position and pose updates -- will not happen until the desired *t* passes each subsequent tick *dt* threshold. See [set_tick_dt\(\)](#).

This function sets both elapsed simulation time and elapsed realtime to the same value. This is usually desired in DI-Guy applications. To pass separate values for simulation time and realtime time, call [update2\(\)](#).

Time can be run backwards. This happens if the passed *t* is lower than any *t* previously passed to the [update\(\)](#) function; the scenario remembers the highest *t* it has reached. If *t* is set to less than the highest *t* reached, characters

that have history data will replay what they were doing at the specified time. Characters that do not have history will remain motionless until *t* advances past the previous highest *t*.

Scenario time will never go below 0.

If *t* reaches the scenario tout ("T Out") time (as set by `set_tout()`, a number of things may happen. The scenario may:

- ◇ simply stop at the tout time
- ◇ continue past the tout time
- ◇ rewind to time 0 and play back recorded history
- ◇ reset and re-simulate the scenario from time 0

The default tout time is 60000 for DI-Guy applications. The tout can be enabled and disabled with `set_tout_enabled()`.

This function cannot explicitly be called for DI-Guy Scenario, but the effects of *t* reaching tout during scenario simulation are outlined here anyway.

For a DI-Guy application:

- ◇ If the scenario tout is enabled the scenario will stop.
- ◇ If the scenario tout is disabled the scenario will continue indefinitely, regardless of scenario tout.

Note that no playback looping happens in a DI-Guy application if tout is reached. Only in DI-Guy Scenario does the loop flag have an effect.

For DI-Guy Scenario:

- ◇ If the loop flag is not set (see `set_playback_loop()`) or a movie is being recorded, the scenario will stop.
- ◇ If the loop flag is set, and any character in the scenario has history data (see the History Functions for `diguyCharacter`), the scenario will be rewound to 0 and play back history data.
- ◇ If the loop flag is set, but no characters in the scenario have history data, the scenario will be reset to 0 and re-simulate from there.

Arguments:

t desired time in seconds

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++

5.74.7 function `diguyScenario::update2`

Prototype:

```
int update2(float t, float realtime_t);
```

Description:

Similar to `update( )`, but the realtime time can be explicitly passed in.

Arguments:

`t` desired simulation time in seconds
`realtime_t` elapsed real time in seconds

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.74.8 function `diguyScenario::set_playback_loop`

Prototype:

```
int set_playback_loop(int playback_loop);
```

Description:

This function sets whether the scenario is designed to loop at tout. The default is for scenarios to loop.

Note that this has an effect only in DI-Guy Scenario. DI-Guy applications do not use this flag.

See `get_playback_loop( )`.

Arguments:

`playback_loop` pass 1 to have scenario loop at tout, 0 to not

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.74.9 function diguyScenario::set_tick_dt

Prototype:

```
int set_tick_dt(float tick_dt);
```

Description:

Scenarios progress at discrete time steps, called "ticks". Each tick is of the same duration. This function sets the amount of time each tick takes.

Because changing the tick interval of a scenario changes when event beads occur, any change may introduce subtle differences in scenario playback.

Note that loading a .dss file will change the tick dt, as tick dt is set on a per scenario basis.

The default tick dt of a new scenario is 0.03125 seconds, or 32 frames per second.

Arguments:

`tick_dt` duration, in seconds, of each scenario tick

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

C++ Example:

```
diguyScenario* sc = diguy_create_scenario();
if (sc)
{
    sc->load("my_scenario.dss");
    sc->set_tick_dt(0.033333); // 30 fps
}
```

5.74.10 function `diguyScenario::get_t`

Prototype:

```
float get_t();
```

Description:

This function returns the time in seconds that the scenario is currently at. See [update\(\)](#).

Note that this value will not necessarily track the `t` passed in the [update\(\)](#) function. This `t` advances only when the scenario's current tick changes. See [set_tick_dt\(\)](#) and [update\(\)](#).

Returns:

current time in seconds

5.74.11 function diguyScenario::get_realtime_t**Prototype:**

```
float get_realtime_t();
```

Description:

This function returns the realtime time in seconds that the scenario is currently at. This may be different than `get_desired_t()` if the scenario is stopped or the time passed in `update()` is not tracking realtime.

Returns:

current time in seconds

5.74.12 function diguyScenario::get_desired_t**Prototype:**

```
float get_desired_t();
```

Description:

This function returns the desired time in seconds that the scenario is currently at. This may be different than `get_t()`, which advances only when the scenario's current tick changes.

Returns:

current time in seconds

5.74.13 function diguyScenario::set_tout

Prototype:

```
int set_tout(float tout);
```

Description:

This function sets the tout ("T Out") time of the scenario. What will happen when the scenario reaches this time depends on a number of factors. See [update\(\)](#) for more information.

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.74.14 function diguyScenario::get_tout

Prototype:

```
float get_tout();
```

Description:

This function returns the tout time of the scenario. This is the time at which a scenario is designed to end.

See [set_tout\(\)](#) and [update\(\)](#).

Returns:

tout of scenario

5.74.15 function diguyScenario::set_tout_enabled**Prototype:**

```
void set_tout_enabled(int tout_enabled);
```

Description:

This function sets what will happen when the scenario reaches its tout time. If set to 0, the scenario will ignore the tout time and continue indefinitely.

See [update\(\)](#) for more information.

5.74.16 function diguyScenario::get_tout_enabled**Prototype:**

```
int get_tout_enabled();
```

Description:

This function returns the whether tout is enabled.

See [set_tout_enabled\(\)](#) and [update\(\)](#).

Returns:

1 if tout enabled, 0 if not

5.74.17 function diguyScenario::get_playback_loop**Prototype:**

```
int get_playback_loop();
```

Description:

This function returns whether the scenario is designed to loop when the scenario's tout time is reached.

See [update\(\)](#) for more information about looping.

Returns:

0 if no loop desired; 1 if loop desired

5.74.18 function diguyScenario::get_tick_dt**Prototype:**

```
float get_tick_dt();
```

Description:

This function returns the most recent setting of [set_tick_dt\(\)](#).

5.74.19 function diguyScenario::get_ticks_can_be_dropped**Prototype:**

```
int get_ticks_can_be_dropped();
```

Returns:

This function returns the most recent setting of `set_ticks_can_be_dropped()`.

5.74.20 function diguyScenario::set_ticks_can_be_dropped**Prototype:**

```
void set_ticks_can_be_dropped(int ticks_can_be_dropped);
```

Description:

This function sets whether a scenario can "drop" ticks in order to catch up if it falls behind.

Scenario updates progress in discrete time steps called ticks (see `set_tick_dt()` and `update()`). If the difference in times specified in subsequent calls to `update()` is greater than the tick dt, which can happen if calls to `update()` are taking too long because the scenario is too complex or if the caller is attempting to "fast forward" to a particular time in the scenario, the scenario may skip processing of intermediate ticks in order to catch up.

The scenario will not consider itself behind until the number of ticks covered by a particular call to `update()` is equal to or greater than the value set by `set_max_ticks_behind_until_drop()`.

Dropping ticks, while it improves performance, may have unintended side effects, including:

- ◇ event beads of characters on paths that fall on the dropped ticks -- including aim beads, decision beads, gaze beads, and script beads -- will not be evaluated and will therefore have no effect on the scenario
- ◇ review playback of characters (see `diguyCharacter::set_history_type( )`) over the dropped intervals will be choppy

Not dropping ticks also has potential side effects:

- ◇ if a scenario is falling behind more and more on each call to `update( )`, it may never be able to catch up and performance of DI-Guy will increasingly degrade

The default setting is that ticks can be dropped.

Arguments:

`ticks_can_be_dropped` pass 1 for ticks to be dropped; else pass 0

5.74.21 function `diguyScenario::get_max_ticks_behind_until_drop`

Prototype:

```
long get_max_ticks_behind_until_drop();
```

Returns:

This function returns the most recent setting of `set_max_ticks_behind_until_drop( )`.

5.74.22 function diguyScenario::set_max_ticks_behind_until_drop

Prototype:

```
void set_max_ticks_behind_until_drop(long max_ticks_behind_until_drop);
```

Description:

This function sets how many ticks a scenario must be behind before it starts dropping ticks in order to catch up.

For a discussion of how a scenario may fall behind, see [set_ticks_can_be_dropped\(\)](#).

Note that if ticks cannot be dropped, (e.g., 0 is passed to [set_ticks_can_be_dropped\(\)](#)) this function will have no effect.

The default value is 2.

Arguments:

`max_ticks_behind_until_drop` how many ticks a scenario must be behind until it drops ticks

5.74.23 function diguyScenario::set_playback_mode

Prototype:

```
int set_playback_mode(diguyScenarioPlaybackMode playback_mode);
```

Description:

This function sets the current playback mode of DI-Guy Scenario. It currently will have no effect in applications using the DI-Guy API.

Arguments:

`playback_mode` new playback mode

Returns:

0 on success, -1 on failure

5.74.24 function diguyScenario::set_playback_mode_stop**Prototype:**

```
int set_playback_mode_stop();
```

Description:

This function is shorthand for the function call `set_playback_mode(DIGUY_SCENARIO_PLAYBACK_MODE_STOP)`. It is useful for pausing the scenario in decisions and decision beads.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.74.25 function diguyScenario::set_playback_mode_play**Prototype:**

```
int set_playback_mode_play();
```

Description:

This function is shorthand for the function call `set_playback_mode(DIGUY_SCENARIO_PLAYBACK_MODE_PLAY)`. It is useful for resuming a paused scenario in decisions and decision beads.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.74.26 function diguyScenario::get_playback_mode

Prototype:

```
diguyScenarioPlaybackMode get_playback_mode();
```

Returns:

current playback mode as set by [set_playback_mode\(\)](#) or the DI-Guy Scenario user interface

5.74.27 function diguyScenario::derive_playback_mode_dt

Prototype:

```
float derive_playback_mode_dt(float dt_in);
```

Description:

This function derives a dt based on an input dt (usually derived from change in realtime from previous to current frame) and the scenario's current playback mode. The resulting dt can then be used to advance the time passed to [diguyScenario::update\(\)](#) so that it emulates the effect of the VCR buttons in DI-Guy Scenario.

This function does not change the scenario's current time or playback mode.

Examples based on a dt_in of 0.03 seconds:

```
◇ DIGUY_SCENARIO_PLAYBACK_MODE_STOP: derived_dt =
  0.0
◇ DIGUY_SCENARIO_PLAYBACK_MODE_PLAY: derived_dt =
  0.03
```


- ◇ DIGUY_SCENARIO_PLAYBACK_MODE_PLAY_REVERSE:
derived_dt = -0.03
- ◇ DIGUY_SCENARIO_PLAYBACK_MODE_FF: derived_dt = 0.12
- ◇ DIGUY_SCENARIO_PLAYBACK_MODE_FF_REVERSE:
derived_dt = -0.12
- ◇ DIGUY_SCENARIO_PLAYBACK_MODE_SS: derived_dt = frame
dt
- ◇ DIGUY_SCENARIO_PLAYBACK_MODE_SS_REVERSE:
derived_dt = - frame dt
- ◇ DIGUY_SCENARIO_PLAYBACK_MODE_REWIND: derived_dt
= - scenario t

Note that the playback mode should be set to [DIGUY_SCENARIO_PLAYBACK_MODE_STOP](#) after time advances in either of the single step ("SS") modes or the rewind mode.

Arguments:

dt_in delta time in seconds from advancing simulation time

5.74.28 function diguyScenario::reset

Prototype:

```
int reset();
```

Description:

This function resets all scenario characters, signals, variables, and all other scenario objects to their original state. Scenario time will be set to 0.

Returns:

0 on success, -1 on failure

Load and Save Functions

5.74.29 function diguyScenario::load

Prototype:

```
int load(const char* filename);
```

Description:

This function loads the specified scenario. Any previously loaded scenario objects will be deleted. To merge a scenario into the existing scenario (load without deleting existing objects), use the call [merge\(\)](#).

The passed filename can be either a ".dss" file, which is a DI-Guy Scenario file, or a ".dsr" file, which is a DI-Guy Review file.

Once the file has been loaded an implicit [reset\(\)](#) will occur.

Arguments:

<code>filename</code>	filename of scenario to be loaded

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.74.30 function diguyScenario::save

Prototype:

```
int save();
```

Description:

Saves the scenario into its current filename. This filename will be the most recent of:

- ◇ the filename used in a `load( )` function call
- ◇ the filename used in a `save_as( )` function call

The filename that will be used can be retrieved using the `get_filename( )` or `get_filename_without_directory( )` functions.

Once saved the "unsaved" flag (as returned by `get_unsaved( )`) will be set to 0.

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.74.31 function diguyScenario::save_as

Prototype:

```
int save_as(const char* filename);
```

Description:

Similar to [save\(\)](#), but new filename is specified by the filename argument.

Arguments:

`filename` new filename of scenario

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.74.32 function diguyScenario::get_filename

Prototype:

```
const char* get_filename();
```

Description:

Returns the most recent of:

- ◇ the filename used in a `load( )` function call
- ◇ the filename used in a `save_as( )` function call

Returns:

filename of saved scenario

Callable From:

- ◇ C++

5.74.33 function diguyScenario::get_filename_without_directory

Prototype:

```
const char* get_filename_without_directory();
```

Description:

Similar to `get_filename( )`, but with any leading directories removed.

Callable From:

- ◇ C++

5.74.34 function diguyScenario::get_unsaved**Prototype:**

```
int get_unsaved();
```

Returns:

1 if scenario has been modified since save or load, else 0

Callable From:

◇ C++

5.74.35 function diguyScenario::set_unsaved**Prototype:**

```
void set_unsaved(int unsaved);
```

Description:

Manually sets the "unsaved" flag to the passed value.

Arguments:

unsaved	new unsaved status; 1 for unsaved, 0 for saved

Callable From:

◇ C++

Draw Functions

5.74.36 function diguyScenario::draw

Prototype:

```
int draw();
```

Description:

This function draws all visible elements of the loaded scenario in immediate mode graphics environments (see below). This is the equivalent of calling [draw_pass1\(\)](#), immediately followed by [draw_pass2\(\)](#), immediately followed by [post_draw\(\)](#).

Note that any character graphics culling, far position rendering, etc., depend on the render camera set by the [set_render_camera\(\)](#) function. A render camera does not always need to be set; see [set_render_camera\(\)](#) for more details.

Returns:

0 on success, -1 on failure

OpenGL Version:

Direct3D Version:

This function immediately draws all scenario elements. Either this function, or `draw_pass1()`, `draw_pass2()`, and `post_draw()`, should be called once per frame.

Callable From:

◇ C++

5.74.37 function diguyScenario::draw_pass1

Prototype:

```
int draw_pass1();
```

Description:

This function, along with `draw_pass2()`, allows the drawing of opaque and transparent polygons to be separated. This function draws all opaque polygons.

Returns:

0 on success, -1 on failure

*OpenGL Version:**Direct3D Version:*

This function immediately draws the opaque scenario elements. Either this function or `draw()` should be called once per frame.

Callable From:

◇ C++

5.74.38 function diguyScenario::draw_pass2**Prototype:**

```
int draw_pass2(int also_call_post_draw = 1);
```

Description:

Same as [draw_pass1\(\)](#), but draws transparent scenario elements. The `also_call_post_draw` argument is set to 1 for backwards compatibility.

Callable From:

◇ C++

5.74.39 function diguyScenario::post_draw**Prototype:**

```
int post_draw(int call_plugins_post_draw = 1,
              int call_lua_packages_draw = 1,
              int call_post_draw_callback = 1);
```

Description:

This function draws visual elements that must be drawn after the main render has been done.

Note that this function is called automatically by the `draw()` call. Only call this function if `draw()` is not called; e.g., if `draw_pass1()` and `draw_pass2()` are used.

This function can be called from scene graph type renderers which typically don't need or want to have the `draw()` function called.

Arguments:

<code>call_plugins_post_draw</code>	pass 1 to call plugin <code>post_draw()</code> functions
<code>call_lua_packages_draw</code>	pass 1 to call lua package draw functions
<code>call_post_draw_callback</code>	pass 1 to invoke callbacks with callback id <code>CALLBACK_ID_POST_DRAW</code>

Returns:

0 on success, -1 on failure

5.74.40 function diguyScenario::set_graphics_attach_ptr**Prototype:**

```
int set_graphics_attach_ptr(void* graphics_attach_ptr);
```

Description:

This function sets the attachment point that to be used for graphics created by the library.

Arguments:

`graphics_attach_ptr` this is a pointer to a group-type object; the specific type is dependent on graphics environment

Returns:

0 on success, -1 on failure

OpenGL Version:

This function does not have any effect.

Direct3D Version:

The passed pointer should be a LPDIRECT3DRMFRAME2.

```
LPDIRECT3DRMFRAME2 attach_frame = get_scene_frame();
scenario->set_graphics_attach_ptr(attach_frame);
```

Callable From:

◇ C++

Merge Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.41 function diguyScenario::create_merge_settings

Prototype:

```
diguyScenarioMergeSettings* create_merge_settings(diguyScenarioMergeInitialSettings
```

Description:

This function creates a merge settings object. This object can allow for fine control over which elements in a scenario are merged during a [merge\(\)](#) call, and what to do if there is a collision with an existing element.

Once the initial settings are made based on the `initial_settings` argument, the settings can be fine-tuned by calling member functions of the [diguyScenarioMergeSettings](#) object.

Destroy the object with a call to [destroy_merge_settings\(\)](#).

Arguments:

`initial_settings` what the initial settings should be when the object is created

DIGUY_MERGE_INITIAL_SETTINGS_DEFAULT

scenario settings	DIGUY_MERGE_FLAG_DONT_MERGE
current view settings	DIGUY_MERGE_FLAG_OVERWRITE_EXISTING
character decision formation group guide info popup library function script sensor region signal variables camera settings face exp fog settings light settings particle system path shape scene object scene object grid sound	DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL

DIGUY_MERGE_INITIAL_SETTINGS_MERGE_EVERYTHING

DI-Guy SDK Version 12.5.2 Index

scenario settings	DIGUY_MERGE_FLAG_OVERWRITE_EXISTING
current view settings	DIGUY_MERGE_FLAG_OVERWRITE_EXISTING
character	DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN
decision	DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN
formation	DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN
group	DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN
guide	DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN
info popup	DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN
library function	DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN
script	DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN
sensor region	DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN
signal	DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN
variables	DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN
camera settings	DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL
face exp	DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL
fog settings	DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL
light settings	DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL
particle system	DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL
path shape	DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL
scene object	DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL
scene object grid	DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL
sound	DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL

DIGUY_MERGE_INITIAL_SETTINGS_MERGE_NOTHING

scenario settings	DIGUY_MERGE_FLAG_DONT_MERGE

DI-Guy SDK Version 12.5.2 Index

current view settings	DIGUY_MERGE_FLAG_DONT_MERGE
character	DIGUY_MERGE_FLAG_DONT_MERGE
decision	DIGUY_MERGE_FLAG_DONT_MERGE
formation	DIGUY_MERGE_FLAG_DONT_MERGE
group	DIGUY_MERGE_FLAG_DONT_MERGE
guide	DIGUY_MERGE_FLAG_DONT_MERGE
info popup	DIGUY_MERGE_FLAG_DONT_MERGE
library function	DIGUY_MERGE_FLAG_DONT_MERGE
script	DIGUY_MERGE_FLAG_DONT_MERGE
sensor region	DIGUY_MERGE_FLAG_DONT_MERGE
signal	DIGUY_MERGE_FLAG_DONT_MERGE
variables	DIGUY_MERGE_FLAG_DONT_MERGE
camera settings	DIGUY_MERGE_FLAG_DONT_MERGE
face exp	DIGUY_MERGE_FLAG_DONT_MERGE
fog settings	DIGUY_MERGE_FLAG_DONT_MERGE
light settings	DIGUY_MERGE_FLAG_DONT_MERGE
particle system	DIGUY_MERGE_FLAG_DONT_MERGE
path shape	DIGUY_MERGE_FLAG_DONT_MERGE
scene object	DIGUY_MERGE_FLAG_DONT_MERGE
scene object grid	DIGUY_MERGE_FLAG_DONT_MERGE
sound	DIGUY_MERGE_FLAG_DONT_MERGE

DIGUY_MERGE_INITIAL_SETTINGS_DIGUY6_COMPATIBLE

scenario settings	DIGUY_MERGE_FLAG_OVERWRITE_EXISTING
current view	DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

settings	
character	DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN
decision	DIGUY_MERGE_FLAG_KEEP_BOTH
formation	DIGUY_MERGE_FLAG_KEEP_BOTH
group	DIGUY_MERGE_FLAG_KEEP_BOTH
guide	DIGUY_MERGE_FLAG_KEEP_BOTH
info	DIGUY_MERGE_FLAG_KEEP_BOTH
popup	
library	DIGUY_MERGE_FLAG_KEEP_BOTH
function	
script	DIGUY_MERGE_FLAG_KEEP_BOTH
sensor	DIGUY_MERGE_FLAG_KEEP_BOTH
region	
signal	DIGUY_MERGE_FLAG_KEEP_BOTH
variables	DIGUY_MERGE_FLAG_KEEP_BOTH
camera	DIGUY_MERGE_FLAG_KEEP_BOTH
settings	
face exp	DIGUY_MERGE_FLAG_KEEP_BOTH
fog	DIGUY_MERGE_FLAG_KEEP_BOTH
settings	
light	DIGUY_MERGE_FLAG_KEEP_BOTH
settings	
particle	DIGUY_MERGE_FLAG_KEEP_EXISTING_ONLY_AND_WARN
system	
path	DIGUY_MERGE_FLAG_KEEP_BOTH
shape	
scene	DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL
object	
scene	DIGUY_MERGE_FLAG_KEEP_BOTH
object	
grid	
sound	DIGUY_MERGE_FLAG_KEEP_BOTH

Returns:

object of type [diguyScenarioMergeSettings](#)

5.74.42 function diguyScenario::destroy_merge_settings**Prototype:**

```
void destroy_merge_settings(diguyScenarioMergeSettings* merge_settings);
```

Description:

This function destroys a merge settings object created by a call to `create_merge_settings()`.

5.74.43 function diguyScenario::merge**Prototype:**

```
int merge(const char* filename,
          diguyScenarioMergeSettings* merge_settings = NULL);
```

Description:

This function merges the specified scenario. Any previously loaded scenario elements will be retained, depending on settings in the `merge_settings` argument.

Arguments:

<code>filename</code>	filename of scenario to be merged
<code>merge_settings</code>	object that controls how elements are merged; pass NULL for default merge handling

Returns:

0 on success, -1 on failure

5.74.44 function diguyScenario::merge_object**Prototype:**

```
int merge_object(const char* string);
```

Description:

This function performs a "micro merge", allowing a scenario to merge in the contents of a text string representing a single object.

One example of use is for a Lua Package to contain a number of text strings representing particle systems or sounds that an agent will need. When needed objects in text form can be merged into the scenario using this function.

The DI-Guy Scenario Particle System page has a "Display Object Text" button that can display the object text of a particle system that is appropriate for use by this function. Other object pages that have this button are Sound, Formation, and Crowd Profile.

Text can also be taken directly from .dss files.

See also the [diguyCharacter::merge_object\(\)](#) function.

Note: the parser is very sensitive to tabs and formatting!

5.74.45 function diguyScenario::include_object_library**Prototype:**

```
int include_object_library(const char* library_name);
```

Description:

This function makes the objects in the specified library available for use in the scenario.

Object libraries allow new content to be created in one scenario and automatically shared to other scenarios. When including a library, objects that are already in the scenario are preserved.

A library with specified name must be available for inclusion. Object libraries are located in the \$(DIGUY)*config*diguy/object_libraries and \$(DIGUY)*customconfig*diguy/object_libraries directories. The filename for the library must be "library_[libname].cfg".

Available object libraries in these directories will automatically be detected by the scenario, but objects in the libraries will not be included in the scenario until this function is called.

For example, if the objects in the file "library_crowd_sounds.cfg" should be made available in the scenario, the library name "crowd_sounds" should be passed into this function.

Currently the following object types can be in libraries:

- ◇ particle system
- ◇ formation
- ◇ crowd profile
- ◇ sound

Arguments:

`library_name` name of library

Returns:

0 on success, -1 on failure

5.74.46 function diguyScenario::uninclude_object_library

Prototype:

```
int uninclude_object_library(const char* library_name);
```

Description:

This function is the opposite of `include_object_library()`. Objects in the named library are made unavailable to the scenario.

Arguments:

`library_name` name of library

Returns:

0 on success, -1 on failure

Time and Date Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

Datetime Functions

5.74.47 function diguyScenario::set_base_datetime**Prototype:**

```
void set_base_datetime(int year = -1,
    int month = -1,
    int day = -1,
    int hour = -1,
    int minute = -1,
    int second = -1,
    int msec = -1);
```

Description:

This function will set what the current datetime will be when the scanario is reset.

Pass -1 for values that should be set based on current datetime.

Note that if [set_base_datetime_default_to_now\(\)](#) and [set_base_datetime_default_to_today\(\)](#) can override these values.

5.74.48 function diguyScenario::set_base_datetime**Prototype:**

```
void set_base_datetime(const char* datetime_string,
    const char* format);
```

Description:

Similar to [set_base_datetime\(\)](#), but datetime is stored in the passed string, formatted in the passed format string.

5.74.49 function diguyScenario::get_base_datetime**Prototype:**

```
void get_base_datetime(int* year = NULL,
    int* month = NULL,
    int* day = NULL,
    int* hour = NULL,
    int* minute = NULL,
    int* second = NULL,
    int* msec = NULL);
```

Description:

Retrieves the base datetime. Pass NULL for arguments that aren't needed.

5.74.50 function diguyScenario::set_base_datetime_default_to_now**Prototype:**

```
void set_base_datetime_default_to_now(int to_now);
```

Description:

Setting this value to 1 will cause the time part of the datetime to be set to whatever "real" time it is when `reset( )` is called, instead of the time stored in the base datetime as set by `set_base_datetime( )`. For example, if it is 10:30 AM in "real time" when the scenario is reset, the current time will be set to 10:30:00:000 on reset.

Setting this value to 0 will restore the default behavior.

5.74.51 function**diguyScenario::get_base_datetime_default_to_now****Prototype:**

```
int get_base_datetime_default_to_now();
```

Returns:

1 if the current time will be set to the "real time" on reset, 0 is not

5.74.52 function**diguyScenario::set_base_datetime_default_to_today****Prototype:**

```
void set_base_datetime_default_to_today(int to_today);
```

Description:

Similar to [set_base_datetime_default_to_now\(\)](#), but for the date part of the datetime.

5.74.53 function diguyScenario::get_base_datetime_default_to_today

Prototype:

```
int get_base_datetime_default_to_today();
```

Returns:

1 if the current date will be set to the "real date" on reset, 0 is not

5.74.54 function diguyScenario::set_current_datetime

Prototype:

```
void set_current_datetime(int year = -1,
    int month = -1,
    int day = -1,
    int hour = -1,
    int minute = -1,
    int second = -1,
    int msec = -1);
```

Description:

Allows current datetime to be explicitly set. Note that in most cases the current datetime should be set by the base datetime plus how much time has advanced in the scenario.

This function can be useful if the datetime needs to be synchronized with that from another source, say, from network exercise data.

5.74.55 function diguyScenario::set_current_datetime**Prototype:**

```
void set_current_datetime(const char* datetime_string,  
    const char* format);
```

Description:

Similar to [set_current_datetime\(\)](#), but datetime is stored in the passed string, formatted in the passed format string.

5.74.56 function diguyScenario::get_current_datetime**Prototype:**

```
void get_current_datetime(int* year = NULL,  
    int* month = NULL,  
    int* day = NULL,  
    int* hour = NULL,  
    int* minute = NULL,  
    int* second = NULL,  
    int* msec = NULL);
```

Description:

Retrieves the current datetime. Pass NULL for arguments that aren't needed.

5.74.57 function diguyScenario::get_current_datetime_string**Prototype:**

```
const char* get_current_datetime_string(const char* format);
```

Description:

Retrieves the current datetime in string format.

Arguments:

format	format returned string should be in; pass NULL for default "Mdyyyy h:mm:ss AP" format

5.74.58 function diguyScenario::set_datetimes_are_utc**Prototype:**

```
void set_datetimes_are_utc(int are_utc);
```

Description:

Pass 1 to specify that datetimes use UTC time, 0 to specify they are local time.

5.74.59 function diguyScenario::get_datetimes_are_utc**Prototype:**

```
int get_datetimes_are_utc();
```

Returns:

1 if datetimes use UTC time, 0 if not

5.74.60 function diguyScenario::set_utc_offset_hours**Prototype:**

```
void set_utc_offset_hours(float utc_offset_hours);
```

Description:

If datetimes use UTC time, this function sets the offset hours of the current time from UTC.

5.74.61 function diguyScenario::get_utc_offset_hours

Prototype:

```
float get_utc_offset_hours();
```

Returns:

offset in hours of datetimes from UTC

5.74.62 function diguyScenario::set_use_local_tz_utc_offset_hours

Prototype:

```
void set_use_local_tz_utc_offset_hours(int use_local_tz);
```

Description:

Documentation Pending

5.74.63 function diguyScenario::get_use_local_tz_utc_offset_hours

Prototype:

```
int get_use_local_tz_utc_offset_hours();
```

Description:

Documentation Pending

5.74.64 function diguyScenario::set_current_datetime_advance_method

Prototype:

```
void set_current_datetime_advance_method(diguyDatetimeAdvanceMethod advance_method);
```

Description:

Documentation Pending

5.74.65 function

diguyScenario::get_current_datetime_advance_method

Prototype:

```
diguyDatetimeAdvanceMethod get_current_datetime_advance_method();
```

Description:

Documentation Pending

5.74.66 function

diguyScenario::set_current_datetime_advance_factor

Prototype:

```
void set_current_datetime_advance_factor(float advance_factor);
```

Description:

Documentation Pending

5.74.67 function diguyScenario::get_current_datetime_advance_factor

Prototype:

```
float get_current_datetime_advance_factor();
```

Description:

Documentation Pending

5.74.68 function diguyScenario::get_network_datetime

Prototype:

```
void get_network_datetime(int* year = NULL,  
    int* month = NULL,  
    int* day = NULL,  
    int* hour = NULL,  
    int* minute = NULL,  
    int* second = NULL,  
    int* msec = NULL);
```

Description:

Documentation Pending

Simple Time Functions

5.74.69 function diguyScenario::set_tin_time_of_day**Prototype:**

```
int set_tin_time_of_day(float t);
```

Description:

This function sets the time of day which corresponds to scenario tin. It is a wall-clock time, and has no date content.

Note that if more detailed time and date information is needed refer to the "datetime" functions.

Arguments:

t	number of seconds after midnight; zero means midnight; 86399.0 means 11:59:59 pm

Returns:

0 on success, -1 on failure

5.74.70 function diguyScenario::get_tin_time_of_day**Prototype:**

```
float get_tin_time_of_day();
```

Description:

This function returns what time of day corresponds to scenario tin. It is a wall-clock time, and has no date content.

See [set_tin_time_of_day\(\)](#).

Note that if more detailed time and date information is needed refer to the "datetime" functions.

Returns:

number of seconds after midnight

5.74.71 function diguyScenario::get_time_of_day

Prototype:

```
float get_time_of_day();
```

Description:

This function returns what the simulated time of day is right now. It is computed by adding (scenario t – tin) to the value passed to [set_tin_time_of_day\(\)](#);

See [set_tin_time_of_day\(\)](#).

Note that if more detailed time and date information is needed refer to the "datetime" functions.

Returns:

number of seconds after midnight (simulated time of day)

Appearance Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.72 function diguyScenario::add_appearance**Prototype:**

```
int add_appearance(const char* appearance,
                  char** equipment_list,
                  char** character_types,
                  const char* actor);
```

Description:

This function adds an appearance with the given name to the list of available appearances. If an appearance with the given name already exists the operation will fail and a message will be printed at debug level to the output log.

Internal copies are made of the passed strings.

Arguments:

<code>appearance</code>	name of the appearance
<code>equipment_list</code>	NULL-terminated list of equipment that makes up the appearance
<code>character_types</code>	NULL-terminated list of character types that this appearance will be available to
<code>actor</code>	name of the actor this appearance fits

If NULL is passed for the actor argument the actor will be set to match the actor of the first character type in the `character_types` array.

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.74.73 function**diguyScenario::add_appearance_based_on_existing_appearance****Prototype:**

```
int add_appearance_based_on_existing_appearance(const char* appearance,
const char* existing_appearance,
char** equipment_list);
```

Description:

This function is similar to [add_appearance \(\)](#), but adds equipment_list shapes to an existing appearance.

The character types that the new appearance will be available to will match that of the specified existing appearance.

Arguments:

appearance	name of the new appearance
existing_appearance	name of the existing appearance on which to base new appearance
equipment_list	NULL-terminated list of equipment to add to the existing appearance

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.74.74 function diguyScenario::remove_appearance**Prototype:**

```
int remove_appearance(const char* appearance);
```

Description:

This function removes an appearance with the given name from the list of available appearances. If an appearance with the given name does not exist the operation will fail and a message will be printed at debug level to the output log.

NOTE:

Currently no check is made for whether the appearance is in use by a character. Removing an in-use appearance will result in undefined behavior.

Arguments:

appearance name of the appearance

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.74.75 function**diguyScenario::get_num_character_types_with_appearance****Prototype:**

```
int get_num_character_types_with_appearance(const char* appearance_name);
```

Description:

This function returns the number of character types associated with the given appearance, if any.

Arguments:

appearance name of the appearance

Returns:

the number of character types, or 0

5.74.76 function**diguyScenario::get_character_type_from_appearance****Prototype:**

```
const char *get_character_type_from_appearance(const char* appearance_name, int index);
```

Description:

This function returns one of the character types associated with the appearance, if any exists

Arguments:

appearance name of the appearance

index which character type of the set to return

Returns:

a character type name on success, "none" on failure

Preload Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.77 function diguyScenario::preload_character_type

Prototype:

```
int preload_character_type(const char* character_type);
```

Description:

This function preloads all base motions and the default appearance for the specified character type.

This can be done during initialization to preload characters that will be needed later in a scenario, to avoid a hitch in frame rate when the characters are loaded.

Arguments:

`character_type` character type name, as returned by `get_character_type_at_index()`, for example

Returns:

0 on success, -1 on failure

5.74.78 function diguyScenario::preload_appearance

Prototype:

```
int preload_appearance(const char* appearance);
```

Description:

This function preloads all geometry for the specified appearance. This can be done during initialization to preload geometry that will be needed later in a scenario, to avoid a hitch in frame rate when the geometry is loaded.

Arguments:

`appearance` name of the appearance whose geometry is to be preloaded

Returns:

0 on success, -1 on failure

5.74.79 function diguyScenario::preload_gesture**Prototype:**

```
int preload_gesture(const char* character_type, const char* gesture_name);
```

Description:

This function preloads all motions for the specified gesture for the specified character type.

This can be done during initialization to preload gestures that will be needed later in a scenario, to avoid a hitch in frame rate when the motions are loaded.

Arguments:

character_type character's type
gesture_name name of the gesture to preload

Returns:

0 on success, -1 on failure

Culling Functions

These functions control culling of various object types.

Note that these functions only work only in the OpenGL version, when a valid render camera is set. See [set_render_camera\(\)](#).

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.80 function diguyScenario::set_character_culling_enabled**Prototype:**

```
void set_character_culling_enabled(int character_culling);
```

Description:

This function enables character culling.

This function works only in the OpenGL version when a valid render camera is set.

5.74.81 function diguyScenario::get_character_culling_enabled**Prototype:**

```
int get_character_culling_enabled();
```

Description:

This function gets the state of character culling.

This function works only in the OpenGL version when a valid render camera is set.

Returns:

1 if enabled, 0 if disabled.

5.74.82 function `diguyScenario::set_scene_object_culling_enabled`

Prototype:

```
void set_scene_object_culling_enabled(int object_culling);
```

Description:

This function enables scene object culling.

This function works only in the OpenGL version when a valid render camera is set.

5.74.83 function `diguyScenario::get_scene_object_culling_enabled`

Prototype:

```
int get_scene_object_culling_enabled();
```

Description:

This function get the state of scene object culling.

This function works only in the OpenGL version when a valid render camera is set.

Returns:

1 if enabled, 0 if disabled

5.74.84 function `diguyScenario::set_scene_object_sub_culling_enabled`

Prototype:

```
void set_scene_object_sub_culling_enabled(int object_culling);
```

Description:

This function enables scene object sub object culling.

This function works only in the OpenGL version when a valid render camera is set.

5.74.85 function `diguyScenario::get_scene_object_sub_culling_enabled`

Prototype:

```
int get_scene_object_sub_culling_enabled();
```

Description:

This function get the state of scene object sub culling.

This function works only in the OpenGL version when a valid render camera is set.

Returns:

1 if enabled, 0 if disabled

5.74.86 function

diguyScenario::set_character_visualize_bounds_enabled

Prototype:

```
void set_character_visualize_bounds_enabled(int visualize_bounds);
```

Description:

This function enables visualizing a character's bounding volume.

This function works only in the OpenGL version when a valid render camera is set.

5.74.87 function

diguyScenario::get_character_visualize_bounds_enabled

Prototype:

```
int get_character_visualize_bounds_enabled();
```

Description:

This function get the state of character bounding volume visualization.

This function works only in the OpenGL version when a valid render camera is set.

Returns:

1 if enabled, 0 if disabled

5.74.88 function

diguyScenario::set_scene_object_visualize_bounds_enabled

Prototype:

```
void set_scene_object_visualize_bounds_enabled(int val);
```

Description:

This function enables visualizing a scene object's bounding volume.

This function works only in the OpenGL version when a valid render camera is set.

5.74.89 function

diguyScenario::get_scene_object_visualize_bounds_enabled

Prototype:

```
int get_scene_object_visualize_bounds_enabled();
```

Description:

This function gets the state of scene objects bounding volume visualization.

This function works only in the OpenGL version when a valid render camera is set.

Returns:

1 if enabled, 0 if disabled

Query Functions

These functions query the scenario for what specific types of scenario elements are available. This includes:

- ◆ character types
- ◆ appearances
- ◆ actions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

C++ Example:

```
int num_character_types = diguy_scenario->get_num_character_types();
int character_type_index;
for (character_type_index = 0;
     character_type_index < num_character_types;
     character_type_index++)
{
    const char* character_type = diguy_scenario->get_character_type_at_index(character_type_index,
                                     character_type_index,
                                     character_type);

    int num_appearances = diguy_scenario->get_character_type_num_appearances(character_type_index,
                                     character_type_index,
                                     character_type);
    int appearance_index;
    for (appearance_index = 0;
         appearance_index < num_appearances;
         appearance_index++)
    {
        const char* appearance = diguy_scenario->get_character_type_appearance_at_index(character_type_index,
                                                                                          appearance_index,
                                                                                          character_type);
        bdi_log_printf(BDI_LOG_INFO, "\tAppearance at index %d: %s\n",
                       appearance_index,
                       appearance);
    }

    int num_actions = diguy_scenario->get_character_type_num_actions(character_type_index,
                                                                      character_type_index,
                                                                      character_type);
    int action_index;
    for (action_index = 0;
         action_index < num_actions;
         action_index++)
    {
        const char* action_name = diguy_scenario->get_character_type_action_at_index(character_type_index,
                                                                                       action_index,
                                                                                       character_type);
        bdi_log_printf(BDI_LOG_INFO, "\tAction at index %d: %s\n",
                       action_index,
                       action_name);
    }
}
```

5.74.90 function diguyScenario::get_num_character_types**Prototype:**

```
int get_num_character_types();
```

Returns:

number of available character types

5.74.91 function diguyScenario::get_character_type_at_index**Prototype:**

```
const char* get_character_type_at_index(int index);
```

Returns:

the string identifying the character type at the given index

Arguments:

<code>index</code>	index of the character type; indices start at 0

5.74.92 function

diguyScenario::get_character_type_abbreviation

Prototype:

```
const char* get_character_type_abbreviation(const char* character_type);
```

Returns:

abbreviation of the given character type, when a shorter identifier is needed

Arguments:

`character_type` character type name, as returned by
`get_character_type_at_index()`, for example

5.74.93 function

diguyScenario::get_character_type_deprecated_at_version

Prototype:

```
const char* get_character_type_deprecated_at_version(const char* character_type);
```

Description:

This function returns the version at which the specified character type was deprecated.

If the character type has not been deprecated, the empty string ("") is returned.

Arguments:

`character_type` character type name, as returned by
`get_character_type_at_index()`, for example

Returns:

version at which character became deprecated; empty string ("") if it hasn't been

C++ Example:

```
const char* version = scenario->get_character_type_deprecated_at_version();
if (strlen(version) > 0)
{
    // do something
}
```

5.74.94 function diguyScenario::get_character_type_deprecated_suggested_alternatives

Prototype:

```
const char* get_character_type_deprecated_suggested_alternatives(const char* character_type)
```

Description:

This function returns the suggested character type(s) that are recommended for deprecated character types.

If the character type has not been deprecated, the empty string ("") is returned.

If there is more than one suggested alternative, they will be separated by spaces.

Arguments:

character_type character type name, as returned by `get_character_type_at_index()`, for example

Returns:

suggested replacement character type(s)

C++ Example:

```
const char* alternatives = scenario->get_character_type_deprecated_suggested_alternatives(version);
if (strlen(version) > 0)
{
    // do something
}
```

5.74.95 function diguyScenario::get_character_type_quality_bias

Prototype:

```
int get_character_type_quality_bias(const char* character_type);
```

Description:

As the number of character types DI-Guy provides has grown, newer character types tend to be better than older ones. This can be because of a greater selection of actions, better quality motions, and other factors.

This function returns a value representing a qualitative "bias" of this character they relative to other character types. The higher the bias, the more recommended the character type is. A bias of 1 means strongly not recommended, 5 means average, 10 means highly recommended.

Returns:

quality bias, a unitless value from 1 to 10

Arguments:

`character_type` character type name, as returned by
`get_character_type_at_index()`, for example

5.74.96 function `diguyScenario::get_character_type_map`

Prototype:

```
diguyCharacterTypeMap* get_character_type_map(const char* character_type);
```

Description:

This function returns the type map of the specified character type. See the documentation for [diguyCharacterTypeMap](#) for what this information means.

Only the character type fields will have useful information; the appearance fields will contain default wildcard values.

Arguments:

`character_type` character type name, as returned by
`get_character_type_at_index()`, for example

Returns:

pointer to type [diguyCharacterTypeMap](#); will never be NULL

5.74.97 function `diguyScenario::get_character_type_default_appearance`

Prototype:

```
const char* get_character_type_default_appearance(const char* character_type);
```

Returns:

default appearance of the specified character type

Arguments:

`character_type` character type name, as returned by
`get_character_type_at_index()`, for example

5.74.98 function diguyScenario::get_character_type_default_skinned_appearance

Prototype:

```
const char* get_character_type_default_skinned_appearance(const char* character_type)
```

Returns:

default skinned appearance of the specified character type

Arguments:

`character_type` character type name, as returned by
get_character_type_at_index(), for example

5.74.99 function diguyScenario::get_character_type_num_appearances

Prototype:

```
int get_character_type_num_appearances(const char* character_type);
```

Returns:

number of available appearances for the given character type

Arguments:

`character_type` character type name, as returned by
get_character_type_at_index(), for example

5.74.100 function**diguyScenario::get_character_type_appearance_at_index****Prototype:**

```
const char* get_character_type_appearance_at_index(const char* character_type,
int index);
```

Returns:

the string identifying the appearance at the given index for the specified character type

Arguments:

<code>character_type</code>	character type name, as returned by <code>get_character_type_at_index()</code> , for example
<code>index</code>	index of the appearance; indices start at 0

5.74.101 function**diguyScenario::get_character_type_num_head_appearances****Prototype:**

```
int get_character_type_num_head_appearances(const char* character_type);
```

Similar to `get_character_type_num_appearances()`, but for head appearances.

5.74.102 function**diguyScenario::get_character_type_head_appearance_at_index****Prototype:**

```
const char* get_character_type_head_appearance_at_index(const char* character_type,
int index);
```

Similar to [get_character_type_appearance_at_index\(\)](#), but for head appearances.

5.74.103 function**diguyScenario::get_character_type_num_actions****Prototype:**

```
int get_character_type_num_actions(const char* character_type);
```

Returns:

number of available actions for the given character type

Arguments:

character_type character type name, as returned by [get_character_type_at_index\(\)](#), for example

5.74.104 function diguyScenario::get_character_type_action_at_index

Prototype:

```
const char* get_character_type_action_at_index(const char* character_type,
                                             int index);
```

Returns:

the string identifying the action at the given index for the specified character type

Arguments:

<code>character_type</code>	character type name, as returned by <code>get_character_type_at_index()</code> , for example
<code>index</code>	index of the action name; indices start at 0

5.74.105 function diguyScenario::get_character_type_action_speed

Prototype:

```
float get_character_type_action_speed(const char* character_type,
                                     const char* action_name);
```

Returns:

the average speed of the specified action for the specified character type

Arguments:

<code>character_type</code>	character type name, as returned by <code>get_character_type_at_index()</code> , for example
<code>action_name</code>	the action to query

5.74.106 function diguyScenario::get_character_type_action_duration

Prototype:

```
float get_character_type_action_duration(const char* character_type,
                                       const char* action_name);
```

Returns:

the duration of one rep of the specified action for the specified character type

Arguments:

<code>character_type</code>	character type name, as returned by <code>get_character_type_at_index()</code> , for example
<code>action_name</code>	the action to query

5.74.107 function diguyScenario::get_character_type_action_distance

Prototype:

```
float get_character_type_action_distance(const char* character_type,
                                       const char* action_name);
```

Returns:

the distance covered by one rep of the specified action for the specified character type

Arguments:

<code>character_type</code>	character type name, as returned by <code>get_character_type_at_index()</code> , for example
<code>action_name</code>	the action to query

5.74.108 function**diguyScenario::get_character_type_action_direction****Prototype:**

```
diguyMotionDirection get_character_type_action_direction(
    const char* character_type,
    const char* action_name);
```

Returns:

the [diguyMotionDirection](#) identifying the direction of travel of the specified action for the specified character type

Arguments:

character_type character type name, as returned by
get_character_type_at_index(), for example

action_name the action to query

5.74.109 function**diguyScenario::get_character_type_action_facing_angle_in****Prototype:**

```
float get_character_type_action_facing_angle_in(const char* character_type,
    const char* action_name);
```

Returns:

the facing angle at the start of each repetition of the specified action for the specified character type, in degrees

Arguments:

character_type character type name, as returned by
get_character_type_at_index(), for example

action_name the action to query

5.74.110 function

diguyScenario::get_character_type_action_facing_angle_out

Prototype:

```
float get_character_type_action_facing_angle_out(const char* character_type,
const char* action_name);
```

Returns:

the facing angle at the end of each repetition of the specified action for the specified character type, in degrees

Arguments:

<code>character_type</code>	character type name, as returned by <code>get_character_type_at_index()</code> , for example
<code>action_name</code>	the action to query

5.74.111 function

diguyScenario::get_character_type_num_gestures

Prototype:

```
int get_character_type_num_gestures(const char* character_type);
```

Returns:

the number of gestures available to the specified character type

Arguments:

<code>character_type</code>	character type name, as returned by <code>get_character_type_at_index()</code> , for example; pass "all" to get all gestures available, regardless of character type

5.74.112 function**diguyScenario::get_character_type_gesture_at_index****Prototype:**

```
const char* get_character_type_gesture_at_index(const char* character_type,
                                              int index);
```

Returns:

the name of the gesture available to the specified character type at the specified index

Arguments:

<code>character_type</code>	character type name, as returned by <code>get_character_type_at_index()</code> , for example; pass "all" to get all gestures available, regardless of character type
<code>index</code>	index of the gesture; indices start at 0

Character Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.113 function diguyScenario::create_character**Prototype:**

```
diguyCharacter* create_character(const char* name,
                                const char* character_type,
                                const char* appearance = NULL);
```

Description:

This function creates a new character and returns a pointer to it.

The function [preload_character_type\(\)](#) should be called at the beginning of a scenario in which this call will be made to avoid a hitch in frame rate.

See also [retrieve_character_from_recycle_bin\(\)](#) for a potentially faster way of creating a character.

Arguments:

name	name of the new character
character_type	character's type
appearance	base appearance of character; pass NULL to use default appearance

Returns:

pointer of type [diguyCharacter](#); NULL if creation failed

5.74.114 function diguyScenario::create_temporary_character**Prototype:**

```
diguyCharacter* create_temporary_character(const char* name,
    const char* character_type,
    const char* appearance = NULL);
```

Description:

This function is very similar to [create_character\(\)](#), but the character created is temporary.

Temporary characters will be destroyed automatically when the scenario is reset, and are not saved in .dss files. They are commonly used for projectiles, effects, and other objects that can come and go multiple times.

Temporary characters can be destroyed before scenario reset by calling [destroy_character\(\)](#), and can be sent to the character recycle bin by calling [send_character_to_recycle_bin\(\)](#).

Arguments:

name	name of the new character
character_type	character's type
appearance	base appearance of character; pass NULL to use default appearance

Returns:

pointer of type [diguyCharacter](#); NULL if creation failed

5.74.115 function diguyScenario::destroy_character**Prototype:**

```
int destroy_character(diguyCharacter* character);
```

Description:

This function destroys a character.

If a character of the same character type and appearance may be needed later in the scenario, consider using [send_character_to_recycle_bin\(\)](#) instead. Characters can be retrieved from the recycle bin more quickly than creating them "from scratch" with [create_character\(\)](#).

Arguments:

`character` pointer to a [diguyCharacter](#)

Returns:

0 on success, -1 on failure

DI-Guy Networking Notes:

This function should not be called on characters that are reflected network characters. Such characters are destroyed automatically by DI-Guy Networking.

5.74.116 function diguyScenario::send_character_to_recycle_bin**Prototype:**

```
int send_character_to_recycle_bin(diguyCharacter* character);
```

Description

This function is an alternative way of removing a character from the scenario. When a character is recycled it is removed from the scenario's list of active characters, but the memory associated with it is not freed. Instead, the character is placed in a "character recycle bin" from which it may be later retrieved with a call to [retrieve_character_from_recycle_bin\(\)](#).

For an application that is creating and destroying a lot of characters with the same character types and appearances, this can avoid performance hiccups that happen when memory is allocated for new characters.

If it is known ahead of time what types and appearances of characters will be needed in the future, the recycle bin can be "pre-stocked" at scenario load time by first creating and then recycling a large number of characters all at once.

Returns:

0 on success, -1 on failure

5.74.117 function**diguyScenario::retrieve_character_from_recycle_bin****Prototype:**

```
diguyCharacter* retrieve_character_from_recycle_bin(const char* name,
    const char* character_type,
    const char* appearance,
    int make_character_temporary = 0);
```

Description

This function is an alternative way of creating a character. The character recycle bin is checked for a recycled character whose character type and appearance match that requested by this call. Characters are sent to the recycle bin by a call to [send_character_to_recycle_bin\(\)](#).

If a match is found the recycled character will be removed from the recycle bin, given the passed name, and placed in the scenario's list of characters, for the most part just as if the character had been created by a call to [create_character\(\)](#). Some things such as position, current action, etc., may need to be immediately set.

If a match is not found, this function returns NULL and a call to [create_character\(\)](#) must be made instead.

Arguments:

name	new name for the character retrieved from bin
character_type	the character type to retrieve from the bin
appearance	the appearance to retrieve from the bin, pass NULL or "default" to retrieve the default appearance
make_character_temporary	pass 1 to make character temporary, 0 to leave permanent

Returns:

0 on success, -1 on failure

5.74.118 function diguyScenario::get_num_characters**Prototype:**

```
int get_num_characters();
```

Returns:

number of characters in the scenario

5.74.119 function diguyScenario::get_character_at_index**Prototype:**

```
diguyCharacter* get_character_at_index(int index);
```

Returns:

pointer of type [diguyCharacter](#); NULL if no character at the specified index

Arguments:

index	index of the character; indices start at 0

5.74.120 function diguyScenario::find_character**Prototype:**

```
diguyCharacter* find_character(const char* name);
```

Description:

This function returns a pointer to the specified character.

Arguments:

name	name of character to be found

Returns:

pointer of type [diguyCharacter](#); NULL if not found

5.74.121 function diguyScenario::look_at_character**Prototype:**

```
void look_at_character(diguyCharacter* character);
```

Description:

This function changes the scenario camera's position and fix point so it will be looking at the specified character.

5.74.122 function diguyScenario::this_character**Prototype:**

```
diguyCharacter* this_character();
```

Description:

This function returns a pointer to the target character of an event bead.

Returns:

pointer of type [diguyCharacter](#); NULL if no current character

5.74.123 function diguyScenario::this_event_bead**Prototype:**

```
diguyCharacterPathEvent* this_event_bead();
```

Description:

This function returns a pointer to the current event bead.

Returns:

pointer of type [diguyCharacterPathEvent](#); NULL if no current event bead

Path Shape Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.124 function diguyScenario::get_num_path_shapes

Prototype:

```
int get_num_path_shapes();
```

Returns:

number of path shapes in the scenario

5.74.125 function diguyScenario::get_path_shape_at_index**Prototype:**

```
diguyPathShape* get_path_shape_at_index(int index);
```

Returns:

pointer of type [diguyPathShape](#); NULL if no path shape at the specified index

Arguments:

index index of the path shape; indices start at 0

5.74.126 function diguyScenario::get_current_path_shape**Prototype:**

```
diguyPathShape* get_current_path_shape();
```

Returns:

pointer of type [diguyPathShape](#); NULL if no path shape at the specified index

5.74.127 function diguyScenario::find_path_shape**Prototype:**

```
diguyPathShape* find_path_shape(const char* name);
```

Description:

This function returns a pointer to the specified path shape.

Arguments:

name	name of path shape to be found

Returns:

pointer of type [diguyPathShape](#); NULL if not found

5.74.128 function diguyScenario::create_path_shape**Prototype:**

```
diguyPathShape* create_path_shape(const char* name);
```

Description:

This function creates a new path shape and returns a pointer to it.

Arguments:

name	name of the new path shape

Returns:

pointer of type [diguyPathShape](#); NULL if creation failed

5.74.129 function diguyScenario::copy_path_shape**Prototype:**

```
diguyPathShape* copy_path_shape(diguyPathShape* existing_path_shape,
                                const char* name);
```

Description:

This function creates a new path shape based on an existing path shape and returns a pointer to it.

Arguments:

existing_path_shape	pointer to diguyPathShape to be copied
name	name of the new path shape

Returns:

pointer of type [diguyPathShape](#); NULL if creation failed

5.74.130 function diguyScenario::destroy_path_shape**Prototype:**

```
int destroy_path_shape(diguyPathShape* path_shape);
```

Description:

This function destroys a path shape.

Arguments:

path_shape	pointer to a diguyPathShape

Returns:

0 on success, -1 on failure

Waypoint Functions

5.74.131 function diguyScenario::create_waypoint

Prototype:

```
diguyWaypoint* create_waypoint(float x = 0.0f, float y = 0.0f, float z = 0.0f,
    float yaw = 0.0f, float roll = 0.0f, float pitch = 0.0f,
    float weight_in = DIGUY_DEFAULT_FLOAT,
    float weight_out = DIGUY_DEFAULT_FLOAT);
```

Description:

This function creates a new waypoint that can be used for creating new paths and path shapes.

The returned pointer must be destroyed using [destroy_waypoint\(\)](#). The scenario keeps no record of waypoints created with this function.

The weights of the waypoint control how long the "control handles" of the waypoint are, in meters. The longer the control handle, the more the path will be stretched in the direction of the waypoint's orientation.

Unless otherwise specified, callable from:

◇ C++
 ◇ Script

Arguments:

<code>tx, ty,</code> <code>tz</code>	position in meters from the origin
<code>rz, rx,</code> <code>ry</code>	orientations in degrees counter-clockwise from the positive X axis
<code>weight_in</code>	how much influence this waypoint exerts over the path curve entering this waypoint
<code>weight_out</code>	how much influence this waypoint exerts over the path curve leaving this waypoint

By default, `weight_in` will be set to 1, and `weight_out` will be coupled to `weight_in` so that changes to one will affect the other. See [diguyWaypoint::set_weight_in\(\)](#). If a non-default value is passed for `weight_out` the weights of the waypoint will not be coupled.

Returns:

pointer to type [diguyWaypoint](#)

5.74.132 function [diguyScenario::destroy_waypoint](#)

Prototype:

```
int destroy_waypoint(diguyWaypoint* waypoint);
```

Description:

This function destroys a waypoint created by [create_waypoint\(\)](#). Path shapes created using this waypoint will not be affected; they make internal copies of the waypoints that are used to create them.

Do **not** call this function on waypoints not created by [create_waypoint\(\)](#)! e.g., a waypoint returned by a call to [diguyCharacterPath::get_waypoint_at_index\(\)](#).

Arguments:

<code>waypoint</code>	waypoint to destroy

Returns:

0 on success, -1 on failure

Character Type Map Functions

See the top-level description of the [diguyCharacterTypeMap](#) for extensive information about character type maps.

See the DIGuyDigest programming example for examples on how to use these functions.

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.133 function

diguyScenario::get_nearest_character_type_map

Prototype:

```
diguyCharacterTypeMap* get_nearest_character_type_map(
    diguyCharacterTypeMapField field1, const char* field1_value, const char* field1_alternate_value,
    diguyCharacterTypeMapField field2 = DIGUY_CHARACTER_TYPE_MAP_FIELD_UNKNOWN, const char* field2_value, const char* field2_alternate_value,
    diguyCharacterTypeMapField field3 = DIGUY_CHARACTER_TYPE_MAP_FIELD_UNKNOWN, const char* field3_value, const char* field3_alternate_value,
    diguyCharacterTypeMapField field4 = DIGUY_CHARACTER_TYPE_MAP_FIELD_UNKNOWN, const char* field4_value, const char* field4_alternate_value,
    diguyCharacterTypeMapField field5 = DIGUY_CHARACTER_TYPE_MAP_FIELD_UNKNOWN, const char* field5_value, const char* field5_alternate_value,
    diguyCharacterTypeMapField field6 = DIGUY_CHARACTER_TYPE_MAP_FIELD_UNKNOWN, const char* field6_value, const char* field6_alternate_value,
    diguyCharacterTypeMapField field7 = DIGUY_CHARACTER_TYPE_MAP_FIELD_UNKNOWN, const char* field7_value, const char* field7_alternate_value,
    diguyCharacterTypeMapField field8 = DIGUY_CHARACTER_TYPE_MAP_FIELD_UNKNOWN, const char* field8_value, const char* field8_alternate_value)
```

Description:

This function searches DI-Guy's available character types and appearances for the closest match to the specified fields and values. See the documentation for [diguyCharacterTypeMap](#) for information on what fields are available, and what values they may take.

Arguments:

field1	field in type map to check
field1_value	value to check for
field1_alternate_value	acceptable alternate value

(Arguments for fields 2 through 8 are the same.)

Returns:

pointer to type [diguyCharacterTypeMap](#)

5.74.134 function**diguyScenario::get_nearest_character_type_map_using_field_str****Prototype:**

```
diguyCharacterTypeMap* get_nearest_character_type_map_using_field_strings(
    const char* field1_string, const char* field1_value, const char* field1_alternat
    const char* field2_string = NULL, const char* field2_value = NULL, const char* f
    const char* field3_string = NULL, const char* field3_value = NULL, const char* f
    const char* field4_string = NULL, const char* field4_value = NULL, const char* f
    const char* field5_string = NULL, const char* field5_value = NULL, const char* f
    const char* field6_string = NULL, const char* field6_value = NULL, const char* f
    const char* field7_string = NULL, const char* field7_value = NULL, const char* f
    const char* field8_string = NULL, const char* field8_value = NULL, const char* f
```

Description:

Same as above, but fields to check are specified by string name instead of enumeration value.

5.74.135 function**diguyScenario::get_nearest_character_type_map_id_match****Prototype:**

```
diguyCharacterTypeMap* get_nearest_character_type_map_id_match(const char* type_map
```

Description:

Same as above, but field values to match against are specified by the given type map id. No alternate values can be provided, though wildcards can be used by putting a "*" in for a field.

5.74.136 function

`diguyScenario::get_num_character_type_map_entries`

Prototype:

```
int get_num_character_type_map_entries();
```

Description:

This function returns the total number of type map entries that are in DI-Guy. It is against these entries that matches will be made.

This function, along with `get_character_type_map_entry_at_index()`, enables all type map information to be queried.

Returns:

number of total type map entries against which matches will be made

5.74.137 function

`diguyScenario::get_character_type_map_entry_at_index`

Prototype:

```
diguyCharacterTypeMap* get_character_type_map_entry_at_index(int index);
```

Returns:

type map entry at specified index; see `get_num_character_type_map_entries()`

5.74.138 function

`diguyScenario::get_num_type_map_character_classes`

Prototype:

```
int get_num_type_map_character_classes();
```

Description:

This function returns the total number of values that may be specified or returned in the character class field.

This function, along with [get_type_map_character_class_at_index\(\)](#), enables all available character classes to be queried.

Returns:

number of character classes

5.74.139 function

`diguyScenario::get_type_map_character_class_at_index`

Prototype:

```
const char* get_type_map_character_class_at_index(int index);
```

Returns:

character class field value at specified index; see [get_num_type_map_character_classes\(\)](#)

5.74.140 function diguyScenario::get_num_type_map_categories**Prototype:**

```
int get_num_type_map_categories(const char* character_class);
```

Description:

This function returns the total number of values that may be specified or returned in the category field of a type map with the given character class.

This function, along with [get_type_map_category_at_index\(\)](#), enables all available categories available to each character class to be queried.

Returns:

number of categories

5.74.141 function diguyScenario::get_type_map_category_at_index**Prototype:**

```
const char* get_type_map_category_at_index(const char* character_class,  
int index);
```

Returns:

category field value at specified index; see [get_num_type_map_categories\(\)](#)

5.74.142 function

diguyScenario::get_num_type_map_subcategories

Prototype:

```
int get_num_type_map_subcategories(const char* character_class,
                                   const char* category);
```

Description:

This function returns the total number of values that may be specified or returned in the subcategory field of a type map with the given character class and category.

This function, along with [get_type_map_subcategory_at_index\(\)](#), enables all available subcategories available to each character class and category to be queried.

Returns:

number of subcategories

5.74.143 function

diguyScenario::get_type_map_subcategory_at_index

Prototype:

```
const char* get_type_map_subcategory_at_index(const char* character_class,
                                              const char* category,
                                              int index);
```

Returns:

subcategory field value at specified index; see [get_num_type_map_subcategories\(\)](#)

Signal Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.144 function diguyScenario::get_num_signals

Prototype:

```
int get_num_signals();
```

Returns:

number of signals in the scenario

5.74.145 function diguyScenario::get_signal_at_index**Prototype:**

```
diguySignal* get_signal_at_index(int index);
```

Returns:

pointer of type [diguySignal](#); NULL if no signal at the specified index

Arguments:

index	index of the signal; indices start at 0

5.74.146 function diguyScenario::find_signal**Prototype:**

```
diguySignal* find_signal(const char* name);
```

Description:

This function returns a pointer to the specified signal.

Arguments:

name	name of signal to be found

Returns:

pointer of type [diguySignal](#); NULL if not found

5.74.147 function diguyScenario::create_signal**Prototype:**

```
diguySignal* create_signal(const char* name);
```

Description:

This function creates a new signal and returns a pointer to it.

Arguments:

name	name of the new signal

Returns:

pointer of type [diguySignal](#)

5.74.148 function diguyScenario::destroy_signal**Prototype:**

```
int destroy_signal(diguySignal* signal);
```

Description:

This function destroys a signal.

Arguments:

signal	pointer to a diguySignal

Returns:

0 on success, -1 on failure

5.74.149 function diguyScenario::reset_signals**Prototype:**

```
void reset_signals();
```

Description:

This function resets all signals in the scenario.

5.74.150 function diguyScenario::signal_palette_hide_all**Prototype:**

```
int signal_palette_hide_all(const char* filter = NULL);
```

Description:

This function hides all signals in the signal palette.

Arguments:

<code>filter</code>	optional filter, will only hide signals tagged with <code>info_text</code> equaling <code>filter</code>

5.74.151 function diguyScenario::signal_palette_show_all**Prototype:**

```
int signal_palette_show_all(const char* filter = NULL);
```

Description:

This function shows all signals in the signal palette.

Arguments:

<code>filter</code>	optional filter, will only show signals tagged with info_text filter

Sound Functions

Note that sounds are not played directly. Instead they are used as templates for sound instances. See [diguyCharacter::play_sound\(\)](#) and [diguyCharacter::create_sound_instance\(\)](#).

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.152 function diguyScenario::get_num_sounds**Prototype:**

```
int get_num_sounds();
```

Returns:

number of sounds in the scenario

5.74.153 function diguyScenario::get_sound_at_index**Prototype:**

```
diguySound* get_sound_at_index(int index);
```

Returns:

pointer of type [diguySound](#); NULL if no sound at the specified index

Arguments:

<code>index</code>	index of the sound; indices start at 0

5.74.154 function diguyScenario::find_sound**Prototype:**

```
diguySound* find_sound(const char* name);
```

Description:

This function returns a pointer to the specified sound.

Arguments:

name	name of sound to be found

Returns:

pointer of type [diguySound](#); NULL if not found

5.74.155 function diguyScenario::create_sound**Prototype:**

```
diguySound* create_sound(const char* name,
    const char* sound_filename = NULL,
    const char* phoneme_filename = NULL);
```

Description:

This function creates a new sound and returns a pointer to it.

Note that sounds are not played directly. Instead they are used as templates for sound instances. See [diguyCharacter::play_sound\(\)](#) and [diguyCharacter::create_sound_instance\(\)](#).

Arguments:

name	name of the new sound
sound_filename	filename of the sound
phoneme_filename	optional phoneme file; see diguySound::set_phoneme_filename()

Note that name isn't the filename of the sound; it is the descriptive name that will show up in the UI and used in [diguyCharacter](#) calls that play sounds.

Pass the filename in the `sound_filename` argument or use the function [diguySound::set_sound_filename\(\)](#) to set the filename of the sound.

See [diguyCharacter::set_current_voice_actor\(\)](#) for information about naming sounds for different voice actors.

Returns:

pointer of type [diguySound](#)

5.74.156 function [diguyScenario::destroy_sound](#)

Prototype:

```
int destroy_sound(diguySound* sound);
```

Description:

This function destroys a sound.

Arguments:

`sound` pointer to [diguySound](#) to be destroyed

Returns:

0 on success, -1 on failure

5.74.157 function diguyScenario::preload_sound**Prototype:**

```
int preload_sound(diguySound* sound);
```

Description:

This function preloads data for the sound.

Arguments:

sound pointer to [diguySound](#)

Returns:

0 on success, -1 on failure

5.74.158 function diguyScenario::stop_all_sounds**Prototype:**

```
int stop_all_sounds(float rampdown_time = 0.0f);
```

Description:

This function stops all sounds playing within the scenario.

Arguments:

rampdown_time how long to allow the sounds to fade out

Returns:

0 on success, -1 on failure

5.74.159 function diguyScenario::play_3d_sound**Prototype:**

```
int play_3d_sound(const char* name,
                  float x, float y, float z,
                  float gain,
                  int save_event);
```

Description:

This function plays a non looping sound.

Arguments:

name	name of the sound
x, y, z	location in the world the sound comes from
gain	a volume multiplier
save_event	if review data should record the creation of this sound

Returns:

0 on success, -1 on failure

Group Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.160 function diguyScenario::get_num_groups**Prototype:**

```
int get_num_groups();
```

Returns:

number of groups in the scenario

5.74.161 function diguyScenario::get_group_at_index**Prototype:**

```
diguyCharacterGroup* get_group_at_index(int index);
```

Returns:

pointer of type [diguyCharacterGroup](#); NULL if no group at the specified index

Arguments:

index	index of the group; indices start at 0

5.74.162 function diguyScenario::find_group**Prototype:**

```
diguyCharacterGroup* find_group(const char* name);
```

Description:

This function returns a pointer to the specified group.

Arguments:

name	name of group to be found

Returns:

pointer of type [diguyCharacterGroup](#); NULL if not found

5.74.163 function diguyScenario::create_group**Prototype:**

```
diguyCharacterGroup* create_group(const char* name);
```

Description:

This function creates a new group and returns a pointer to it.

Arguments:

name	name of the new group

Returns:

pointer of type [diguyCharacterGroup](#)

5.74.164 function diguyScenario::find_or_create_group**Prototype:**

```
diguyCharacterGroup* find_or_create_group(const char* name);
```

Description:

This function finds the group with the given name or creates it if it doesn't exist.

Arguments:

name	name of the group to find or create

Returns:

pointer of type diguyGroup; should never be NULL

5.74.165 function diguyScenario::destroy_group**Prototype:**

```
int destroy_group(diguyCharacterGroup* group);
```

Description:

This function destroys a group.

Arguments:

group	pointer to a diguyCharacterGroup

Returns:

0 on success, -1 on failure

Sensor Region Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.166 function diguyScenario::get_num_sensor_regions

Prototype:

```
int get_num_sensor_regions();
```

Returns:

number of sensor regions in the scenario

5.74.167 function diguyScenario::get_sensor_region_at_index**Prototype:**

```
diguySensorRegion* get_sensor_region_at_index(int index);
```

Returns:

pointer of type [diguySensorRegion](#); NULL if no sensor region at the specified index

Arguments:

index	index of the sensor region; indices start at 0

5.74.168 function diguyScenario::find_sensor_region**Prototype:**

```
diguySensorRegion* find_sensor_region(const char* name);
```

Description:

This function returns a pointer to the specified sensor region.

Arguments:

name	name of sensor region to be found

Returns:

pointer of type [diguySensorRegion](#); NULL if not found

Scene Object Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.169 function diguyScenario::get_num_scene_objects

Prototype:

```
int get_num_scene_objects();
```

Returns:

number of scene objects in the scenario

5.74.170 function diguyScenario::get_scene_object_at_index**Prototype:**

```
diguySceneObject* get_scene_object_at_index(int index);
```

Returns:

pointer of type [diguySceneObject](#); NULL if no scene object at the specified index

Arguments:

index	index of the scene object; indices start at 0

5.74.171 function diguyScenario::find_scene_object**Prototype:**

```
diguySceneObject* find_scene_object(const char* name);
```

Returns:

pointer of type [diguySceneObject](#); NULL if no scene object with specified name

5.74.172 function diguyScenario::set_scene_objects_enabled**Prototype:**

```
void set_scene_objects_enabled(int scene_objects_enabled);
```

Description:

This function sets whether scene objects defined in the scenario are enabled, and therefore potentially visible.

Scene objects are **not** enabled by default.

Arguments:

`scene_objects_enabled` flag determining whether scene objects are enabled; 1 for enabled, 0 for disabled

5.74.173 function diguyScenario::get_scene_objects_enabled**Prototype:**

```
int get_scene_objects_enabled();
```

Returns:

the most recent setting made by a call to `set_scene_objects_enabled()`

5.74.174 function diguyScenario::create_scene_object**Prototype:**

```
diguySceneObject* create_scene_object(const char* name,
                                     const char* filename);
```

Description:

This function creates a new scene object and returns a pointer to it.

Arguments:

name	name of the new scene object
filename	file the scene object should use

Returns:

pointer of type [diguySceneObject](#); NULL if creation failed

5.74.175 function diguyScenario::create_user_defined_scene_object**Prototype:**

```
diguySceneObject* create_user_defined_scene_object(const char* name);
```

Description:

This function creates a new user defined scene object and returns a pointer to it.

Arguments:

name	name of the new scene object

Returns:

pointer of type [diguySceneObject](#); NULL if creation failed

5.74.176 function `diguyScenario::destroy_scene_object`

Prototype:

```
int destroy_scene_object(diguySceneObject* scene_object);
```

Description:

This function destroys a scene object.

Arguments:

`scene_object` pointer to a `diguySceneObject`

Returns:

0 on success, -1 on failure

View Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.177 function diguyScenario::get_primary_view**Prototype:**

```
diguyView* get_primary_view();
```

Returns:

pointer to the scenario's primary view; there is one and only one primary view in the scenario

5.74.178 function diguyScenario::get_num_secondary_views**Prototype:**

```
int get_num_secondary_views();
```

Returns:

number of secondary views in the scenario

5.74.179 function diguyScenario::get_secondary_view_at_index**Prototype:**

```
diguyView* get_secondary_view_at_index(int index);
```

Returns:

pointer of type **diguyView**; NULL if no secondary view at the specified index

Arguments:

index index of the secondary view; indices start at 0

5.74.180 function diguyScenario::find_view**Prototype:**

```
diguyView* find_view(const char* name);
```

Description:

This function returns a pointer to the view with the given name. The primary view and all of the secondary views are checked for a name match.

Returns:

pointer of type **diguyView**; NULL if not found

Camera Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.181 function diguyScenario::get_scenario_camera

Prototype:

```
diguyViewCamera* get_scenario_camera();
```

Description:

This function returns a pointer to the primary view's camera. This is equivalent to:

```
diguyView* v = scenario->get_primary_view();  
return v->get_camera();
```

Returns:

pointer of type [diguyViewCamera](#); should never be NULL

5.74.182 function diguyScenario::find_camera**Prototype:**

```
diguyViewCamera* find_camera(const char* name);
```

Description:

This function returns a pointer to the camera with the given name. The primary view's camera and all of the secondary views' cameras are checked for a name match.

Returns:

pointer of type [diguyViewCamera](#); NULL if not found

5.74.183 function diguyScenario::set_scenario_camera_affected_by_script_events**Prototype:**

```
void set_scenario_camera_affected_by_script_events(int flag);
```

Description:

This function sets a flag that determines whether script calls can set the primary view's camera settings. Specifically, if flag is 0, calls to `diguyViewCamera::load( )` will have no effect; the camera should remain completely under user control.

Arguments:

<code>flag</code>	1 to enable, 0 to disable

5.74.184 function**diguyScenario::get_scenario_camera_affected_by_script_events****Prototype:**

```
int get_scenario_camera_affected_by_script_events();
```

Returns:

the most recent setting made by

[set_scenario_camera_affected_by_script_events\(\)](#)

5.74.185 function**diguyScenario::set_scenario_camera_affected_by_script_events_wh****Prototype:**

```
void set_scenario_camera_affected_by_script_events_when_iguy_active(int flag);
```

Description:

This function, like

[set_scenario_camera_affected_by_script_events\(\)](#),

determines whether script calls can set the primary view's camera settings,

but only disables camera changes if there is an active I-Guy character as set by the [set_iguy_character\(\)](#) call.

If [set_scenario_camera_affected_by_script_events\(\)](#) has been called with a value of 0, then this function has no effect.

Arguments:

flag	1 to enable, 0 to disable

5.74.186 function**diguyScenario::get_scenario_camera_affected_by_script_events_wh****Prototype:**

```
int get_scenario_camera_affected_by_script_events_when_iguy_active();
```

Returns:

the most recent setting made by

[set_scenario_camera_affected_by_script_events\(\)](#)

5.74.187 function diguyScenario::get_num_camera_settings**Prototype:**

```
int get_num_camera_settings();
```

Returns:

number of camera settings saved in the scenario

5.74.188 function diguyScenario::get_camera_settings_at_index**Prototype:**

```
diguyViewCameraSettings* get_camera_settings_at_index(int index);
```

Returns:

pointer of type [diguyViewCameraSettings](#); NULL if no camera settings at the specified index

Arguments:

index	index of the camera settings; indices start at 0

5.74.189 function diguyScenario::find_camera_settings**Prototype:**

```
diguyViewCameraSettings* find_camera_settings(const char* settings_name);
```

Description:

This function returns a pointer to the specified camera settings.

Arguments:

settings_name	name of camera settings to be found

Returns:

pointer of type [diguyViewCamera](#); NULL if not found

5.74.190 function diguyScenario::load_camera_settings**Prototype:**

```
int load_camera_settings(const char* settings_name, int update_current_camera = 1);
```

Description:

This function loads the specified camera settings into the primary view's camera.

Arguments:

<code>settings_name</code>	name of camera settings to be loaded
<code>update_current_camera</code>	whether the current camera should be updated; defaults to 1

Not updating the current camera will cause the camera history to be lost.

Returns:

0 on success, -1 on failure

5.74.191 function diguyScenario::set_render_camera**Prototype:**

```
void set_render_camera(diguyViewCamera* camera);
```

Description:

This function sets camera that will be used for down-stream camera-related operations. This includes:

- ◊ far position rendering in the OpenGL renderer
- ◊ various culling operations (e.g. `set_character_culling_enabled()`)
- ◊ graphics LOD range scaling; see `set_scale_graphics_lod_ranges_from_view_settings()`

Typically the passed camera should be the primary view's camera.

A render camera does not always need to be set. If one is not set, the above camera-related operations will not be performed.

If a render camera is set the actual rendering environment camera settings (e.g. as set by `gluLookAt()` in OpenGL) should match the settings in the render camera, or there may be unexpected visual results.

Arguments:

`camera` camera to use for camera-related operations

Fog Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.192 function diguyScenario::get_scenario_fog**Prototype:**

```
diguyViewFog* get_scenario_fog();
```

Description:

This function returns a pointer to the scenario fog. This is the fog of the primary view.

Returns:

pointer of type [diguyViewFog](#); should never be NULL

5.74.193 function diguyScenario::find_fog**Prototype:**

```
diguyViewFog* find_fog(const char* name);
```

Description:

This function returns a pointer to the fog with the given name. The primary view's fog and all of the secondary views' fogs are checked for a name match.

Returns:

pointer of type [diguyViewFog](#); NULL if not found

5.74.194 function diguyScenario::find_or_create_fog**Prototype:**

```
diguyViewFog* find_or_create_fog(const char* name);
```

Description:

This function returns the fog object with the given name, or creates one if not found.

Returns:

pointer of type [diguyViewFog](#)

name

5.74.195 function diguyScenario::load_current_fog**Prototype:**

```
void load_current_fog(const char* name);
```

Description:

This function loads the named fog settings the currently active one.

name

5.74.196 function diguyScenario::get_num_fog_settings**Prototype:**

```
int get_num_fog_settings();
```

Returns:

number of fog settings in the scenario

5.74.197 function diguyScenario::get_fog_settings_at_index**Prototype:**

```
diguyViewFogSettings* get_fog_settings_at_index(int index);
```

Returns:

pointer of type [diguyViewFogSettings](#); NULL if no fog at the specified index

Arguments:

index	index of the fog settings; indices start at 0

5.74.198 function diguyScenario::find_fog_settings**Prototype:**

```
diguyViewFogSettings* find_fog_settings(const char* settings_name);
```

Description:

This function returns a pointer to the specified fog settings.

Arguments:

`settings_name` name of fog settings to be found

Returns:

pointer of type [diguyViewFogSettings](#); NULL if not found

5.74.199 function diguyScenario::load_fog_settings**Prototype:**

```
int load_fog_settings(const char* settings_name);
```

Description:

This function loads the specified fog settings into the scenario fog.

Arguments:

`settings_name` name of fog settings to be loaded

Returns:

pointer of type [diguyViewFog](#); NULL if not found

Light Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.200 function diguyScenario::get_scenario_light

Prototype:

```
diguyViewLight* get_scenario_light(int i = 0);
```

Description:

This function returns a pointer to the scenario light. These are the lights of the primary view.

Arguments:

number	number of light to be found

Returns:

pointer of type `diguyViewLight`; can be null if asked for a non-existent light

5.74.201 function diguyScenario::find_light**Prototype:**

```
diguyViewLight* find_light(const char* name);
```

Description:

This function returns a pointer to the light with the given name. The primary view's light and all of the secondary views' lights are checked for a name match.

Returns:

pointer of type [diguyViewLight](#); NULL if not found

5.74.202 function diguyScenario::get_num_light_settings**Prototype:**

```
int get_num_light_settings();
```

Returns:

number of light settings in the scenario

5.74.203 function diguyScenario::get_light_settings_at_index**Prototype:**

```
diguyViewLightSettings* get_light_settings_at_index(int index);
```

Returns:

pointer of type [diguyViewLightSettings](#); NULL if no light settings at the specified index

Arguments:

index	index of the light settings; indices start at 0

5.74.204 function diguyScenario::find_light_settings**Prototype:**

```
diguyViewLightSettings* find_light_settings(const char* settings_name);
```

Description:

This function returns a pointer to the specified light settings.

Arguments:

name	name of light settings to be found

Returns:

pointer of type [diguyViewLightSettings](#); NULL if not found

5.74.205 function diguyScenario::load_light_settings

Prototype:

```
int load_light_settings(const char* settings_name, int light_num = 0);
```

Description:

This function loads the specified light settings into the scenario light.

Arguments:

<code>settings_name</code>	name of light settings to be loaded
<code>light_num</code>	which light to load into

Returns:

pointer of type `diguyViewLight`; NULL if not found

Info Popup Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.206 function diguyScenario::get_num_info_popups**Prototype:**

```
int get_num_info_popups();
```

Returns:

number of info popups in the scenario

5.74.207 function diguyScenario::get_info_popup_at_index**Prototype:**

```
diguyInfoPopup* get_info_popup_at_index(int index);
```

Returns:

pointer of type [diguyInfoPopup](#); NULL if no info popup at the specified index

Arguments:

<code>index</code>	index of the info popup; indices start at 0

5.74.208 function diguyScenario::find_info_popup**Prototype:**

```
diguyInfoPopup* find_info_popup(const char* name);
```

Description:

This function returns a pointer to the specified info popup.

Arguments:

name	name of info popup to be found

Returns:

pointer of type [diguyInfoPopup](#); NULL if not found

5.74.209 function diguyScenario::set_info_popup_default_encoding**Prototype:**

```
int set_info_popup_default_encoding(const char* info_popup_default_encoding);
```

Description:

This function sets the default encoding for info popups that do not have an encoding specified.

Arguments:

info_popup_default_encoding	new default encoding

Returns:

0 on success, -1 on failure

5.74.210 function

diguyScenario::get_info_popup_default_encoding

Prototype:

```
const char* get_info_popup_default_encoding();
```

Description:

This function returns the default encoding of info popups.

Returns:

default encoding; value will never be NULL, but may be the empty string ("") if no default encoding has been specified

Variable Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.211 function diguyScenario::get_num_variables**Prototype:**

```
int get_num_variables();
```

Returns:

number of variables in the scenario

5.74.212 function diguyScenario::get_variable_at_index**Prototype:**

```
diguyVariable* get_variable_at_index(int index);
```

Returns:

pointer of type [diguyVariable](#); NULL if no variable at the specified index

Arguments:

index	index of the variable; indices start at 0

5.74.213 function diguyScenario::find_variable**Prototype:**

```
diguyVariable* find_variable(const char* name);
```

Description:

This function returns a pointer to the specified variable.

Arguments:

name	name of variable to be found

Returns:

pointer of type [diguyVariable](#); NULL if not found

5.74.214 function diguyScenario::find_or_create_variable**Prototype:**

```
diguyVariable* find_or_create_variable(const char* name);
```

Description:

This function finds the variable with the given name or creates it if it doesn't exist.

Arguments:

name	name of the variable to find or create

Returns:

pointer of type [diguyVariable](#); should never be NULL

5.74.215 function diguyScenario::destroy_variable

Prototype:

```
int destroy_variable(diguyVariable* variable);
```

Description:

This function destroys the passed variable.

Arguments:

`variable` pointer to a `diguyVariable`

Returns:

0 on success, -1 on failure

Face Expression Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.216 function diguyScenario::get_num_face_expressions**Prototype:**

```
int get_num_face_expressions();
```

Returns:

number of face_expressions in the scenario

5.74.217 function diguyScenario::get_face_expression_at_index**Prototype:**

```
diguyCharacterFaceExpression* get_face_expression_at_index(int index);
```

Returns:

pointer of type [diguyCharacterFaceExpression](#); NULL if no info
popup at the specified index

Arguments:

index	index of the face_expression; indices start at 0

5.74.218 function diguyScenario::find_face_expression**Prototype:**

```
diguyCharacterFaceExpression* find_face_expression(const char* name);
```

Description:

This function returns a pointer to the specified face expression.

Arguments:

name	name of face_expression to be found

Returns:

pointer of type [diguyCharacterFaceExpression](#); NULL if not found

5.74.219 function diguyScenario::create_face_expression**Prototype:**

```
diguyCharacterFaceExpression* create_face_expression(const char* name,
const char* target_actor = "exface");
```

Description:

This function creates a new face expression with the given name.

Arguments:

name	name of the new face expression
target_actor	name of the actor to use as template; defaults to "exface" for backwards compatibility

Returns:

pointer of type [diguyCharacterFaceExpression](#)

5.74.220 function diguyScenario::destroy_face_expression**Prototype:**

```
int destroy_face_expression(diguyCharacterFaceExpression* face_expression);
```

Description:

This function destroys the passed face expression.

Arguments:

`face_expression` pointer to a `diguyCharacterFaceExpression`

Returns:

0 on success, -1 on failure

Chain Settings Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.221 function diguyScenario::get_num_chain_settings**Prototype:**

```
int get_num_chain_settings();
```

Returns:

number of chain settings in the scenario

5.74.222 function diguyScenario::get_chain_settings_at_index**Prototype:**

```
diguyChainSettings* get_chain_settings_at_index(int index);
```

Returns:

pointer of type [diguyChainSettings](#); NULL if no chain settings at the specified index

Arguments:

index	index of the chain settings; indices start at 0

5.74.223 function diguyScenario::find_chain_settings**Prototype:**

```
diguyChainSettings* find_chain_settings(const char* name);
```

Description:

This function returns a pointer to the specified chain settings.

Arguments:

name	name of chain settings to be found

Returns:

pointer of type [diguyChainSettings](#); NULL if not found

5.74.224 function diguyScenario::create_chain_settings**Prototype:**

```
diguyChainSettings* create_chain_settings(const char* name);
```

Description:

This function creates a new chain settings and returns a pointer to it.

Arguments:

name	name of the new chain settings

Returns:

pointer of type [diguyChainSettings](#)

5.74.225 function diguyScenario::destroy_chain_settings**Prototype:**

```
int destroy_chain_settings(diguyChainSettings* chain_settings);
```

Description:

This function destroys a chain settings.

Arguments:

chain settings	pointer to a diguyChainSettings

Returns:

0 on success, -1 on failure

Callback Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.226 enumeration

```
enum {
    CALLBACK_ID_CREATE = 1,
    CALLBACK_ID_DESTROY,
    CALLBACK_ID_RESET,
    CALLBACK_ID_WAIT_CURSOR_SHOW,
    CALLBACK_ID_WAIT_CURSOR_HIDE,
    CALLBACK_ID_LOAD,
    CALLBACK_ID_SAVE,
    CALLBACK_ID_LOAD_SCENARIO_FILE,
    CALLBACK_ID_SAVE_SCENARIO_FILE,
    CALLBACK_ID_SCENE_OBJECT_IMPACT,
    CALLBACK_ID_TIMED_EVENT,
    CALLBACK_ID_MANUALLY_INVOKED,
    CALLBACK_ID_POST_DRAW,
    CALLBACK_ID_PLAYBACK_MODE_CHANGED,
    CALLBACK_ID_INPUT_MODE_CHANGED,
```

DI-Guy SDK Version 12.5.2 Index

```
CALLBACK_ID_RENDER_MODE_CHANGED,  
CALLBACK_ID_PRE_REINITIALIZE,  
CALLBACK_ID_POST_REINITIALIZE,  
CALLBACK_ID_POST_LOAD_CHECKPOINT  
};
```

Description:

This is an enumeration of the different callbacks that can be registered with `add_callback()` and `add_callback_script()`.

Callbacks return a value of type `diguyCallbackReturn`, which will be `DIGUY_CALLBACK_STOP` or `DIGUY_CALLBACK_CONTINUE`. If the callback returns `DIGUY_CALLBACK_STOP`, the default handler of the function will not be called; the callback is asserting that it has done everything necessary for the function call. If the callback returns `DIGUY_CALLBACK_CONTINUE`, the default handler for the function will be called after the callback.

Callback Enums:

CALLBACK_ID_CREATE

This callback will be called when a new scenario is created. It should only be used by `diguyApp::add_default_scenario_callback()`. Using it in `set_callback()` will have no effect, as by that time the scenario has already been created.

CALLBACK_ID_RESET

This callback will be called when the scenario is reset.

CALLBACK_ID_WAIT_CURSOR_SHOW

This callback will be called when a diguy operation is likely to take some time, allowing an application to display a wait cursor.

CALLBACK_ID_WAIT_CURSOR_HIDE

This callback will be called when a diguy operation that caused a wait cursor to be shown has completed, allowing an application to hide the wait cursor.

CALLBACK_ID_LOAD

This callback will be called when a scenario is loaded, thereby giving a chance for custom code to read extra elements from the loaded file.

CALLBACK_ID_SAVE

This callback will be called when a scenario is saved, thereby giving a chance for custom code to insert extra elements into the saved file.

CALLBACK_ID_LOAD_SCENARIO_FILE

This callback will be called just before a scenario loads a new file.

CALLBACK_ID_SAVE_SCENARIO_FILE

This callback will be called just after a scenario saves a new file. With a char * pointer to the file name.

CALLBACK_ID_MANUALLY_INVOKED

This callback id will be supplied to event handlers invoked by a call to `manually_invoke_event_handler()`.

The user's returned `diguyCallbackReturn` value will be ignored.

CALLBACK_ID_POST_DRAW

This callback will be called just after a scenario finishes its draw commands. The `diguyViewPainter` class can be used to issue abstract draw commands in DI-Guy Scenario.

Lua Example:

```
local painter = this_app:get_view_painter();
painter:set_pen_color(1,1,0);
painter:draw_line(0,0,0, 1,1,1);
```

CALLBACK_ID_PLAYBACK_MODE_CHANGED

This callback will be called when the playback mode of the scenario has changed. For example, the Play or Stop buttons are pressed in DI-Guy Scenario, or `set_playback_mode()` is called.

The user's returned `diguyCallbackReturn` value will be ignored.

CALLBACK_ID_INPUT_MODE_CHANGED

This callback will be called when the input mode of the scenario has changed. For example, when an Input Mode button is pressed in DI-Guy Scenario, or `diguyApp::set_base_input_mode()` is called.

The user's returned `diguyCallbackReturn` value will be ignored.

CALLBACK_ID_RENDER_MODE_CHANGED

This callback will be called if the render mode of the scenario changes, typically due to a `set_current_render_mode()`.

The user's returned `diguyCallbackReturn` value will be ignored.

CALLBACK_ID_POST_LOAD_CHECKPOINT

This callback will be called just after a scenario finishes loading a checkpoint file.

5.74.227 function diguyScenario::add_callback

Prototype:

```
int add_callback(int callback_id,
                diguyScenarioCallback* callback,
                void* callback_params = 0,
                void* callback_user_data = 0,
                int remove_on_scenario_load = 1);
```

Description:

This function adds a scenario callback.

Arguments:

callback	pointer to function with prototype diguyScenarioCallback (typedefed above)
callback_id	integer id of when this callback is to be called
callback_params	not currently used; pass NULL
callback_user_data	

pointer for user's own use; DI-Guy will do nothing to the contents of this pointer beyond passing it back when the callback is invoked

`remove_on_scenario_load` pass 1 to remove the callback on a scenario load, 0 to not

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.74.228 function `diguyScenario::remove_callback`

Prototype:

```
int remove_callback(int callback_id,
    diguyScenarioCallback* callback);
```

Description:

This function removes a user callback. All callbacks matching the specified `callback_id` and callback function will be removed.

Arguments:

`callback_id` integer id of callback
`callback` pointer to function with prototype `diguyCharacterCallback` (typedefed above)

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.74.229 function diguyScenario::remove_callback_with_user_data

Prototype:

```
int remove_callback_with_user_data(int callback_id,
    void* callback_user_data);
```

Description:

This function removes a user callback. All callbacks matching the specified `callback_id` and `callback_user_data` pointer will be removed.

Arguments:

<code>callback_id</code>	integer id of callback
<code>callback_user_data</code>	pointer for user's own use

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.74.230 function
diguyScenario::add_default_character_callback**Prototype:**

```
int add_default_character_callback(int callback_id,  
    diguyCharacterCallback* callback,  
    void* callback_params,  
    void* callback_user_data,  
    int add_to_existing_objects_flag = 0,  
    int remove_on_scenario_load = 1);
```

Description:

This function sets a default user callback that will be added to all new characters. See [diguyCharacter::add_callback\(\)](#) for details.

Callable From:

◇ C++

5.74.231 function**diguyScenario::remove_default_character_callback****Prototype:**

```
int remove_default_character_callback(int callback_id,
                                     diguyCharacterCallback* callback);
```

Description:

This function removes a user callback. All default character callbacks matching the specified `callback_id` and callback function will be removed.

Arguments:

<code>callback_id</code>	integer id of callback
<code>callback</code>	pointer to function with prototype <code>diguyCharacterCallback</code> (typedefed above)

Callable From:

◇ C++

5.74.232 function**diguyScenario::remove_default_character_callback_with_user_data****Prototype:**

```
int remove_default_character_callback_with_user_data(int callback_id,
                                                     void* callback_user_data);
```

Description:

This function removes a default user callback previously added by [add_default_character_callback\(\)](#). See [diguyCharacter::remove_callback_with_user_data\(\)](#) for details.

Callable From:

◇ C++

5.74.233 function diguyScenario::add_default_path_callback**Prototype:**

```
int add_default_path_callback(int callback_id,
    diguyCharacterPathCallback* callback,
    void* callback_params,
    void* callback_user_data,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
```

Description:

This function add a default user callback that will be added to all new paths.

Callable From:

◇ C++

5.74.234 function diguyScenario::remove_default_path_callback**Prototype:**

```
int remove_default_path_callback(int callback_id,
    diguyCharacterPathCallback* callback);
```

Description:

This function removes a user callback. All default path callbacks matching the specified callback_id and callback function will be removed.

Arguments:

callback_id	integer id of callback
callback	pointer to function with prototype diguyCharacterCallback (typedefed above)

Callable From:

◇ C++

5.74.235 function**diguyScenario::remove_default_path_callback_with_user_data****Prototype:**

```
int remove_default_path_callback_with_user_data(int callback_id,
void* callback_user_data);
```

Description:

This function removes a default user callback. All default path callbacks matching the specified `callback_id` and `callback_user_data` pointer will be removed.

Arguments:

<code>callback_id</code>	integer id of callback
<code>callback_user_data</code>	pointer for user's own use

Callable From:

◇ C++

5.74.236 function**diguyScenario::add_default_sensor_region_callback****Prototype:**

```
int add_default_sensor_region_callback(int callback_id,
diguySensorRegionCallback* callback,
void* callback_params,
void* callback_user_data,
int add_to_existing_objects_flag = 0,
int remove_on_scenario_load = 1);
```

Description:

This function sets a default user callback that will be added to all new sensor regions. See [diguySensorRegion::add_callback\(\)](#) for details.

Callable From:

◇ C++

5.74.237 function**diguyScenario::remove_default_sensor_region_callback****Prototype:**

```
int remove_default_sensor_region_callback(int callback_id,
    diguySensorRegionCallback* callback);
```

Description:

This function removes a user callback. All default sensor region callbacks matching the specified callback_id and callback function will be removed.

Arguments:

callback_id	integer id of callback
callback	pointer to function with prototype diguyCharacterCallback (typedefed above)

Callable From:

◇ C++

5.74.238 function**diguyScenario::remove_default_sensor_region_callback_with_us****Prototype:**

```
int remove_default_sensor_region_callback_with_user_data(int callback_id,
    void* callback_user_data);
```

Description:

This function removes a default user callback. All default sensor region callbacks matching the specified callback_id and callback_user_data pointer will be removed.

Arguments:

callback_id	integer id of callback
callback_user_data	pointer for user's own use

Callable From:

◇ C++

5.74.239 function diguyScenario::add_default_signal_callback**Prototype:**

```
int add_default_signal_callback(int callback_id,
    diguySignalCallback* callback,
    void* callback_params,
    void* callback_user_data,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
```

Description:

This function adds a default user callback that will be added to all new signals.

Callable From:

◇ C++

5.74.240 function diguyScenario::remove_default_signal_callback**Prototype:**

```
int remove_default_signal_callback(int callback_id,
    diguySignalCallback* callback);
```

Description:

This function removes a user callback. All default signal callbacks matching the specified callback_id and callback function will be removed.

Arguments:

callback_id	integer id of callback
callback	pointer to function with prototype diguyCharacterCallback (typedefed above)

Callable From:

◇ C++

5.74.241 function**diguyScenario::remove_default_signal_callback_with_user_data****Prototype:**

```
int remove_default_signal_callback_with_user_data(int callback_id,
void* callback_user_data);
```

Description:

This function removes a default user callback. All default signal callbacks matching the specified `callback_id` and `callback_user_data` pointer will be removed.

Arguments:

<code>callback_id</code>	integer id of callback
<code>callback_user_data</code>	pointer for user's own use

Callable From:

◇ C++

5.74.242 function diguyScenario::add_default_variable_callback**Prototype:**

```
int add_default_variable_callback(int callback_id,
diguyVariableCallback* callback,
void* callback_params,
void* callback_user_data,
int add_to_existing_objects_flag = 0,
int remove_on_scenario_load = 1);
```

Description:

This function adds a default user callback that will be added to all new variables.

Callable From:

◇ C++

5.74.243 function**diguyScenario::remove_default_variable_callback****Prototype:**

```
int remove_default_variable_callback(int callback_id,
    diguyVariableCallback* callback);
```

Description:

This function removes a user callback. All default variable callbacks matching the specified callback_id and callback function will be removed.

Arguments:

callback_id integer id of callback
 callback pointer to function with prototype diguyCharacterCallback (typedefed above)

Callable From:

◇ C++

5.74.244 function**diguyScenario::remove_default_variable_callback_with_user_data****Prototype:**

```
int remove_default_variable_callback_with_user_data(int callback_id,
    void* callback_user_data);
```

Description:

This function removes a default user callback. All default variable callbacks matching the specified callback_id and callback_user_data pointer will be removed.

Arguments:

callback_id integer id of callback
 callback_user_data pointer for user's own use

Callable From:

◇ C++

5.74.245 function diguyScenario::add_default_view_callback**Prototype:**

```
int add_default_view_callback(int callback_id,
    diguyViewCallback* callback,
    void* callback_params,
    void* callback_user_data,
    int add_to_existing_objects_flag = 1,
    int remove_on_scenario_load = 1);
```

Description:

This function adds a default user callback that will be added to all new views.

Note that unlike most of the other add default callback functions, this one's `add_to_existing_objects_flag` argument defaults to a value of 1, since views in scenarios always exist and are neither created nor destroyed.

Callable From:

◇ C++

5.74.246 function diguyScenario::remove_default_view_callback**Prototype:**

```
int remove_default_view_callback(int callback_id,
    diguyViewCallback* callback);
```

Description:

This function removes a user callback. All default view callbacks matching the specified `callback_id` and callback function will be removed.

Arguments:

<code>callback_id</code>	integer id of callback
<code>callback</code>	pointer to function with prototype <code>diguyCharacterCallback</code> (typedefed above)

Callable From:

◇ C++

5.74.247 function**diguyScenario::remove_default_view_callback_with_user_data****Prototype:**

```
int remove_default_view_callback_with_user_data(int callback_id,
void* callback_user_data);
```

Description:

This function removes a default user callback. All default view callbacks matching the specified callback_id and callback_user_data pointer will be removed.

Arguments:

callback_id	integer id of callback
callback_user_data	pointer for user's own use

Callable From:

◇ C++

5.74.248 function diguyScenario::add_default_camera_callback**Prototype:**

```
int add_default_camera_callback(int callback_id,
diguyViewCameraCallback* callback,
void* callback_params,
void* callback_user_data,
int add_to_existing_objects_flag = 0,
int remove_on_scenario_load = 1);
```

Description:

This function adds a default user callback that will be added to all cameras.

Callable From:

◇ C++

5.74.249 function**diguyScenario::remove_default_camera_callback****Prototype:**

```
int remove_default_camera_callback(int callback_id,
    diguyViewCameraCallback* callback);
```

Description:

This function removes a user callback. All default camera callbacks matching the specified callback_id and callback function will be removed.

Arguments:

callback_id	integer id of callback
callback	pointer to function with prototype diguyCharacterCallback (typedefed above)

Callable From:

◇ C++

5.74.250 function**diguyScenario::remove_default_camera_callback_with_user_data****Prototype:**

```
int remove_default_camera_callback_with_user_data(int callback_id,
    void* callback_user_data);
```

Description:

This function removes a default user callback. All default camera callbacks matching the specified callback_id and callback_user_data pointer will be removed.

Arguments:

callback_id	integer id of callback
callback_user_data	pointer for user's own use

Callable From:

◇ C++

5.74.251 function diguyScenario::add_default_fog_callback**Prototype:**

```
int add_default_fog_callback(int callback_id,
    diguyViewFogCallback* callback,
    void* callback_params,
    void* callback_user_data,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
```

Description:

This function adds a default user callback that will be added to all new fogs.

Callable From:

◇ C++

5.74.252 function diguyScenario::remove_default_fog_callback**Prototype:**

```
int remove_default_fog_callback(int callback_id,
    diguyViewFogCallback* callback);
```

Description:

This function removes a user callback. All default fog callbacks matching the specified callback_id and callback function will be removed.

Arguments:

callback_id	integer id of callback
callback	pointer to function with prototype diguyCharacterCallback (typedefed above)

Callable From:

◇ C++

5.74.253 function**diguyScenario::remove_default_fog_callback_with_user_data****Prototype:**

```
int remove_default_fog_callback_with_user_data(int callback_id,
void* callback_user_data);
```

Description:

This function removes a default user callback. All default fog callbacks matching the specified callback_id and callback_user_data pointer will be removed.

Arguments:

callback_id	integer id of callback
callback_user_data	pointer for user's own use

Callable From:

◇ C++

5.74.254 function diguyScenario::add_default_light_callback**Prototype:**

```
int add_default_light_callback(int callback_id,
diguyViewLightCallback* callback,
void* callback_params,
void* callback_user_data,
int add_to_existing_objects_flag = 0,
int remove_on_scenario_load = 1);
```

Description:

This function adds a default user callback that will be added to all new lights.

Callable From:

◇ C++

5.74.255 function diguyScenario::remove_default_light_callback**Prototype:**

```
int remove_default_light_callback(int callback_id,
    diguyViewLightCallback* callback);
```

Description:

This function removes a user callback. All default light callbacks matching the specified callback_id and callback function will be removed.

Arguments:

callback_id	integer id of callback
callback	pointer to function with prototype diguyCharacterCallback (typedefed above)

Callable From:

◇ C++

5.74.256 function diguyScenario::remove_default_light_callback_with_user_data**Prototype:**

```
int remove_default_light_callback_with_user_data(int callback_id,
    void* callback_user_data);
```

Description:

This function removes a default user callback. All default light callbacks matching the specified callback_id and callback_user_data pointer will be removed.

Arguments:

callback_id	integer id of callback
callback_user_data	pointer for user's own use

Callable From:

◇ C++

5.74.257 function `diguyScenario::manually_invoke_callbacks_now`

Prototype:

```
void manually_invoke_callbacks_now(int callback_id);
```

Description:

This function causes all of the scenario callbacks with the given `callback_id` to be called now.

Arguments:

`callback_id` integer id of callback

5.74.258 function `diguyScenario::add_callback_script`

Prototype:

```
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL,
    int remove_on_scenario_load = 1);
```

Description:

This function adds a user callback script. Callback scripts can be removed with [remove_callback_script\(\)](#).

See [diguyCharacter::add_callback_script\(\)](#) for an example of use.

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	Script text of callback to be added
<code>callback_script_type</code>	

the type of script contained in
callback_script

`remove_on_scenario_load` if the callback is removed when a new
scenario is loaded

If NULL is passed for `callback_script_type`, a default script type will be
derived based on the default script interpreter of the scenario.

Lua specific:

When the script is called, the object for which it is being called will be in the
callback_object global.

To pass NULL when calling from a lua script, use nil.

Returns:

0 on success, -1 on failure

5.74.259 function diguyScenario::remove_callback_script

Prototype:

```
int remove_callback_script(int callback_id,  
    const char* callback_script,  
    const char* callback_script_type = NULL);
```

Description:

This function removes a user callback script previously added with
[add_callback_script\(\)](#).

See [diguyCharacter::remove_callback_script\(\)](#) for an
example of use.

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	Script text of callback previously added
<code>callback_script_type</code>	the type of script contained in callback_script

If NULL is passed for `callback_script`, all callback scripts whose ids match `callback_id` and whose types match `callback_script_type` will be removed.

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

Lua specific:

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

5.74.260 function `diguyScenario::add_default_character_callback_script`

Prototype:

```
int add_default_character_callback_script(int callback_id,  
    const char* callback_script,  
    const char* callback_script_type = NULL,  
    int add_to_existing_objects_flag = 0,  
    int remove_on_scenario_load = 1);
```

Description:

This function adds a default character callback script. The callback script will be added to all new characters. See

[`diguyCharacter::add\_callback\_script\(\)`](#) for more details.

Returns:

0 on success, -1 on failure

5.74.261 function**diguyScenario::remove_default_character_callback_script****Prototype:**

```
int remove_default_character_callback_script(int callback_id,
      const char* callback_script,
      const char* callback_script_type = NULL);
```

Description:

This function removes a default character callback script. See [diguyCharacter::remove_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

5.74.262 function**diguyScenario::add_default_sensor_region_callback_script****Prototype:**

```
int add_default_sensor_region_callback_script(int callback_id,
      const char* callback_script,
      const char* callback_script_type = NULL,
      int add_to_existing_objects_flag = 0,
      int remove_on_scenario_load = 1);
```

Description:

This function adds a default sensor region callback script. The callback script will be added to all new sensor regions. See [diguySensorRegion::add_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

5.74.263 function**diguyScenario::remove_default_sensor_region_callback_script****Prototype:**

```
int remove_default_sensor_region_callback_script(int callback_id,
const char* callback_script,
const char* callback_script_type = NULL);
```

Description:

This function removes a default sensor region callback script. See [diguySensorRegion::remove_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

5.74.264 function**diguyScenario::add_default_signal_callback_script****Prototype:**

```
int add_default_signal_callback_script(int callback_id,
const char* callback_script,
const char* callback_script_type = NULL,
int add_to_existing_objects_flag = 0,
int remove_on_scenario_load = 1);
```

Description:

This function adds a default signal callback script. The callback script will be added to all new signals. See [diguySignal::add_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

5.74.265 function**diguyScenario::remove_default_signal_callback_script****Prototype:**

```
int remove_default_signal_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function removes a default signal callback script. See [diguySignal::remove_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

5.74.266 function**diguyScenario::add_default_variable_callback_script****Prototype:**

```
int add_default_variable_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
```

Description:

This function adds a default variable callback script. The callback script will be added to all new variables. See [diguySignal::add_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

5.74.267 function**diguyScenario::remove_default_variable_callback_script****Prototype:**

```
int remove_default_variable_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function removes a default variable callback script. See [diguySignal::remove_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

5.74.268 function**diguyScenario::add_default_view_callback_script****Prototype:**

```
int add_default_view_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
```

Description:

This function adds a default view callback script. The callback script will be added to all new views. See [diguyView::add_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

5.74.269 function**diguyScenario::remove_default_view_callback_script****Prototype:**

```
int remove_default_view_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function removes a default character view script. See [diguyView::remove_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

5.74.270 function**diguyScenario::add_default_camera_callback_script****Prototype:**

```
int add_default_camera_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
```

Description:

This function adds a default camera callback script. The callback script will be added to all new cameras. See [diguyViewCamera::add_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

5.74.271 function**diguyScenario::remove_default_camera_callback_script****Prototype:**

```
int remove_default_camera_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function removes a default camera callback script. See [diguyViewCamera::remove_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

5.74.272 function**diguyScenario::add_default_fog_callback_script****Prototype:**

```
int add_default_fog_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
```

Description:

This function adds a default fog callback script. The callback script will be added to all new fogs. See [diguyViewFog::add_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

5.74.273 function**diguyScenario::remove_default_fog_callback_script****Prototype:**

```
int remove_default_fog_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function removes a default fog callback script. See [diguyViewFog::remove_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

5.74.274 function**diguyScenario::add_default_light_callback_script****Prototype:**

```
int add_default_light_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL,
    int add_to_existing_objects_flag = 0,
    int remove_on_scenario_load = 1);
```

Description:

This function adds a default light callback script. The callback script will be added to all new lights. See [diguyViewLight::add_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

5.74.275 function**diguyScenario::remove_default_light_callback_script****Prototype:**

```
int remove_default_light_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function removes a default light callback script. See [diguyViewLight::remove_callback_script\(\)](#) for more details.

Returns:

0 on success, -1 on failure

Event Handler Functions

DI-Guy event handlers are similar to callbacks, but have a name with which they can be referenced. This name allows DI-Guy Scenario to map event handlers to callback ids using a UI, and allows for these mappings to be saved and restored as part of the scenario.

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.276 function**diguyScenario::map_event_handler_to_callback_id****Prototype:**

```
int map_event_handler_to_callback_id(int callback_id,
    const char* handler_name);
```

Description:

This function maps the event handler with the given name to a callback id. This mapping will be saved in the .dss file and restored when the .dss file is loaded.

Mappings can also be made via the DI-Guy Scenario UI.

The event handler is one of the following:

- ◇ a scenario callback function registered by
`diguyApp::register_scenario_event_handler()` or
`diguyApp::register_scenario_event_handler_from_library()`
- ◇ a script registered by
`diguyApp::register_scenario_event_handler_script()`
- ◇ a Script, Decision, or Library Function in the scenario whose "Event Type" is "Scenario"

Arguments:

`callback_id` integer id of callback

`handler_name` name of the event handler to map

Returns:

0 on success, -1 on failure

5.74.277 function**diguyScenario::unmap_event_handler_from_callback_id****Prototype:**

```
int unmap_event_handler_from_callback_id(int callback_id,
    const char* handler_name,
    int unmap_all_matches = 0);
```

Description:

This function unmaps the event handler with the given name from a callback id.

Arguments:

callback_id	integer id of callback
handler_name	name of the event handler to map
unmap_all_matches	pass 0 to unmap only the first match, pass 1 to unmap all matches

Returns:

0 on success, -1 on failure

5.74.278 function**diguyScenario::manually_invoke_event_handler****Prototype:**

```
diguyCallbackReturn manually_invoke_event_handler(const char* handler_name);
```

Description:

This function manually invokes the named scenario event handler. The callback_id that will be passed to the event handler will be CALLBACK_ID_MANUALLY_INVOKED.

Note that care should be taken not to end up in an infinite loop of event calls. In general an event handler should not end up directly or indirectly invoking itself.

The event handler is one of the following:

◇ a scenario callback function registered by
[diguyApp::register_scenario_event_handler\(\)](#) or
[diguyApp::register_scenario_event_handler_from_library\(\)](#)

- ◇ a script registered by
[diguyApp::register_scenario_event_handler_script\(\)](#)
- ◇ a Script, Decision, or Library Function in the scenario whose "Event Type" is "Scenario"

Arguments:

handler_name name of the event handler to invoke

Returns:

[DIGUY_CALLBACK_CONTINUE](#) or [DIGUY_CALLBACK_STOP](#)

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.74.279 function [diguyScenario::register_character_event_handler](#)

Prototype:

```
int register_character_event_handler(const char* handler_name,
    diguyCharacterCallback* callback,
    void* callback_user_data = 0);
```

Description:

This function registers a character event handler function that can later be mapped via a call to [diguyCharacter::map_event_handler_to_callback_id\(\)](#).

Call [unregister_character_event_handler\(\)](#) to unregister the function.

Arguments:

handler_name	name of the event handler
callback	pointer to handler function

callback_user_data pointer for user's own use; DI-Guy will do nothing to the contents of this pointer beyond passing it back when the handler function is called

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.74.280 function `diguyScenario::register_character_event_handler_from_library`

Prototype:

```
int register_character_event_handler_from_library(const char* handler_name,
const char* library_name,
const char* function_name,
void* callback_user_data = 0);
```

Description:

This function registers a character event handler function that can later be mapped via a call to `diguyCharacter::map_event_handler_to_callback_id()`.

Unlike `register_character_event_handler()`, which directly passes a function pointer, this function looks up the function pointer from a shared library (a .dll under Windows, a .so under Unix).

The passed `library_name` should not include the .dll or .so extension; these will be added automatically. This allows for cross-platform scenarios that don't have differing dynamic library extensions built into them.

Call `unregister_character_event_handler()` to unregister the function.

Arguments:

<code>handler_name</code>	name of the event handler
<code>library_name</code>	pointer to handler function
<code>function_name</code>	pointer to handler function
<code>callback_user_data</code>	pointer for user's own use; DI-Guy will do nothing to the contents of this pointer beyond passing it back when the handler function is called

Returns:

0 on success, -1 on failure

5.74.281 function `diguyScenario::register_character_event_handler_script`

Prototype:

```
int register_character_event_handler_script(const char* handler_name,
    const char* handler_script,
    const char* handler_script_type = 0);
```

Description:

This function registers a character event handler script that can later be mapped via a call to `diguyCharacter::map_event_handler_to_callback_id()`.

Call `unregister_character_event_handler()` to unregister the script.

Arguments:

<code>handler_name</code>	name of the event handler
<code>handler_script</code>	Script text

`handler_script_type` type of script; pass NULL for scenario to use the default script interpreter

Returns:

0 on success, -1 on failure

5.74.282 function `diguyScenario::has_registered_character_event_handler`

Prototype:

```
int has_registered_character_event_handler(const char* handler_name);
```

Returns:

1 if there is a character event handler with the given name, 0 if not

Arguments:

`handler_name` name of the event handler

5.74.283 function diguyScenario::unregister_character_event_handler

Prototype:

```
int unregister_character_event_handler(const char* handler_name);
```

Description:

This function unregisters a character event handler. This will unmap the event handler from any mappings it is a part of.

Arguments:

`handler_name` name of the event handler

Returns:

0 on success, -1 on failure

5.74.284 function diguyScenario::register_signal_event_handler

Prototype:

```
int register_signal_event_handler(const char* handler_name,
    diguySignalCallback* callback,
    void* callback_user_data = 0);
```

Description:

This function is analogous to the [register_character_event_handler \(\)](#) function, but is for signals rather than characters.

Callable From:

◇ C++

5.74.285 function**diguyScenario::register_signal_event_handler_from_library****Prototype:**

```
int register_signal_event_handler_from_library(const char* handler_name,  
const char* library_name,  
const char* function_name,  
void* callback_user_data = 0);
```

Description:

This function is analogous to the [register_character_event_handler_from_library\(\)](#) function, but is for signals rather than characters.

5.74.286 function**diguyScenario::register_signal_event_handler_script****Prototype:**

```
int register_signal_event_handler_script(const char* handler_name,  
const char* handler_script,  
const char* handler_script_type = 0);
```

Description:

This function is analogous to the [register_character_event_handler_script\(\)](#) function, but is for signals rather than characters.

5.74.287 function**diguyScenario::has_registered_signal_event_handler****Prototype:**

```
int has_registered_signal_event_handler(const char* handler_name);
```

Description:

This function is analogous to the [has_registered_character_event_handler\(\)](#) function, but is for signals rather than characters.

5.74.288 function**diguyScenario::unregister_signal_event_handler****Prototype:**

```
int unregister_signal_event_handler(const char* handler_name);
```

Description:

This function is analogous to the [unregister_character_event_handler\(\)](#) function, but is for signals rather than characters.

5.74.289 function**diguyScenario::register_sensor_region_event_handler****Prototype:**

```
int register_sensor_region_event_handler(const char* handler_name,
    diguySensorRegionCallback* callback,
    void* callback_user_data = 0);
```

Description:

This function is analogous to the [register_character_event_handler\(\)](#) function, but is for sensor regions rather than characters.

Callable From:

◇ C++

5.74.290 function**diguyScenario::register_sensor_region_event_handler_from_library****Prototype:**

```
int register_sensor_region_event_handler_from_library(const char* handler_name,
    const char* library_name,
    const char* function_name,
    void* callback_user_data = 0);
```

Description:

This function is analogous to the [register_character_event_handler_from_library\(\)](#) function, but is for sensor regions rather than characters.

5.74.291 function**diguyScenario::register_sensor_region_event_handler_script****Prototype:**

```
int register_sensor_region_event_handler_script(const char* handler_name,  
const char* handler_script,  
const char* handler_script_type = 0);
```

Description:

This function is analogous to the [register_character_event_handler_script\(\)](#) function, but is for sensor regions rather than characters.

5.74.292 function**diguyScenario::has_registered_sensor_region_event_handler****Prototype:**

```
int has_registered_sensor_region_event_handler(const char* handler_name);
```

Description:

This function is analogous to the [has_registered_character_event_handler\(\)](#) function, but is for sensor regions rather than characters.

5.74.293 function**diguyScenario::unregister_sensor_region_event_handler****Prototype:**

```
int unregister_sensor_region_event_handler(const char* handler_name);
```

Description:

This function is analogous to the [unregister_character_event_handler\(\)](#) function, but is for sensor regions rather than characters.

5.74.294 function**diguyScenario::register_variable_event_handler****Prototype:**

```
int register_variable_event_handler(const char* handler_name,
    diguyVariableCallback* callback,
    void* callback_user_data = 0);
```

Description:

This function is analogous to the [register_character_event_handler\(\)](#) function, but is for variables rather than characters.

Callable From:

◇ C++

5.74.295 function**diguyScenario::register_variable_event_handler_from_library****Prototype:**

```
int register_variable_event_handler_from_library(const char* handler_name,  
const char* library_name,  
const char* function_name,  
void* callback_user_data = 0);
```

Description:

This function is analogous to the [register_character_event_handler_from_library\(\)](#) function, but is for variables rather than characters.

5.74.296 function**diguyScenario::register_variable_event_handler_script****Prototype:**

```
int register_variable_event_handler_script(const char* handler_name,  
const char* handler_script,  
const char* handler_script_type = 0);
```

Description:

This function is analogous to the [register_character_event_handler_script\(\)](#) function, but is for variables rather than characters.

5.74.297 function**diguyScenario::has_registered_variable_event_handler****Prototype:**

```
int has_registered_variable_event_handler(const char* handler_name);
```

Description:

This function is analogous to the [has_registered_character_event_handler\(\)](#) function, but is for variables rather than characters.

5.74.298 function**diguyScenario::unregister_variable_event_handler****Prototype:**

```
int unregister_variable_event_handler(const char* handler_name);
```

Description:

This function is analogous to the [unregister_character_event_handler\(\)](#) function, but is for variables rather than characters.

Review Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.299 function diguyScenario::save_review_data

Prototype:

```
int save_review_data(const char* filename);
```

Description:

This function saves the current run of the scenario to the specified file, and other files whose names are derived from the passed filename. The passed filename should end with the extension ".dsr" (DI-Guy Review).

The review data can be reloaded later by calling the [load\(\)](#) function, and passing the same filename.

The following is saved for later review:

- ◇ all character positions and poses
- ◇ sounds played by characters due to [play_sound\(\)](#) and

[fire_weapon_n_times\(\)](#) calls

Note that Paths, Waypoints, and Event Beads per se are **not** saved, but rather their effects on the characters' positions and poses.

Arguments:

filename file in which to save review data

Returns:

0 on success, -1 on failure

C++ Example:

```
// when shutting down the current run:  
scenario->save_review_data("review0.dsr");  
  
// when after action review is needed:  
scenario->load("review0.dsr");
```

5.74.300 function diguyScenario::get_loaded_from_review_data

Prototype:

```
int get_loaded_from_review_data();
```

Returns:

1 if scenario was loaded from review data (from a .dsr file); 0 if not (from a .dss file)

5.74.301 function diguyScenario::set_history_type**Prototype:**

```
int set_history_type(diguyHistoryType history_type);
```

Description:

This function sets what type of "history" will be kept for the scenario.

If time is run backwards in a scenario with history, the following will be "remembered" and replayed:

- ◇ camera settings currently in the primary view's camera
- ◇ signal trigger counts
- ◇ enabled/disabled status of scene objects
- ◇ sounds

The default is [DIGUY_HISTORY_TYPE_NONE](#) for scenarios created using the DI-Guy API, and [DIGUY_HISTORY_TYPE_COMPLETE](#) for scenarios created using the DI-Guy Scenario editor.

Arguments:

`history_type` history type to be used for the scenario

`history_type` should be one of the following values:

DIGUY_HISTORY_TYPE_NONE

This history type saves no history.

DIGUY_HISTORY_TYPE_COMPLETE

This history type saves a complete history.

[DIGUY_HISTORY_TYPE_LAST](#) and [DIGUY_HISTORY_TYPE_FIRST](#) are not supported for scenario history.

Returns:

0 on success, -1 on failure

5.74.302 function diguyScenario::get_history_type**Prototype:**

```
diguyHistoryType get_history_type();
```

Returns:

the current history type of the scenario; see [set_history_type\(\)](#)

5.74.303 function diguyScenario::set_overall_history_type**Prototype:**

```
void set_overall_history_type(diguyHistoryType overall_history_type);
```

Description:

This function sets the history type for this scenario, as well as the history types of all characters.

See also [diguyScenario::set_history_type\(\)](#) and [diguyCharacter::set_history_type\(\)](#).

Arguments:

`history_type` history type to be used for the scenario and all characters

5.74.304 function diguyScenario::get_overall_history_type**Prototype:**

```
diguyHistoryType get_overall_history_type();
```

Description:

This function returns the overall history type of the scenario.

The return value will be:

- ◇ [DIGUY_HISTORY_TYPE_COMPLETE](#) if history is enabled for the scenario and **all** characters
- ◇ [DIGUY_HISTORY_TYPE_PARTIAL](#) if history is enabled for at least one of the scenario or any character, but not all
- ◇ [DIGUY_HISTORY_TYPE_NONE](#) if history is not enabled for the scenario or any character

See also [diguyScenario::get_history_type\(\)](#) and [diguyCharacter::get_history_type\(\)](#).

5.74.305 function diguyScenario::get_replaying_history**Prototype:**

```
int get_replaying_history();
```

Returns:

whether the scenario is playing back stored history

5.74.306 function**diguyScenario::set_history_max_sound_instances****Prototype:**

```
void set_history_max_sound_instances(int num_sound_instances);
```

Description:

This function sets how many sound instances are kept in memory when history is enabled. By keeping them in memory, users can "scrub" backwards in time and hear sounds that were started earlier in time and will be resumed at the proper offset into the sound. This function lets users balance that need with memory management.

Arguments:

`num_sound_instances` override system default of 100

Returns:

none

LOD Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.307 function**diguyScenario::set_automatic_graphics_lod_switching****Prototype:**

```
void set_automatic_graphics_lod_switching(int enable_graphics_lod_switching);
```

Description:

This function sets whether automatic graphics LOD switching should be enabled for certain graphics environments (see below).

By default it is enabled in DI-Guy Scenario, and disabled in a DI-Guy API application.

There are two ways that DI-Guy calculates the proper graphics LOD. For some graphics environments the calculation is done "manually", using data from a [diguyViewCamera](#). In other environments the calculation is done by the graphics environment itself. The method used in each graphics environment is noted below.

OpenGL Version:

This function enables or disables automatic LOD switching. In this graphics environment LOD calculations are done by the renderer; information from the [diguyViewCamera](#) is not used.

Direct3D Version:

This function enables or disables automatic LOD switching. In this graphics environment LOD calculations are done relative to a [diguyViewCamera](#).

DI-Guy Graphics API Version:

This function enables or disables automatic LOD switching. In this graphics environment LOD calculations are done by the renderer; information from the [diguyViewCamera](#) is not used.

5.74.308 function diguyScenario::get_automatic_graphics_lod_switching

Prototype:

```
int get_automatic_graphics_lod_switching();
```

Returns:

most recent setting of
[set_automatic_graphics_lod_switching\(\)](#)

5.74.309 function diguyScenario::set_default_lod_ranges

Prototype:

```
void set_default_lod_ranges(const char* character_type, float* ranges);
```

Description:

Sets the distances at which level of detail switching occurs for characters of the specified type. Existing characters are not affected; only newly created characters will have the specified LOD switching ranges.

Characters will display with the highest level of detail when viewed from a distance between ranges[0] and ranges[1]. Characters will not display at all when viewed from further away than the highest switching range.

Note: ranges[0] should almost always be 0.0.

Arguments:

`character_type` character type name, as returned by
[get_character_type_at_index\(\)](#)

ranges an array of 8 non-negative floating point numbers, each larger than the one before, specifying the LOD switching ranges in meters

Pass "all" for `character_type` to set the default ranges of all character types.

The table below indicates the number of polygons for one of the default soldier models in each level of detail, and the LOD switching ranges in effect before this function is called.

lod	polys	min	max
1	2300	0	5
2	900	5	10
3	550	10	20
4	280	20	40
5	130	40	70
6	60	70	100
7	40	100	1000

Callable From:

◇ C++

5.74.310 function `diguyScenario::set_scale_graphics_lod_ranges_from_view_settings`

Prototype:

```
void set_scale_graphics_lod_ranges_from_view_settings(int do_scale);
```

Description:

This function sets whether view and camera settings such as field-of-view (FOV) and window size should be taken into account when determining the graphics LOD from a character's LOD ranges.

The camera settings that are used are taken from the current render camera, as set by `set_render_camera()`. This happens automatically in DI-Guy Scenario, but must be set manually in the DI-Guy SDK.

DI-Guy graphics LODs are distance-based, but this approach can be problematic if the current camera's FOV is very narrow; a narrow FOV essentially acts like a telephoto lens. In this case a character that is far away, and would therefore be rendered with few polygons, can actually be quite large in the 3D view.

Setting this flag to 1 will cause the FOV to be taken into account when calculating the effective distance of a character from the camera.

Many DI-Guy LOD distances assume that the vertical FOV of the view is around 40. Smaller FOVs will scale the distances larger so that LOD changes happen further out. Likewise larger FOVs will scale the distances smaller; in this case even nearby characters will appear small and therefore need less resolution.

Arguments:

do_scale – pass 1 to enable scaling, 0 to disable

5.74.311 function diguyScenario::get_scale_graphics_lod_ranges_from_view_settings()

Prototype:

```
int get_scale_graphics_lod_ranges_from_view_settings();
```

Returns:

most recent setting of
`set_scale_graphics_lod_ranges_from_view_settings()`

5.74.312 function**diguyScenario::set_automatic_motion_lod_switching****Prototype:**

```
void set_automatic_motion_lod_switching(int enable_motion_lod_switching);
```

Description:

This function sets whether automatic motion LOD switching should be enabled.

By default it is enabled in DI-Guy Scenario, and disabled in a DI-Guy API application.

When running in a user application that uses the DI-Guy API, the primary view's camera's position and orientation must be kept consistent with the application's perception of current camera settings. A pointer to the primary view's camera can be obtained by calling [get_scenario_camera\(\)](#).

The following describes the algorithm used by automatic motion LOD switching:

- ◇ if a character is in front of the primary view's camera, the motion LOD is set to 1 (animate all joints)
- ◇ else set motion LOD to 5 (stop animating everything but base position)

Note that if there are multiple views open on the scenario, the character must be behind **all** cameras for the motion LOD to be set to 5.

Also, if the history type of the character (as set by [diguyCharacter::set_history_type\(\)](#)) is anything but [DIGUY_HISTORY_TYPE_NONE](#), motion LOD 1 will be used since during scenario playback the character might be in front of any camera.

5.74.313 function**diguyScenario::get_automatic_motion_lod_switching****Prototype:**

```
int get_automatic_motion_lod_switching();
```

Description:

This function returns whether automatic motion LOD switching is be enabled for characters in the scenario, as set by [set_automatic_motion_lod_switching\(\)](#).

Returns:

1 if LODs enabled; 0 if not

Script Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.314 function diguyScenario::eval_script**Prototype:**

```
int eval_script(const char* script_text,
               int save_as_transient = 0,
               const char* script_type = NULL,
               const char* script_source = NULL);
```

Description:

This function sends the passed string through the script interpreter.

Arguments:

<code>script_text</code>	string to be evaluated
<code>save_as_transient</code>	flag stating whether <code>script_text</code> should be remembered as a transient event at the current scenario <code>t</code> ; see <code>push_transient_script_event()</code>
<code>script_type</code>	type of script contained in <code>script_text</code> ; pass <code>NULL</code> to use scenario's default interpreter
<code>script_source</code>	string identifying the 'source' of the script; will be printed in error output if there are syntax errors in the script

Returns:

0 on success, -1 on failure

C++ Example:

```
scenario->eval_script("signal1:trigger();",
0,
"lua",
"User Script");
```

5.74.315 function diguyScenario::eval_script_file**Prototype:**

```
int eval_script_file(const char* filename,  
                    const char* script_type = NULL);
```

Description:

This function sends the contents of the specified file through the script interpreter.

Arguments:

filename	filename of script to be evaluated
script_type	type of script contained in file; pass NULL to use the scenario's default interpreter

Returns:

0 on success, -1 on failure

C++ Example:

```
scenario->eval_script_file("my_scripts.pl");
```

5.74.316 function diguyScenario::trigger_script**Prototype:**

```
int trigger_script(const char* script_name);
```

Description:

This function manually triggers a script.

Arguments:

`script_name` name of the script to be triggered

Returns:

return result of script; 0 if script not found

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.74.317 function diguyScenario::trigger_script_delayed**Prototype:**

```
void trigger_script_delayed(const char* script_name, float t_delay);
```

Description:

This function is similar to `trigger_script()`, but delays the trigger by the specified time.

Arguments:

`script_name` name of the script to be triggered
`t_delay` how many seconds to delay trigger

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.74.318 function diguyScenario::trigger_decision**Prototype:**

```
int trigger_decision(const char* decision_name);
```

Description:

This function manually triggers a decision.

Arguments:

`decision_name` name of the decision to be triggered

Returns:

return result of decision; 0 if decision not found

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.74.319 function diguyScenario::trigger_decision_delayed**Prototype:**

```
void trigger_decision_delayed(const char* decision_name, float t_delay);
```

Description:

This function is similar to [trigger_decision\(\)](#), but delays the trigger by the specified time.

Arguments:

`decision_name` name of the decision to be triggered
`t_delay` how many seconds to delay trigger

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision

5.74.320 function diguyScenario::push_transient_script_event**Prototype:**

```
void push_transient_script_event(float t, const char* script_text, const char* script_type)
```

Description:

This function places the passed script text on a list of scripts that should be re-evaluated when the scenario is replayed from review data.

Note that the script is **not** evaluated at this time. This can be done by calling [eval_script\(\)](#) and passing 1 for the save_as_transient argument.

If the scenario is reset (as in a call to [reset\(\)](#)), all transient scripts will be deleted. The transient script events can be saved and restored by calling [save_transient_script_events\(\)](#) and [load_transient_script_events\(\)](#), respectively.

Arguments:

t	time at which script should be re-evaluated during scenario playback
script_text	Script to be evaluated
script_type	Scripting language to use (lua)

A copy of the passed script text is made.

Returns:

0 on success, -1 on failure

5.74.321 function diguyScenario::clear_transient_script_events**Prototype:**

```
void clear_transient_script_events();
```

Description:

This function deletes all transient script events.

5.74.322 function diguyScenario::save_transient_script_events**Prototype:**

```
int save_transient_script_events(const char* filename);
```

Description:

This function saves the transient script events to the specified file. They can be restored by calling `load_transient_script_events( )` with the same filename.

Arguments:

`filename` file in which transient script events should be saved

Returns:

0 on success, -1 on failure

5.74.323 function diguyScenario::load_transient_script_events**Prototype:**

```
int load_transient_script_events(const char* filename);
```

Description:

This function loads transient script events saved by a call to [save_transient_script_events\(\)](#).

Arguments:

filename file from which transient script events should be loaded

Returns:

0 on success, -1 on failure

5.74.324 function diguyScenario::load_package**Prototype:**

```
int load_package(const char* filename,  
                const char* language = NULL,  
                int warn_if_not_found = 1);
```

Description:

This function creates a package object that points to an external file. If the language is not specified it will be determined based on the filename's extension. DI-Guy will attempt to derive an appropriate name for the package. In DI-Guy Scenario package dependencies are also extracted and loaded; currently this must be done manually in the SDK.

5.74.325 function**diguyScenario::load_package_with_dependency****Prototype:**

```
int load_package_with_dependency(const char* filename,  
    const char* depends_on_package,  
    const char* language = NULL,  
    int warn_if_not_found = 1);
```

Description:

Documentation Pending

Meta-Action Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.326 function

diguyScenario::get_character_type_num_postures

Prototype:

```
int get_character_type_num_postures(const char* character_type);
```

Description:

This function returns the number of postures the specified character type's actions support.

Use this function along with [get_character_type_posture_at_index\(\)](#) to enumerate the character's postures.

Arguments:

character_type character type name, as returned by [get_character_type_at_index\(\)](#), for example

Returns:

the number of postures the specified character type's actions support

5.74.327 function

diguyScenario::get_character_type_posture_at_index

Prototype:

```
diguyMotionPosture get_character_type_posture_at_index(const char* character_type,
int index);
```

Description:

This function returns the posture at the specified index.

Use this function along with [get_character_type_num_postures\(\)](#) to enumerate the character's postures.

Arguments:

character_type character type name, as returned by [get_character_type_at_index\(\)](#), for example

index index of the posture; indices start at 0

Returns:

the posture at the specified index

5.74.328 function

diguyScenario::get_character_type_action_posture

Prototype:

```
diguyMotionPosture get_character_type_action_posture(const char* character_type,
                                                    const char* action_name);
```

Description:

This function returns the overall posture of the specified action.

The results of this function call can be useful in calls to the [diguyCharacter::get_action_from_description\(\)](#) function, if for example an action with the same posture but a faster speed is desired.

Arguments:

<code>character_type</code>	character type name, as returned by <code>get_character_type_at_index()</code> , for example
<code>action_name</code>	the action to query

Returns:

the overall posture of the specified action

5.74.329 function diguyScenario::get_character_type_num_variants

Prototype:

```
int get_character_type_num_variants(const char* character_type);
```

Description:

This function returns the number of variants the specified character type's actions support.

Use this function along with

[get_character_type_variant_at_index\(\)](#) to enumerate the character's variants.

This function differs from

[get_character_type_action_num_variants\(\)](#) in that this function returns the number of variants of **all** actions of the character. The other function returns the number of variants of a single action.

Arguments:

`character_type` character type name, as returned by
`get_character_type_at_index()`, for example

Returns:

the number of variants the specified character type's actions support

5.74.330 function `diguyScenario::get_character_type_variant_at_index`

Prototype:

```
diguyMotionVariant get_character_type_variant_at_index(const char* character_type,  
int index);
```

Description:

This function returns the variant at the specified index.

Use this function along with
[get_character_type_num_variants\(\)](#) to enumerate the
character's variants.

This function differs from
[get_character_type_action_variant_at_index\(\)](#) in that this
function returns one of the variants of **all** actions of the character. The other
function returns one of the variants of a single action.

Arguments:

character_type character type name, as returned by
get_character_type_at_index(), for example

index index of the posture; indices start at 0

Returns:

the variant at the specified index

5.74.331 function diguyScenario::get_character_type_action_primary_variant

Prototype:

```
diguyMotionVariant get_character_type_action_primary_variant(const char* character_t
const char* action_name);
```

Description:

This function returns the primary variant of the specified action.

Arguments:

character_type character type name, as returned by
get_character_type_at_index(), for example

action_name the action to query

Returns:

the overall variant of the specified action

5.74.332 function

diguyScenario::get_character_type_action_num_variants

Prototype:

```
int get_character_type_action_num_variants(const char* character_type,
    const char* action_name);
```

Description:

This function returns the number of variants the specified action of the specified character type has.

Use this function along with

[get_character_type_action_variant_at_index\(\)](#) to enumerate the character's variants.

This function differs from [get_character_type_num_variants\(\)](#) in that this function returns the number of variants of a single action of the character. The other function returns the number of variants of **all** actions.

Arguments:

<code>character_type</code>	character type name, as returned by <code>get_character_type_at_index()</code> , for example
<code>action_name</code>	the action to query

Returns:

the number of variants of the specified action of the specified character type

5.74.333 function**diguyScenario::get_character_type_action_variant_at_index****Prototype:**

```
diguyMotionVariant get_character_type_action_variant_at_index(const char* character_
    const char* action_name,
    int index);
```

Description:

This function returns the variant at the specified index.

Use this function along with

[get_character_type_action_num_variants\(\)](#) to enumerate the character's variants.

This function differs from

[get_character_type_variant_at_index\(\)](#) in that this function returns one of the variants of a single action of the character. The other function returns one of the variants of **all** actions.

Arguments:

<code>character_type</code>	character type name, as returned by <code>get_character_type_at_index()</code> , for example
<code>index</code>	index of the posture; indices start at 0

Returns:

the variant at the specified index

5.74.334 function `diguyScenario::get_character_type_num_directions`

Prototype:

```
int get_character_type_num_directions(const char* character_type);
```

Description:

This function returns the number of directions the specified character type's actions can move.

Use the `get_character_type_direction_at_index()` function to enumerate the directions available to a character.

The results of this function call can be useful in calls to the `diguyCharacter::get_action_from_description()` function.

Arguments:

`character_type` character type name, as returned by `get_character_type_at_index()`, for example

Returns:

the number of directions the specified character type's actions can move

5.74.335 function

diguyScenario::get_character_type_direction_at_index

Prototype:

```
diguyMotionDirection get_character_type_direction_at_index(const char* character_type,
    int index);
```

Description:

Use the [get_character_type_num_directions\(\)](#) function to get the number of directions available to a character.

The results of this function call can be useful in calls to the [diguyCharacter::get_action_from_description\(\)](#) function.

Arguments:

<code>character_type</code>	character type name, as returned by <code>get_character_type_at_index()</code> , for example
<code>index</code>	index of the direction; indices start at 0

Returns:

the direction at the specified index

Interaction Machine Functions

Interaction machines are designed to provide "conversation tree" type interactions with the scenario user. Interactions occur in the form of information text and multiple choice responses shown to the user.

The interaction can be as simple as showing a quick notification to the user and waiting for them to acknowledge it, or as complex as having a full-fledged conversation with another character in the scenario.

The name "interaction machine" comes from the fact that the interactions are based on a straight-forward state machine approach.

Interaction machines are for use in DI-Guy Scenario only, not with the DI-Guy API.

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.336 function diguyScenario::get_num_interaction_machines

Prototype:

```
int get_num_interaction_machines();
```

Returns:

number of interaction machines in the scenario

5.74.337 function diguyScenario::create_interaction_machine**Prototype:**

```
diguyInteractionMachine* create_interaction_machine(const char* name);
```

Description:

This function creates a new interaction machine and returns a pointer to it.

Note that there is a shared interaction machine that always exists; a pointer to it can be obtained by calling [get_shared_interaction_machine\(\)](#). See that function for more details.

Arguments:

name	name of the new interaction machine

Returns:

pointer of type [diguyInteractionMachine](#); NULL if creation failed

5.74.338 function diguyScenario::destroy_interaction_machine**Prototype:**

```
void destroy_interaction_machine(diguyInteractionMachine* interaction_machine);
```

Description:

This function destroys the passed interaction machine.

NOTE: This function should not be called on the shared interaction machine returned by [get_shared_interaction_machine\(\)](#).

Arguments:

interaction_machine	pointer to a diguyInteractionMachine to be destroyed

5.74.339 function diguyScenario::get_interaction_machine_at_index

Prototype:

```
diguyInteractionMachine* get_interaction_machine_at_index(int index);
```

Returns:

pointer of type [diguyInteractionMachine](#); NULL if no interaction machine at the specified index

Arguments:

index	index of the interaction machine; indices start at 0

5.74.340 function diguyScenario::find_interaction_machine

Prototype:

```
diguyInteractionMachine* find_interaction_machine(const char* name);
```

Description:

This function returns a pointer to the specified interaction machine.

Arguments:

name	name of interaction machine to be found

Returns:

pointer of type [diguyInteractionMachine](#); NULL if not found

5.74.341 function diguyScenario::get_active_interaction_machine

Prototype:

```
diguyInteractionMachine* get_active_interaction_machine();
```

Description:

This function returns a pointer to the "active" interaction machine. Only one interaction machine can be showing and accepting input at a time.

Returns:

pointer of type [diguyInteractionMachine](#); NULL if no interaction machine is active.

5.74.342 function diguyScenario::get_shared_interaction_machine

Prototype:

```
diguyInteractionMachine* get_shared_interaction_machine();
```

Description:

This function returns a pointer to the shared interaction machine that is always available.

This shared interaction machine is convenient for quick alerts, notifications, or questions. Because it is shared, however, most or all of its parameters need to be set each time it is to be shown.

Do not call [destroy_interaction_machine\(\)](#) on the returned object. It is owned by the scenario.

Returns:

pointer of type [diguyInteractionMachine](#); should never be NULL

5.74.343 function**diguyScenario::show_notification_with_shared_interaction_mach****Prototype:**

```
int show_notification_with_shared_interaction_machine(const char* heading,
    const char* info,
    int pause_scenario = 0,
    const char* input_text = NULL,
    int ui_appearance = diguyInteractionMachine::UI_APPEARANCE_NOTIFICATION);
```

Description:

This is a convenience function for using the shared interaction machine to show a notification to the user. It sets parameters of the shared interaction machine based on the passed info and shows/activates the machine.

Arguments:

heading	text that will be at the top of the dialog
info	informational text
pause_scenario	pass 1 to pause the scenario while the interaction machine is active, 0 to leave the scenario running if it was when this function was called
input_text	text that user will have to click to dismiss the interaction machine; something like "Ok" is typical
ui_appearance	the color theme of the interaction machine

See the [diguyInteractionMachine](#) documentation for more information on what the various parameters mean.

Returns:

0 on success, -1 on failure

Lua Example:

```
this_scenario:show_notification_with_shared_interaction_machine(
    "WARNING!",
    "Moving any closer to the harmful gas is not advised.",
    0,
    "Ok",
    diguyInteractionMachine_UI_APPEARANCE_NOTIFICATION);
```


I-Guy Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.344 function diguyScenario::get_iguy_controller

Prototype:

```
diguyIGuyController* get_iguy_controller();
```

Description:

This function returns a pointer to the I-Guy controller object of the scenario.

Returns:

pointer of type [diguyIGuyController](#); should never be NULL

Label Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.345 function diguyScenario::find_or_create_label

Prototype:

```
diguyViewLabel* find_or_create_label(const char* name);
```

Description:

This function finds the [diguyViewLabel](#) with the given name or creates it if it doesn't exist.

Arguments:

name	name of the diguyViewLabel to find or create

Returns:

pointer of type [diguyViewLabel](#); should never be NULL

5.74.346 function diguyScenario::find_label**Prototype:**

```
diguyViewLabel* find_label(const char* name);
```

Description:

This function returns a pointer to the specified `diguyViewLabel`.

Arguments:

name	name of <code>diguyViewLabel</code> to be found

Returns:

pointer of type `diguyViewLabel`; NULL if not found

5.74.347 function diguyScenario::destroy_label**Prototype:**

```
int destroy_label(diguyViewLabel* label);
```

Description:

This function destroys the passed in `diguyViewLabel`.

Arguments:

label	pointer to a <code>diguyViewLabel</code>

Returns:

0 on success, -1 on failure

5.74.348 function diguyScenario::get_num_labels**Prototype:**

```
int    get_num_labels();
```

Returns:

The number of [diguyViewLabel](#) objects in the scenario.

5.74.349 function diguyScenario::get_label_at_index**Prototype:**

```
diguyViewLabel* get_label_at_index(int index);
```

Returns:

The [diguyViewLabel](#) at the given index.

Arguments:

`index` index of the [diguyViewLabel](#); indices start at 0

5.74.350 function diguyScenario::reset_labels**Prototype:**

```
void reset_labels();
```

Description:

This function removes all diguyViewLabels in the scenario.

5.74.351 function diguyScenario::send_label_to_front**Prototype:**

```
void send_label_to_front(diguyViewLabel* label);
```

Description:

This function moves the specified [diguyViewLabel](#) to the front of the drawing list. This is sometimes necessary if labels overlap.

Arguments:

`label` pointer to [diguyViewLabel](#) to move

5.74.352 function diguyScenario::send_label_to_back

Prototype:

```
void send_label_to_back(diguyViewLabel* label);
```

Description:

This function moves the specified `diguyViewLabel` to the back of the drawing list. This is sometimes necessary if labels overlap.

Arguments:

<code>label</code>	pointer to <code>diguyViewLabel</code> to move

Button Panel Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.353 function diguyScenario::find_or_create_panel**Prototype:**

```
diguyViewButtonPanel* find_or_create_panel(const char* name,  
    int horizontal = 0,  
    int title = 1);
```

Description:

This function finds the [diguyViewButtonPanel](#) with the given name or creates it if it doesn't exist.

Arguments:

name	name of the diguyViewButtonPanel to find or create

Returns:

pointer of type [diguyViewButtonPanel](#); should never be NULL

5.74.354 function diguyScenario::find_panel**Prototype:**

```
diguyViewButtonPanel* find_panel(const char* name);
```

Description:

This function returns a pointer to the specified [diguyViewButtonPanel](#).

Arguments:

name	name of diguyViewButtonPanel to be found

Returns:

pointer of type [diguyViewButtonPanel](#); NULL if not found

5.74.355 function diguyScenario::destroy_panel**Prototype:**

```
int destroy_panel(diguyViewButtonPanel* panel);
```

Description:

This function destroys the passed in `diguyViewButtonPanel`.

Arguments:

`panel` pointer to a `diguyViewButtonPanel`

Returns:

0 on success, -1 on failure

5.74.356 function diguyScenario::get_num_panels**Prototype:**

```
int get_num_panels();
```

Returns:

The number of `diguyViewButtonPanel` objects in the scenario.

5.74.357 function diguyScenario::get_panel_at_index**Prototype:**

```
diguyViewButtonPanel* get_panel_at_index(int index);
```

Returns:

The [diguyViewButtonPanel](#) at the given index.

Arguments:

`index` index of the [diguyViewButtonPanel](#); indices start at 0

Misc. DI-Guy AI Functions

Note that these functions require a DI-Guy AI license, and that DI-Guy AI has been initialized by a call to [diguy_ai_initialize\(\)](#).

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.358 function diguyScenario::get_num_minds**Prototype:**

```
int get_num_minds();
```

Description:

This function returns the number of AI minds that are available to AI agent characters.

Returns:

number of available minds

5.74.359 function diguyScenario::get_mind_name_at_index**Prototype:**

```
const char* get_mind_name_at_index(int index);
```

Returns:

name of the AI mind at the specified index

5.74.360 function diguyScenario::is_valid_character_pointer**Prototype:**

```
int is_valid_character_pointer(diguyCharacter* character);
```

Description:

This function tests if a [diguyCharacter](#) pointer is a valid pointer. This is done by comparing the passed address to those of all characters in the scenario. The function is not exceedingly fast but can be useful when building AI that functions in a networked environment.

Returns:

0 if invalid, 1 if valid

5.74.361 function diguyScenario::is_valid_character_group_pointer**Prototype:**

```
int is_valid_character_group_pointer(diguyCharacterGroup* character_group);
```

Description:

This function tests if a [diguyCharacterGroup](#) pointer is a valid pointer. This is done by comparing the passed address to those of all groups in the scenario. The function is not exceedingly fast but can be useful when building AI that functions in a networked environment.

Returns:

0 if invalid, 1 if valid

Crowd Functions

These functions control accessing `diguyCrowd` objects. Crowds are used by DI-Guy AI.

Note that these functions require a DI-Guy AI license, and that DI-Guy AI has been initialized by a call to `diguy_ai_initialize()`.

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.362 function `diguyScenario::create_crowd`

Prototype:

```
diguyCrowd* create_crowd(const char* name,
                        int make_companion_of_all = 1);
```

Description:

This function creates a new crowd and returns a pointer to it.

A Lua object is also created that has a pointer to this crowd. This object can be retrieved by calling `find_lua_crowd(crowd_name)` or `find_lua_crowd(diguyCrowd* pointer)` in lua.

Arguments:

<code>name</code>	name of crowd to be created
<code>make_companion_of_all</code>	pass 1 to make this crowd a companion of all existing crowds

Returns:

pointer of type [diguyCrowd](#)

5.74.363 function diguyScenario::destroy_crowd

Prototype:

```
int destroy_crowd(diguyCrowd* crowd,
    int also_destroy_crowd_members = 0,
    int also_destroy_path_shapes = 0,
    int delay_destruction = 0);
```

Description:

This function destroys a crowd. It can also optionally destroy the characters that are crowd members, and the path shape that is associated with the crowd.

Arguments:

<code>crowd</code>	pointer to a diguyCrowd
<code>also_destroy_crowd_members</code>	pass 1 to also destroy crowd member characters
<code>also_destroy_path_shapes</code>	pass 1 to also destroy crowd-associated path shapes
<code>delay_destruction</code>	

pass 1 to wait till the next update call
to destroy the crowd; can avoid
crashes when AI's receive destruction
messages

Returns:

0 on success, -1 on failure

5.74.364 function diguyScenario::get_num_crowds

Prototype:

```
int get_num_crowds();
```

Returns:

the number of crowds in the scenario

5.74.365 function diguyScenario::get_crowd_at_index**Prototype:**

```
diguyCrowd* get_crowd_at_index(int index);
```

Returns:

pointer of type **diguyCrowd**; NULL if no crowd at the specified index

Arguments:

index index of the crowd; indices start at 0

5.74.366 function diguyScenario::get_unique_crowd_name**Prototype:**

```
const char* get_unique_crowd_name(const char* base_name);
```

Returns:

A unique name based on **base_name**, e.g. "my_crowd7" from "my_crowd".

Arguments:

base_name base string from which to build a unique name

5.74.367 function diguyScenario::find_crowd**Prototype:**

```
diguyCrowd* find_crowd(const char* crowd_name);
```

Description:

This function returns a pointer to the specified crowd.

Arguments:

name	name of crowd to be found

Returns:

pointer of type [diguyCrowd](#); NULL if not found

5.74.368 function diguyScenario::find_or_create_crowd**Prototype:**

```
diguyCrowd* find_or_create_crowd(const char* crowd_name,  
int make_companion_of_all = 1);
```

Description:

This function returns a pointer to the crowd with the given name. A new crowd with the given name will be created if it doesn't already exist.

Arguments:

name	name of the crowd to find or create
make_companion_of_all	pass 1 to make this crowd a companion of all existing crowds

Returns:

pointer of type [diguyCrowd](#); should never be NULL

5.74.369 function diguyScenario::set_create_network_crowds**Prototype:**

```
void set_create_network_crowds(int do_create);
```

Description:

This function sets whether crowds should be created for incoming network entities.

Arguments:

`do_create` pass 1 to create network crowds; 0 to not

5.74.370 function diguyScenario::get_create_network_crowds**Prototype:**

```
int get_create_network_crowds();
```

1 if network crowds will be created; 0 if not

Crowd Profile Functions

These functions control accessing [diguyCrowdProfile](#) objects. Crowd profiles are used by DI-Guy AI.

Note that these functions require a DI-Guy AI license.

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.371 function `diguyScenario::create_crowd_profile`

Prototype:

```
diguyCrowdProfile* create_crowd_profile(const char* name);
```

Description:

This function creates a new crowd profile and returns a pointer to it.

Arguments:

name	name of crowd profile to be created

Returns:

pointer of type [diguyCrowdProfile](#)

5.74.372 function diguyScenario::destroy_crowd_profile**Prototype:**

```
int destroy_crowd_profile(diguyCrowdProfile* crowd_profile);
```

Description:

This function destroys a crowd profile.

Arguments:

`crowd_profile` pointer to a [diguyCrowdProfile](#)

Returns:

0 on success, -1 on failure

5.74.373 function diguyScenario::get_num_crowd_profiles**Prototype:**

```
int get_num_crowd_profiles();
```

Returns:

The number of crowd profiles in the scenario.

5.74.374 function diguyScenario::get_crowd_profile_at_index**Prototype:**

```
diguyCrowdProfile* get_crowd_profile_at_index(int index);
```

Returns:

pointer of type [diguyCrowdProfile](#); NULL if no crowd profile at the specified index

Arguments:

index	index of the crowd profile; indices start at 0

5.74.375 function diguyScenario::find_crowd_profile**Prototype:**

```
diguyCrowdProfile* find_crowd_profile(const char* name);
```

Description:

This function returns a pointer to the specified crowd profile.

Arguments:

name	name of crowd profile to be found

Returns:

pointer of type [diguyCrowdProfile](#); NULL if not found

Load Manager Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.376 function diguyScenario::get_character_load_manager

Prototype:

```
diguyLoadManager* get_character_load_manager();
```

Returns:

pointer of type [diguyLoadManager](#); this scenario's character load manager

Octtree Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.377 function diguyScenario::force_octtree_generation

Prototype:

```
int force_octtree_generation();
```

Description:

This function forces the octtree to rebuild if it's out of date.

5.74.378 function diguyScenario::add_character_to_octtree

Prototype:

```
void add_character_to_octtree(diguyCharacter* character,  
    int use_bounding_box_only = 0);
```

Description:

Documentation Pending

5.74.379 function diguyScenario::remove_character_from_octtree

Prototype:

```
void remove_character_from_octtree(diguyCharacter* character);
```

Description:

Documentation Pending

5.74.380 function diguyScenario::preload_octtree**Prototype:**

```
int preload_octtree(const char* character_type, const char* appearance);
```

Description:

Documentation Pending

5.74.381 function diguyScenario::check_visibility**Prototype:**

```
int check_visibility(float x1, float y1, float z1,  
                    float x2, float y2, float z2);
```

Description:

This function uses the octtree to check if the path between x1, y1, z1 and x2, y2, z2 has any static obstructions.

Returns:

1 if the path is clear, 0 if static obstructions are present.

Altitude Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.382 function `diguyScenario::set_default_character_altitude_function`

Prototype:

```
int set_default_character_altitude_function(diguyAltitudeFunction* altitude_function)
```

Description:

This function sets a default altitude function that will be added to all new characters. It can be overridden on a per-character basis by a call to `diguyCharacter::set_altitude_function()`. See that function for details.

Callable From:

- ◇ C++

5.74.383 function diguyScenario::set_altitude_function**Prototype:**

```
void set_altitude_function(diguyScenarioAltitudeFunction* altitude_function);
```

Description:

This function sets a generic altitude function for the scenario that is used for local path clamping and other operations that require the altitude at specific x, y coordinates to be calculated.

The [ground_clamp\(\)](#) function can also optionally use the registered function.

DI-Guy Scenario has a default altitude function; replacing the default function is not recommended.

Callable From:

◇ C++

5.74.384 function diguyScenario::get_altitude_function**Prototype:**

```
diguyScenarioAltitudeFunction* get_altitude_function();
```

Returns:

the altitude function for the scenario as set by
[set_altitude_function\(\)](#)

Callable From:

◇ C++

5.74.385 function diguyScenario::set_altitude_max_climb**Prototype:**

```
void set_altitude_max_climb(float altitude_max_climb);
```

5.74.386 function diguyScenario::get_altitude_max_climb**Prototype:**

```
float get_altitude_max_climb();
```

5.74.387 function diguyScenario::set_altitude_max_drop**Prototype:**

```
void set_altitude_max_drop(float altitude_max_drop);
```

5.74.388 function diguyScenario::get_altitude_max_drop**Prototype:**

```
float get_altitude_max_drop();
```

Impact and Hit Detection Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.389 function `diguyScenario::set_default_intersection_function`

Prototype:

```
static void set_default_intersection_function(diguyScenarioIntersectionFunction* int
```

Description:

Sets the default intersection function. This function will be used to determine whether there is an intersection between a specified ray and the world.

5.74.390 function diguyScenario::set_intersection_function**Prototype:**

```
void set_intersection_function(diguyScenarioIntersectionFunction* intersection_function)
```

5.74.391 function diguyScenario::trigger_detonation**Prototype:**

```
int trigger_detonation(const char* munition_name,
    float x, float y, float z,
    const char* attacker_name = NULL,
    float radius_override = -1.0f,
    int broadcast_on_network = 1);
```

Description:

Creates a detonation using the specified munition name. If the DI-Guy networking module is enabled the detonation is broadcast over the network.

Arguments:

<code>munition_name</code>	name of the munition to detonate; this will potentially trigger special effects depending on the munition
<code>x, y, z</code>	location of the detonation
<code>attacker_name</code>	this MUST be specified if the detonation is

	being broadcast over the network
<code>radius_override</code>	defaults to the data in the munition config file
<code>broadcast_on_network</code>	pass 0 to not broadcast detonation on DIS network

Note: Overriding the detonation radius will NOT work on broadcast detonations since the override value is not transmitted.

5.74.392 function `diguyScenario::find_character_at_screen_coords`

Prototype:

```
diguyImpact* find_character_at_screen_coords(diguyView* view,
      float sx,
      float sy);
```

Description:

This function checks for intersection against characters in the specified view.

To check for intersections against both characters and scene objects, call [get_intersection_at_screen_coords\(\)](#).

The returned [diguyImpact](#) pointer is owned by the scenario and should not be deleted. It will remain valid until the next call to any one of the following functions:

- ◇ [find_character_at_screen_coords\(\)](#)
- ◇ [get_impact_at_screen_coords\(\)](#)
- ◇ [get_intersection_at_screen_coords\(\)](#)

This function should only be called from a DI-Guy Scenario Plugin.

Arguments:

<code>view</code>	view to check
<code>sx, sy</code>	view coordinates to check

Returns:

pointer of type `diguyImpact`; NULL if no character intersected

Callable From:

◇ DI-Guy Scenario Plugin

5.74.393 function `diguyScenario::get_impact_at_screen_coords`

Prototype:

```
diguyImpact* get_impact_at_screen_coords(diguyView* view,
    float sx,
    float sy,
    int use_octtree = 0);
```

Description:

This function creates an impact object based on the `sx` and `sy` in the specified view. This function will trigger an impact callback on individuals who are touched, (it will not however kill characters who don't have impact callbacks.

The returned `diguyImpact` pointer is owned by the scenario and should not be deleted. It will remain valid until the next call to any one of the following functions:

```
◇ find_character_at_screen_coords()
◇ get_impact_at_screen_coords()
◇ get_intersection_at_screen_coords()
```


This function should only be called from a DI-Guy Scenario Plugin.

Arguments:

<code>view</code>	view to check
<code>sx, sy</code>	view coordinates to check
<code>use_octtree</code>	use the octtree; will not test against all characters unless they are explicitly added, but much faster

Returns:

pointer of type `diguyImpact`;

Callable From:

◇ DI-Guy Scenario Plugin

5.74.394 function `diguyScenario::get_intersection_at_screen_coords`

Prototype:

```
diguyImpact* get_intersection_at_screen_coords(diguyView* view,
float sx,
float sy,
int use_octtree = 0);
```

Description:

This function checks for intersection against the world **and** characters in the specified view. Unlike the function `get_impact_at_screen_coords()`, this function doesn't trigger callbacks; instead, it just fills out the impact info.

The returned `diguyImpact` pointer is owned by the scenario and should not be deleted. It will remain valid until the next call to any one of the following functions:

- ◇ `find_character_at_screen_coords()`
- ◇ `get_impact_at_screen_coords()`

◇ `get_intersection_at_screen_coords()`

This function should only be called from a DI-Guy Scenario Plugin.

Arguments:

<code>view</code>	view to check
<code>sx, sy</code>	view coordinates to check
<code>use_octtree</code>	use the octtree; will not test against all characters unless they are explicitly added, but much faster

Returns:

`diguyImpact` with the xyz location of the intersection; NULL if there was nothing to intersect there

Callable From:

◇ DI-Guy Scenario Plugin

5.74.395 function `diguyScenario::intersect_static_geometry`

Prototype:

```
int intersect_static_geometry(
    float origin_x, float origin_y, float origin_z,
    float dir_x,    float dir_y,    float dir_z,
    float max_distance_to_test,
    float* intersection_x, float* intersection_y, float* intersection_z,
    float* normal_x = NULL, float* normal_y = NULL, float* normal_z = NULL,
    int use_callback = 1);
```

Description:

This function checks for intersection against the world and returns basic collision results.

This function will only return valid results in environments where the DI-Guy geometry octtree is available; currently that is when using OpenGL and the DI-Guy flight geometry loader. DI-Guy Scenario is such a case.

Arguments:

`origin_x, origin_y, origin_z` starting location of the ray trace
`dir_x, dir_y, dir_z` direction of ray trace
`max_distance_to_test` max distance ray will travel; -1.0 will test an infinitely long ray
`intersection_x, _y, and _z` world location of intersection
`normal_x, normal_y, normal_z` normal of intersection surface

Smaller values for `max_distance_to_test` will result in better performance.

The location of the intersection point is returned in the intersection x, y, and z pointers.

The normal of the surface intersected is returned in the normal x, y, and z pointers.

Returns:

1 if intersection occurred, 0 if not.

5.74.396 function `diguyScenario::ground_clamp`

Prototype:

```
float ground_clamp(float x, float y, float z,
                  int* valid = NULL,
                  int use_altitude_function = 1,
                  int use_octtree = 0);
```

Description:

This function returns the z height, or altitude, under the passed x, y, z point.

Two methods can be used to determine the altitude: using the scenario altitude function or using the internal octtree calculated from terrain geometry.

See [set_altitude_function\(\)](#) for information on the scenario altitude function. In DI-Guy Scenario it defaults to using the octtree when it's available, and falls back on using a screen render and Z-buffer check when it's not.

Note that the octtree is not always available. In general it is available only in

DI-Guy Scenario, or when the built-in OpenGL renderer and DI-Guy flight geometry loader are in use.

Arguments:

<code>x, y, z</code>	world location to ground clamp
<code>valid</code>	optional pointer to get an explicit result if anything was hit
<code>use_altitude_function</code>	pass 1 to use scenario altitude function
<code>use_octtree</code>	pass 1 to use internal octtree

Returns:

`new_z` if ground clamp was possible, original `z` if not

Impact Callback Functions:

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.397 function `diguyScenario::set_fire_weapon_intersection_function`

Prototype:

```
int set_fire_weapon_intersection_function(diguyScenarioFindImpactOnLineFunction* imp
```

Description:

The following prototype should be used for the impact function:

```
int impact_func(diguyImpact* impact,
float from_x,
float from_y,
float from_z,
float to_x,
float to_y,
float to_z,
diguyScenario* s);
```

When a character fires their weapon, DI-Guy calls this function with a pointer to the [diguyImpact](#) that must be filled out, the x, y, z position of the shot, and the target x, y, z.

The impact function should then do an intersection test and return 1 if a hit occurred.

By default the impact should have both the attacker and the munition type already specified. The intersection function must at least call `diguyImpact::set_valid_impact( )` for the system to consider the impact valid. See `diguyImpact` for a sample function.

Callable From:

◇ C++

5.74.398 function `diguyScenario::set_default_fire_weapon_intersection_function`

Prototype:

```
static void set_default_fire_weapon_intersection_function(diguyScenarioFindImpactOnl
```

Description:

This function sets an impact function that will be added to all new scenarios. It can be overridden by an explicit call to `diguyScenario::set_fire_weapon_intersection_function( )`. See that function for details.

Callable From:

◇ C++

5.74.399 function diguyScenario::map_color_to_impact**Prototype:**

```
diguyImpact* map_color_to_impact(char r, char g, char b);
```

Description:

Most DI-Guy visual objects (e.g., characters, path shapes, waypoints) have a unique identified, or UID. In many cases these UID values can be coded into 24-bit color values. This can be useful for implementing some types of intersection detection functions, in which each visual object is rendered with a different color.

This function will take the passed color values and update internal DI-Guy state of which objects have been hit.

It will also return an impact record containing information a subset of that information.

This can include:

- ◇ the hit character, if one was hit
- ◇ the hit link and shape of the character
- ◇ the hit scene object, if one was hit

Note that the returned impact pointer is owned by DI-Guy and is temporary. The information you need from it should be immediately read, and the pointer should not be stored.

5.74.400 function diguyScenario::get_last_env_impact_info**Prototype:**

```
diguyImpact* get_last_env_impact_info();
```

Description:

When the callback with id `diguyScenario::CALLBACK_ID_SCENE_OBJECT_IMPACT` is called, this function will return the impact object containing information about that impact.

There may be multiple impacts that result from some operations. In this case this function should be called multiple times, until it returns NULL. When processing of data from each successive impact is done, call `set_last_impact_been_processed()` to queue up the next impact object for processing.

The maximum number of unprocessed impacts is finite, and set by the function `set_max_unprocessed_impacts()`.

Note that the returned impact pointers are owned by DI-Guy and are temporary. The information you need from them should be immediately read, and the pointers should not be stored.

5.74.401 function diguyScenario::set_last_impact_been_processed**Prototype:**

```
void set_last_impact_been_processed();
```

Description:

Reading of data from the last impact information will be considered complete when this function is called.

5.74.402 function diguyScenario::set_last_env_impact_info**Prototype:**

```
void set_last_env_impact_info(const diguyImpact&impact);
```

Description:

This function will add the passed impact object to the list of impact objects to be processed by calls to [get_last_env_impact_info\(\)](#). An internal copy of the impact object is made.

5.74.403 function diguyScenario::set_max_unprocessed_impacts**Prototype:**

```
void set_max_unprocessed_impacts(int max_unprocessed_impacts);
```

Description:

Sets the maximum number of unprocessed impacts there can be.

The unprocessed impact array is a first-in-first-out queue. If an impact occurs that pushes the number of unprocessed impacts past this maximum number, earlier impacts will be dropped from the array.

The initial value is 10.

AI Line of Sight, Intersection Callback Functions

[diguyScenario](#) has a number of number of replaceable functions that govern intersecting and detecting the world. See the following functions to identify the full set of functionality that's necessary to make DI-Guy characters interact with the environment.

5.74.404 function [diguyScenario::set_default_point_los_function](#)

Prototype:

```
int set_default_point_los_function(diguyPointLOSFunction* point_los_function);
```

Description:

This function sets a default point line of sight function that will be added to all new characters. It can be overridden by an explicit call to [diguyCharacter::set_point_los_function\(\)](#). See that function for details.

Callable From:

◇ C++

5.74.405 function**diguyScenario::set_default_character_los_function****Prototype:**

```
int set_default_character_los_function(diguyCharacterLOSFunction* character_los_function);
```

Description:

This function sets a default character line of sight function that will be added to all new characters. It can be overridden by an explicit call to [diguyCharacter::set_character_los_function\(\)](#). See that function for details.

Callable From:

◇ C++

5.74.406 function**diguyScenario::set_default_agent_feeler_function****Prototype:**

```
int set_default_agent_feeler_function(diguyFeelerFunction* feeler_function);
```

Description:

This function sets a default feeler function that will be added to all new DI-Guy AI agents. It can be overridden by an explicit call to [diguyCharacter::agent_set_feeler_function\(\)](#).

See the "User-Defined Feelers" section in [diguyCharacter.h](#) for more information.

Arguments:

`feeler_function` pointer to user-defined feeler function

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.74.407 function diguyScenario::exp_intersect_geometry

Prototype:

```
diguyIntersectionResult exp_intersect_geometry(
    float origin_x, float origin_y, float origin_z,
    float dir_x,    float dir_y,    float dir_z,
    float max_distance_to_test,
    int use_callback = 1);
```

Description:

Experimental intersection-detection function.

Arguments:

origin_x, origin_y, origin_z	starting point
dir_x, dir_y, dir_z	direction vector
max_distance_to_test	distance to test to
use_callback	1 to use

Returns:

`diguyIntersectionResult` structure

Path Planning and Regions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.408 function diguyScenario::find_or_create_region**Prototype:**

```
diguyRegion* find_or_create_region(const char* name);
```

Description:

This function returns a pointer to the region with the given name. A new region with the given name will be created if it doesn't already exist.

Arguments:

name	name of the region to find or create

Returns:

pointer of type [diguyRegion](#); should never be NULL

5.74.409 function diguyScenario::find_region**Prototype:**

```
diguyRegion* find_region(const char* name);
```

Description:

This function returns a pointer to the specified region.

Arguments:

name	name of region to be found

Returns:

pointer of type [diguyRegion](#); NULL if not found

5.74.410 function diguyScenario::create_region**Prototype:**

```
diguyRegion* create_region(const char* name);
```

Description:

This function creates a new [diguyRegion](#) and returns a pointer to it.

Arguments:

name	name of the new region object

Returns:

pointer of type [diguyRegion](#)

5.74.411 function diguyScenario::destroy_region**Prototype:**

```
void destroy_region(diguyRegion* region);
```

Description:

This function destroys the passed region.

Arguments:

region	region to destroy

5.74.412 function diguyScenario::get_num_regions**Prototype:**

```
int get_num_regions();
```

Returns:

number of regions in the scenario

5.74.413 function diguyScenario::get_region_at_index**Prototype:**

```
diguyRegion* get_region_at_index(int index);
```

Returns:

pointer of type [diguyRegion](#); NULL if no region at the specified index

Arguments:

<code>index</code>	index of the region; indices start at 0

5.74.414 function `diguyScenario::find_navigation_path`

Prototype:

```
int find_navigation_path(float x1, float y1, float z1,
    float x2, float y2, float z2,
    const char* via_region,
    diguyPathShape* path_shape,
    float cost_bias_for_preferred_regions = 0.25f,
    int preferred_subregions_mask = DIGUY_SUBREGION_MASK_NONE,
    float cost_bias_for_neutral_regions = 1.1f,
    int repulsed_regions_mask = DIGUY_SUBREGION_MASK_NONE,
    float cost_bias_for_repulsive_regions = 100.0f);
```

Description:

The scenario will attempt to create a navigation path on the `via_region`. An A* path planning algorithm is used to find the path. See also [diguyCharacter::agent_move_to_point\(\)](#) and [diguyCharacter::agent_move_to_point_via_subregions\(\)](#).

Arguments:

<code>x1, y1, z1</code>	start location
<code>x2, y2, z2</code>	end location
<code>via_region</code>	name of the region to run A* on
<code>path_shape</code>	path shape that stores the created path
<code>preferred_subregions_mask</code>	diguySubregionMask value of regions preferred for travel
<code>cost_bias_for_preferred_regions</code>	how much cheaper it will be to cross spaces that are part of desired subregion; should be < 1.0
<code>cost_bias_for_neutral_regions</code>	how much more expensive it will be to cross spaces that are not part of desired subregion; see below for more information
<code>repulsed_regions_mask</code>	diguySubregionMask value of regions not preferred for travel
<code>cost_bias_for_repulsed_regions</code>	how much more expensive it will be to cross regions that are marked as repulsive; pass <code>DIGUY_DEFAULT_FLOAT</code> to avoid completely

Pass `DIGUY_DEFAULT_FLOAT` for `cost_bias_for_neutral_regions` and `cost_bias_for_repulsed_regions` to specify that they should be avoided completely.

The `preferred_subregions_mask` and `repulsed_regions_mask` use DI-Guy subregion mask values combined together. For example,

`DIGUY_SUBREGION_MASK_SIDEWALK |`
`DIGUY_SUBREGION_MASK_CROSSWALK.`

For `cost_bias_for_neutral_regions`, specifying a non-zero value for this lowers the likelihood that searches fail on disjointed subregions. A value < 1 will cause the planner to run faster but explore fewer points, possibly missing preferred regions. Values greater than 1 will explore more points but will be slower.

Returns:

0 on success, -1 on failure

5.74.415 function `diguyScenario::set_max_distance_planning_multiplier`

Prototype:

```
void set_max_distance_planning_multiplier(float mul);
```

Description:

Sets an upper limit on how far a character is willing to travel as a multiple of the straight line distance between point A and point B before declaring that it can't get to a location.

Note that internally the straight line distance is forced to have a lower bound of 10 meters, so a character is always willing to travel a minimum of $10 * \text{max\_distance\_planning\_multiplier}$ meters.

Defaults to -1, which means off.

5.74.416 function `diguyScenario::get_max_distance_planning_multiplier`

Prototype:

```
float get_max_distance_planning_multiplier();
```

Returns:

float constant that's used to decide if planner should give up; -1 means off

5.74.417 function `diguyScenario::get_time_spent_path_planning_this_frame`

Prototype:

```
float get_time_spent_path_planning_this_frame();
```

Description:

This function returns in seconds how much time has been spent path planning during the current frame. It can be used to avoid 100 characters simultaneously hitting the path planner.

Returns:

time in seconds

5.74.418 function**diguyScenario::set_multithreaded_path_planning_enabled****Prototype:**

```
void set_multithreaded_path_planning_enabled(int value);
```

Description:

This function enables the path planner to run in a background thread. Note this only occurs if the `diguyCharacter::agent_move_to_point_bg()` or `diguyCharacter::agent_move_to_point_via_subregions_bg()` api functions are used.

By default the multi-threaded path planner is on.

5.74.419 function**diguyScenario::get_multithreaded_path_planning_enabled****Prototype:**

```
int get_multithreaded_path_planning_enabled();
```

Returns:

Returns 1 if the path planner is enabled to run in a background thread. By default the multi-threaded path planner is on.

Formation Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.420 function diguyScenario::get_num_formation

Prototype:

```
int get_num_formation();
```

Returns:

number of formations in the scenario

5.74.421 function diguyScenario::get_formation_at_index**Prototype:**

```
diguyFormation* get_formation_at_index(int index);
```

Returns:

pointer of type [diguyFormation](#); NULL if no formation at the specified index

Arguments:

index	index of the formation; indices start at 0

5.74.422 function diguyScenario::find_formation**Prototype:**

```
diguyFormation* find_formation(const char* name);
```

Description:

This function returns a pointer to the specified formation.

Arguments:

name	name of formation to be found

Returns:

pointer of type [diguyFormation](#); NULL if not found

5.74.423 function diguyScenario::find_formation_of_size**Prototype:**

```
diguyFormation* find_formation_of_size(int size);
```

Description:

This function returns a pointer to the first formation it finds of a given size.

Arguments:

size	size of formation to be found

Returns:

pointer of type [diguyFormation](#); NULL if not found

5.74.424 function diguyScenario::create_formation**Prototype:**

```
diguyFormation* create_formation(const char* name);
```

Description:

This function creates a new formation and returns a pointer to it.

Arguments:

name	name of the new formation

Returns:

pointer of type [diguyFormation](#)

5.74.425 function diguyScenario::find_or_create_formation**Prototype:**

```
diguyFormation* find_or_create_formation(const char* name);
```

Description:

This function finds the formation with the given name or creates it if it doesn't exist.

Arguments:

name	name of the formation to find or create

Returns:

pointer of type [diguyFormation](#); should never be NULL

5.74.426 function diguyScenario::destroy_formation**Prototype:**

```
int destroy_formation(diguyFormation* formation);
```

Description:

This function destroys a formation.

Arguments:

formation	pointer to a diguyFormation

Returns:

0 on success, -1 on failure

DI-Guy Scenario User Interface Functions

These functions show or hide various supplemental visuals in the DI-Guy Scenario 3D window. They will have no effect outside of DI-Guy Scenario.

They correspond to the UI in the DI-Guy Scenario page Scenario Objects | Scenario Objects | Visibility.

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.427 function diguyScenario::set_action_bead_labels_visible

Prototype:

```
void set_action_bead_labels_visible(diguyVisibleFlag vflag);
```


5.74.428 function diguyScenario::set_aim_trajectories_visible**Prototype:**

```
void set_aim_trajectories_visible(diguyVisibleFlag vflag);
```

**5.74.429 function
diguyScenario::set_author_selection_handles_visible****Prototype:**

```
void set_author_selection_handles_visible(diguyVisibleFlag vflag);
```

5.74.430 function diguyScenario::set_bead_array_visible**Prototype:**

```
void set_bead_array_visible(diguyVisibleFlag vflag);
```

5.74.431 function diguyScenario::set_character_labels_visible**Prototype:**

```
void set_character_labels_visible(diguyVisibleFlag vflag);
```

5.74.432 function diguyScenario::set_character_visible**Prototype:**

```
void set_character_visible(diguyVisibleFlag vflag);
```

5.74.433 function diguyScenario::set_crowd_behavior_visible**Prototype:**

```
void set_crowd_behavior_visible(diguyVisibleFlag flag);
```

5.74.434 function diguyScenario::set_crowd_feelers_visible**Prototype:**

```
void set_crowd_feelers_visible(diguyVisibleFlag flag);
```

5.74.435 function diguyScenario::set_crowd_influence_visible**Prototype:**

```
void set_crowd_influence_visible(diguyVisibleFlag flag);
```

5.74.436 function diguyScenario::set_crowd_regions_visible**Prototype:**

```
void set_crowd_regions_visible(diguyVisibleFlag flag);
```

5.74.437 function diguyScenario::set_light_array_visible**Prototype:**

```
void set_light_array_visible(diguyVisibleFlag vflag);
```

5.74.438 function diguyScenario::set_lua_objects_visible**Prototype:**

```
void set_lua_objects_visible(diguyVisibleFlag flag);
```

5.74.439 function diguyScenario::set_mesh_region_array_visible**Prototype:**

```
void set_mesh_region_array_visible(diguyVisibleFlag vflag);
```

5.74.440 function
diguyScenario::set_sensor_region_array_visible**Prototype:**

```
void set_sensor_region_array_visible(diguyVisibleFlag vflag);
```

5.74.441 function diguyScenario::set_spath_visible**Prototype:**

```
void set_spath_visible(diguyVisibleFlag vflag);
```

5.74.442 function diguyScenario::set_waypoint_array_visible**Prototype:**

```
void set_waypoint_array_visible(diguyVisibleFlag vflag);
```

5.74.443 function diguyScenario::get_action_bead_labels_visible**Prototype:**

```
diguyVisibleFlag get_action_bead_labels_visible();
```


5.74.444 function diguyScenario::get_aim_trajectories_visible**Prototype:**

```
diguyVisibleFlag get_aim_trajectories_visible();
```

**5.74.445 function
diguyScenario::get_author_selection_handles_visible****Prototype:**

```
diguyVisibleFlag get_author_selection_handles_visible();
```

5.74.446 function diguyScenario::get_bead_array_visible**Prototype:**

```
diguyVisibleFlag get_bead_array_visible();
```

5.74.447 function diguyScenario::get_character_labels_visible**Prototype:**

```
diguyVisibleFlag get_character_labels_visible();
```

5.74.448 function diguyScenario::get_character_visible**Prototype:**

```
diguyVisibleFlag get_character_visible();
```

5.74.449 function diguyScenario::get_crowd_behavior_visible**Prototype:**

```
diguyVisibleFlag get_crowd_behavior_visible();
```

5.74.450 function diguyScenario::get_crowd_feelers_visible**Prototype:**

```
diguyVisibleFlag get_crowd_feelers_visible();
```

5.74.451 function diguyScenario::get_crowd_influence_visible**Prototype:**

```
diguyVisibleFlag get_crowd_influence_visible();
```

5.74.452 function diguyScenario::get_crowd_regions_visible**Prototype:**

```
diguyVisibleFlag get_crowd_regions_visible();
```

5.74.453 function diguyScenario::get_light_array_visible**Prototype:**

```
diguyVisibleFlag get_light_array_visible();
```

5.74.454 function diguyScenario::get_lua_objects_visible**Prototype:**

```
diguyVisibleFlag get_lua_objects_visible();
```

5.74.455 function diguyScenario::get_mesh_region_array_visible**Prototype:**

```
diguyVisibleFlag get_mesh_region_array_visible();
```

5.74.456 function
diguyScenario::get_sensor_region_array_visible**Prototype:**

```
diguyVisibleFlag get_sensor_region_array_visible();
```

5.74.457 function diguyScenario::get_spath_visible**Prototype:**

```
diguyVisibleFlag get_spath_visible();
```

5.74.458 function diguyScenario::get_waypoint_array_visible**Prototype:**

```
diguyVisibleFlag get_waypoint_array_visible();
```

5.74.459 function diguyScenario::set_hide_author_selection_handles_during_play**Prototype:**

```
void set_hide_author_selection_handles_during_play(int visible);
```


5.74.460 function**diguyScenario::get_hide_author_selection_handles_during_play****Prototype:**

```
int get_hide_author_selection_handles_during_play();
```

DI-Guy Weapon Fire Effects Functions

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.461 function
diguyScenario::get_weapon_fire_effects_enabled**Prototype:**

```
int get_weapon_fire_effects_enabled();
```

Returns:

If weapon fire will create light flashes.

5.74.462 function
diguyScenario::set_weapon_fire_effects_enabled**Prototype:**

```
int set_weapon_fire_effects_enabled(int enabled);
```

Description:

Sets if weapon fire will create light flashes.

5.74.463 function diguyScenario::get_num_active_fire_effects**Prototype:**

```
int get_num_active_fire_effects();
```

Returns:

The number of light flashes that are currently active

5.74.464 function diguyScenario::get_weapon_fire_effect_data**Prototype:**

```
int get_weapon_fire_effect_data(int index, float * radius,
    float * position_x, float * position_y, float * position_z,
    float * color_r, float * color_g, float * color_b,
    float * falloff_r,
    float * falloff_rsqr);
```

Description:

This function gets the data needed for producing a weapon flash light source in the world. See `diguyOglGraphicsShaderProgram::update_light_uniforms()` for an example of its use. In the default OpenGL renderer the convenience function `diguy_ogl_apply_active_lights()` does this work for you.

DI-Guy Scenario selection functions

These functions allow you to get and set the currently active item in the DI-Guy Scenario UI.

They correspond to the UI in the DI-Guy Scenario Scenario Objects page.

Unless otherwise specified, callable from:

- ◆ C++
- ◆ Script

5.74.465 function diguyScenario::set_current_character

Prototype:

```
int set_current_character(diguyCharacter* current_character);
```

Description:

Sets the current selected character in scenario

Returns:

0 on success, -1 on failure

5.74.466 function diguyScenario::get_current_character**Prototype:**

```
diguyCharacter* get_current_character();
```

Returns:

The current selected character in scenario

5.74.467 function diguyScenario::set_current_crowd**Prototype:**

```
int set_current_crowd(diguyCrowd* current_crowd);
```

Description:

Sets the current selected crowd in scenario

Returns:

0 on success, -1 on failure

5.74.468 function diguyScenario::get_current_crowd**Prototype:**

```
diguyCrowd* get_current_crowd();
```

Returns:

The current selected crowd in scenario

5.74.469 function diguyScenario::set_current_crowd_profile**Prototype:**

```
int set_current_crowd_profile(diguyCrowdProfile* current_profile);
```

Description:

Sets the active crowd profile, which determines the attributes of subsequently-created crowds. See [diguyCrowdProfile](#).

Returns:

0 on success, -1 on failure

5.74.470 function diguyScenario::set_current_crowd_profile_by_name**Prototype:**

```
int set_current_crowd_profile_by_name(const char* crowd_profile_name);
```

Description:

Sets the active crowd profile by name.

Returns:

0 on success, -1 on failure

5.74.471 function diguyScenario::get_current_crowd_profile**Prototype:**

```
diguyCrowdProfile* get_current_crowd_profile();
```

Returns:

Active crowd profile

5.74.472 function diguyScenario::set_current_region**Prototype:**

```
int set_current_region(diguyRegion* current_region);
```

Description:

Sets the current region. See [diguyRegion](#).

Returns:

0 on success, -1 on failure

5.74.473 function diguyScenario::get_current_region**Prototype:**

```
diguyRegion* get_current_region();
```

Returns:

Current region.

Particle System Functions

5.74.474 function diguyScenario::set_particle_module_disabled

Prototype:

```
void set_particle_module_disabled(int disable_particle_module);
```

Description:

This function enables or disables the particle module. By default the particle module is enabled.

5.74.475 function diguyScenario::create_particle_system**Prototype:**

```
int create_particle_system(const char* description_name,
    float x, float y, float z,
    int record_transient_event = 1,
    float duration = -1.0f);
```

Description:

This function creates a particle system with type `description_name` at `x`, `y`, `z`. If duration is specified the system will automatically stop emitting after that amount of time.

Arguments:

<code>description_name</code>	name of particle description to create
<code>x</code> , <code>y</code> , <code>z</code>	the world space position of the system
<code>record_transient_event</code>	pass 1 to have this particle system play in history playback
<code>duration</code>	how long the emitter should emit; if -1 is specified then the description must have a lifetime specified

5.74.476 function diguyScenario::create_particle_system_with_orientation**Prototype:**

```
int create_particle_system_with_orientation(const char* description_name,
    float x, float y, float z,
    float rz, float rx, float ry,
    int record_transient_event = 1,
    float duration = -1.0f);
```

Description:

Similar to `create_particle_system()`, but allows orientation to be set.

Arguments:

<code>rz</code> , <code>rx</code> , <code>ry</code>	orientation axes

(See `create_particle_system()` for other parameters)

5.74.477 function diguyScenario::has_particle_description**Prototype:**

```
int has_particle_description(const char* description_name);
```

Returns:

1 if the particle description named description_name exists, else 0

5.74.478 function diguyScenario::set_global_wind**Prototype:**

```
void set_global_wind(float x, float y, float z);
```

Description:

Set wind velocity vector for particle system

Lua Scripting Module Functions

5.74.479 function diguyScenario::lua_get_object_field_as_string

Prototype:

```
const char* lua_get_object_field_as_string(const char* lua_object,  
const char* field_name);
```

Returns:

string representation of lua_object.field_name

Note that the returned string pointer will not remain valid, so the returned string should be copied.

Arguments:

lua_object a global lua object

field_name name of the field; field_name can include ".", allowing retrieval of fields in complex data structures

Callable From:

◇ C++

5.74.480 function diguyScenario::lua_evaluate_object_function**Prototype:**

```
const char* lua_evaluate_object_function(const char* lua_object,  
    const char* function_name,  
    const char* argument = NULL,  
    int has_return_string = 0);
```

Description:

Runs a member function of a lua object.

Arguments:

lua_object	a global lua object
function_name	name of the function to call
argument	optional string argument
has_return_string	if set to 1 function will pop the top value of the lua stack and return it as a string

Returns:

NULL or string representation of lua_object:function_name(argument).

The returned string should be copied if it needs to be used later.

Callable From:

◇ C++

5.74.481 function diguyScenario::lua_evaluate_object_function_2a

Prototype:

```
const char* lua_evaluate_object_function_2a(const char* lua_object,
const char* function_name,
const char* argument,
const char* argument2,
int has_return_string = 0);
```

Description:

A two argument version of [lua_evaluate_object_function\(\)](#).

Callable From:

◇ C++

5.74.482 function**diguyScenario::lua_evaluate_object_function_3a****Prototype:**

```
const char* lua_evaluate_object_function_3a(const char* lua_object,
    const char* function_name,
    const char* argument,
    const char* argument2,
    const char* argument3,
    int has_return_string = 0);
```

Description:

A three argument version of [lua_evaluate_object_function\(\)](#).

Callable From:

◇ C++

5.74.483 function**diguyScenario::lua_evaluate_object_function_4a****Prototype:**

```
const char* lua_evaluate_object_function_4a(const char* lua_object,
    const char* function_name,
    const char* argument,
    const char* argument2,
    const char* argument3,
    const char* argument4,
    int has_return_string = 0);
```

Description:

A four argument version of [lua_evaluate_object_function\(\)](#).

Callable From:

◇ C++

5.74.484 function diguyScenario::lua_send_message_to_object**Prototype:**

```
int lua_send_message_to_object(const char* lua_object,
    const char* sender,
    const char* message_type,
    const char* message,
    const char* message_params = NULL);
```

Description:

Executes a lua_object:state_manager() function call. Useful for creating sleep-able coroutine based objects that aren't characters.

Callable From:

◇ C++

5.74.485 function diguyScenario::get_lua_state**Prototype:**

```
void* get_lua_state();
```

Description:

This function allows low level access to the lua_State pointer. This pointer can be used by a programmer to query and run functions on the Lua virtual machine. This object can also be used to register new C functions to lua, which allows you to instrument and create callbacks from script to your code. See luaL_register on-line.

For more information on how the Lua C api works see:
<http://www.lua.org/pil24.html>

NOTE:

Currently all scenarios share the same underlying Lua state object. This may cause issues in applications with multiple scenarios.

Callable From:

◇ C++

5.74.486 function diguyScenario::push_qt_pointer_to_lua**Prototype:**

```
int push_qt_pointer_to_lua(void* lua_state,
    const char* class_name,
    void* ptr);
```

Description:

Mainly used to pass a qt pointer to lua so lqt can be used to modify/read from the widget. This requires the that the lqt package is loaded; see the lqt.lua utility package for more info.

Arguments:

<code>lua_state</code>	a Lua_State pointer, might be the same as <code>get_lua_state()</code> ; depends if the calling function is inside a coroutine
<code>class_name</code>	class name should be a core class of qt with a star after it; i.e. "QWidget*" or "QLineEdit"
<code>ptr</code>	pointer to that is returned to the calling function in lua

Callable From:

◇ C++

C++ Example:

```
int push_graphics_view_to_lua(lua_State *L)
{
    bdiQGraphicsView* graphics_view = get_primary_3d_window_graphics_view();
    if (m_scenario->push_qt_pointer_to_lua(L, "QGraphicsView*", graphics_view)
    {
        return 0;
    }
    return 1;
}
```

Plug-in init code:

```
lua_State* L = (lua_State*)scenario->get_lua_state();
if (L)
{
    lua_register(L, "get_graphics_view", push_graphics_view_to_lua);
}
```

In lua:

```
local graphics_view = get_graphics_view();
```

5.74.487 function**diguyScenario::lua_evaluate_object_function_4a_flex****Prototype:**

```
const char* lua_evaluate_object_function_4a_flex(const char* lua_object,
const char* function_name,
const char* arg1_type, void* argument1,
const char* arg2_type, void* argument2,
const char* arg3_type, void* argument3,
const char* arg4_type, void* argument4,
int has_return_string = 0);
```

Description:

The most open function calling function available, if this doesn't manage to accomplish what you need you probably want to start using the lua interpreter directly.

The type arguments are string versions of the type being sent to lua, options include:

- " – empty argument
- 'b' boolean – void* argument1 is assumed to be an int
- 'f' field – void* argument1 assumed to be character string of a field of a global lua object; supports nested fields like foo.bar.a
- 's' string – void* argument1 assumed to be character string
- 'd' double – void* argument is assumed to be a double
- diguy... – void* argument assumed to be a diguy class pointer ie "[diguyCharacter](#)", "[diguyCrowd](#)" (no star used for diguy classes)
- Q...* – void* argument is assumed to be a pointer to a child of QObject; the class name should be a core class of qt with a star after it, e.g. "QWidget*" or "QLineEdit"

Arguments:

lua_object	a global lua object
function_name	name of the function to call
arg(1/2/3/4)_type	is the data type
argument(1/2/3/4)	void* pointers to data
has_return_string	if set to 1 function will pop the top value of the lua stack and return it as a string

Returns:

NULL or string representation of lua_object:function_name(argument)

The returned string should be copied if it needs to be used later.

Callable From:

◇ C++

5.74.488 function diguyScenario::lua_evaluate_global_function_4a_flex

Prototype:

```
const char* lua_evaluate_global_function_4a_flex(const char* function_name,
const char* arg_type, void* argument,
const char* arg_type2, void* argument2,
const char* arg_type3, void* argument3,
const char* arg_type4, void* argument4,
int has_return_string = 0);
```

Description:

Effectively the same as
[diguyScenario::lua_evaluate_object_function_4a_flex\(\)](#)
 but works on global functions.

Callable From:

◇ C++

5.74.489 function diguyScenario::lua_log_printf**Prototype:**

```
int lua_log_printf(int notify_level, const char* string);
```

Description:

Prints a message to lua interpreter log object. In DI-Guy Scenario this is sent to the AI Mind Editor and used to fill out the error log. The Mind editor log parses `debug.traceback()` calls.

5.74.490 function diguyScenario::dump_package_info**Prototype:**

```
const char* dump_package_info(const char* package_name);
```

Returns:

package meta data as a formatted string; useful for debugging, only valid inside DI-Guy Scenario

Experimental Functions

The following functions are experimental and may change or disappear with no warning.

5.74.491 function diguyScenario::get_eval_decisions_as_scripts

Prototype:

```
int get_eval_decisions_as_scripts();
```

Returns:

value set by last call to `set_eval_decisions_as_scripts()`

5.74.492 function diguyScenario::set_eval_decisions_as_scripts**Prototype:**

```
void set_eval_decisions_as_scripts(int eval_decisions_as_scripts);
```

Description:

If called with an argument of 1, all decision beads and decision beads will be converted to scripts before being run.

5.74.493 function diguyScenario::get_impact_on_line**Prototype:**

```
diguyImpact* get_impact_on_line(float from_x,  
    float from_y,  
    float from_z,  
    float to_x,  
    float to_y,  
    float to_z);
```

Description:

Gets the point of impact with the world along specified line segment

Returns:

pointer to [diguyImpact](#) object or NULL

5.74.494 function diguyScenario::wait_cursor_push**Prototype:**

```
void wait_cursor_push();
```

Description:

Increments wait cursor counter. If non-zero, then wait cursor is displayed in DI-GUY scenario.

5.74.495 function diguyScenario::wait_cursor_pop**Prototype:**

```
void wait_cursor_pop();
```

Description:

Decrements wait cursor counter.

5.74.496 function diguyScenario::set_graphics_api_node_ptr**Prototype:**

```
void set_graphics_api_node_ptr(void* node_ptr);
```

Description:

This function sets a generic node pointer that can later be retrieved by the [get_graphics_api_node_ptr \(\)](#) call. The pointer is otherwise not used.

Arguments:

node_ptr generic void* pointer

Callable From:

◇ C++

5.74.497 function diguyScenario::get_graphics_api_node_ptr**Prototype:**

```
void* get_graphics_api_node_ptr();
```

Returns:

pointer set by most recent call to set_graphics_api_node_ptr

Callable From:

◇ C++

5.74.498 function**diguyScenario::set_draw_character_shadow_disks****Prototype:**

```
void set_draw_character_shadow_disks(int draw_character_shadow_disks);
```

Description:

Sets whether shadow disks are drawn for characters. Call with 1 to turn on, 0 to turn off.

5.74.499 function**diguyScenario::get_draw_character_shadow_disks****Prototype:**

```
int get_draw_character_shadow_disks();
```

Returns:

1 if shadow disks are drawn, 0 if not

5.74.500 function diguyScenario::get_shadow_map_size**Prototype:**

```
int get_shadow_map_size();
```

Returns:

size of texture map for shadows

5.74.501 function diguyScenario::set_shadow_map_size**Prototype:**

```
int set_shadow_map_size(int shadow_size);
```

Description:

Sets size of texture map for shadows. Defaults to 1024. Allowed values are powers of two up to 4096.

5.74.502 function**diguyScenario::set_default_character_scene_graph_mask****Prototype:**

```
void set_default_character_scene_graph_mask(unsigned long mask);
```

Description:

Used for DI-Guy OSG Author Programming Example. Sets the default value used for characters. See also [diguyCharacter::set_scene_graph_mask\(\)](#)

5.74.503 function**diguyScenario::set_default_scene_object_scene_graph_mask****Prototype:**

```
void set_default_scene_object_scene_graph_mask(unsigned long mask);
```

Description:

Used for DI-Guy OSG Author Programming Example. Sets the default value used for characters. See also [diguySceneObject::set_scene_graph_mask\(\)](#)

5.74.504 function diguyScenario::set_internal_int**Prototype:**

```
void set_internal_int(int var, int val);
```

Description:

For internal use.

5.74.505 function diguyScenario::set_internal_float**Prototype:**

```
void set_internal_float(int var, float val);
```

5.74.506 function diguyScenario::set_num_active_point_lights**Prototype:**

```
int set_num_active_point_lights(int active_lights);
```

Description:

This sets the number of point lights that get handed down to `diguyGraphicsShaderTechnique::pick_shader_program()`.

5.74.507 function diguyScenario::get_num_active_point_lights**Prototype:**

```
int get_num_active_point_lights();
```

Description:

This gets the number of point lights that get handed down to `diguyGraphicsShaderTechnique::pick_shader_program()` the default is 0. `diguy_ogl_apply_active_lights()` automatically updates this value in the OpenGL renderer.

Experimental Performance Functions

5.74.508 function diguyScenario::optimized_update

Prototype:

```
int optimized_update(float t);
```

Description:

Experimental. For internal use.

5.74.509 function diguyScenario::set_world_bounds**Prototype:**

```
int set_world_bounds(float x_min, float y_min, float z_min,
                    float x_max, float y_max, float z_max);
```

Description:

Sets bounding box of world

Arguments:

x_min, y_min, z_min, x_max, y_max, z_max bounds

Returns:

0 on success, -1 on failure (float * arguments converted to return values in Lua)

5.74.510 function diguyScenario::get_world_bounds**Prototype:**

```
int get_world_bounds(float* x_min, float* y_min, float* z_min,
                    float* x_max, float* y_max, float* z_max);
```

Description:

Gets bounding box of world

Arguments:

x_min, y_min, z_min, x_max, y_max, z_max output parameters

Returns:

0 on success, -1 on failure (float * arguments converted to return values in Lua)

Experimental Checkpoint Functions

5.74.511 function diguyScenario::set_checkpoint_frequency

Prototype:

```
void set_checkpoint_frequency(float freq);
```

Description:

Internal use

5.74.512 function diguyScenario::get_checkpoint_frequency

Prototype:

```
float get_checkpoint_frequency();
```

Description:

Internal use

5.74.513 function diguyScenario::set_checkpointing_enabled

Prototype:

```
void set_checkpointing_enabled(int enable_checkpointing);
```

Description:

Internal use

5.74.514 function diguyScenario::get_checkpointing_enabled**Prototype:**

```
int get_checkpointing_enabled();
```

Description:

Internal use

Experimental Shader Functions

5.74.515 function diguyScenario::set_initial_render_mode

Prototype:

```
void set_initial_render_mode(const char* render_mode);
```

Description:

Sets initial render mode

Arguments:

`render_mode` string containing render mode setting name

Built-in render modes:

5.74.516 function diguyScenario::get_initial_render_mode

Prototype:

```
const char* get_initial_render_mode();
```

Returns:

initial render mode, as string

5.74.517 function diguyScenario::set_current_render_mode**Prototype:**

```
void set_current_render_mode(const char* render_mode);
```

Description:

Sets current render mode. See [set_initial_render_mode\(\)](#)

5.74.518 function diguyScenario::derive_render_mode_from_light_settings**Prototype:**

```
const char* derive_render_mode_from_light_settings(int set_current_to_derived = 1);
```

Returns:

render mode, as string, derived from current light settings

Arguments:

`set_current_to_derived` if 1, current render mode becomes derived

5.74.519 function diguyScenario::get_current_render_mode**Prototype:**

```
const char* get_current_render_mode();
```

Returns:

current render mode, as string

5.74.520 function diguyScenario::set_default_character_shader**Prototype:**

```
void set_default_character_shader(const char* render_mode,
    const char* shader_name,
    int update_existing_characters = 1);
```

Description:

Sets default shader for characters

Arguments:

render_mode	string containing render mode name
shader_name	string containing name of shader technique
update_existing_characters	if 1, all existing characters will now

5.74.521 function diguyScenario::get_default_character_shader**Prototype:**

```
const char* get_default_character_shader(const char* render_mode);
```

Description:

Returns name of shader technique used for specified render mode

5.74.522 function diguyScenario::set_default_scene_object_shader**Prototype:**

```
void set_default_scene_object_shader(const char* render_mode,
    const char* shader_name,
    int update_existing_scene_objects = 1);
```

Description:

Sets default shader for scene objects

Arguments:

render_mode	string containing render mode name
shader_name	string containing name of shader technique
update_existing_scene_objects	if 1, all existing scene objects will now

5.74.523 function

diguyScenario::get_default_scene_object_shader

Prototype:

```
const char* get_default_scene_object_shader(const char* render_mode);
```

Description:

Returns name of shader technique used for specified render mode

5.74.524 function

diguyScenario::set_default_particle_system_shader

Prototype:

```
void set_default_particle_system_shader(const char* render_mode,
    const char* shader_name,
    int update_existing_particle_systems = 1);
```

Description:

Sets default shader for particle systems

Arguments:

render_mode	string containing render mode name
shader_name	string containing name of shader technique
update_existing_particle_systems	if 1, all existing particle systems will now

5.74.525 function diguyScenario::get_default_particle_system_shader

Prototype:

```
const char* get_default_particle_system_shader(const char* render_mode);
```

Description:

Returns name of shader technique used for specified render mode

5.74.526 function diguyScenario::facefx_mount_animset

Prototype:

```
int facefx_mount_animset(const char* actor_name, const char* file_name);
```

Description:

Sets FaceFX animation set for a given actor.

Arguments:

`actor_name` actor's name

`file_name` fully qualified file name ending in ".animset_ingame"

5.74.527 function `diguyScenario::set_vehicle_smoothing_enabled`

Prototype:

```
int set_vehicle_smoothing_enabled(int enable_smoothing);
```

Description:

Sets vehicle smoothing on or off. If 1, vehicle will be updated at a higher rate than frame dt

Default is on.

5.74.528 function `diguyScenario::get_vehicle_smoothing_enabled`

Prototype:

```
int get_vehicle_smoothing_enabled();
```

5.74.529 function diguyScenario::set_character_culling_distance**Prototype:**

```
int set_character_culling_distance(float dist);
```

Description:

Culling distance (from camera) for drawing

5.74.530 function diguyScenario::get_character_culling_distance**Prototype:**

```
float get_character_culling_distance();
```

5.74.531 function diguyScenario::set_vehicle_culling_distance**Prototype:**

```
int set_vehicle_culling_distance(float dist);
```

Description:

Culling distance (from camera)

5.74.532 function diguyScenario::get_vehicle_culling_distance**Prototype:**

```
float get_vehicle_culling_distance();
```

5.74.533 function diguyScenario::set_prop_culling_distance**Prototype:**

```
int set_prop_culling_distance(float dist);
```

Description:

Culling distance of props (from camera)

5.74.534 function diguyScenario::get_prop_culling_distance**Prototype:**

```
float get_prop_culling_distance();
```

Deprecated Functions

The following functions are deprecated.

As these functions may disappear in a future version of DI-Guy, it is recommended that you remove calls to these functions from your application.

5.74.535 function `diguyScenario::add_default_callback`

Prototype:

```
static int add_default_callback(int callback_id,  
    diguyScenarioCallback* callback,  
    void* callback_params = 0,  
    void* callback_user_data = 0);
```

Description:

Deprecated; use
`diguyApp::add_default_scenario_callback()` instead.

5.74.536 function diguyScenario::remove_default_callback**Prototype:**

```
static int remove_default_callback(int callback_id,  
    diguyScenarioCallback* callback);
```

Description:

Deprecated; use
`diguyApp::remove_default_scenario_callback( )` instead.

5.74.537 function diguyScenario::remove_default_callback_with_user_data**Prototype:**

```
static int remove_default_callback_with_user_data(int callback_id,  
    void* callback_user_data);
```

Description:

Deprecated; use
`diguyApp::remove_default_scenario_callback_with_User_data( )` instead.

5.74.538 function diguyScenario::add_default_callback_script**Prototype:**

```
static int add_default_callback_script(int callback_id,  
    const char* callback_script,  
    const char* callback_script_type);
```

Description:

Deprecated; use
[diguyApp::add_default_scenario_callback_script\(\)](#)
instead.

**5.74.539 function
diguyScenario::remove_default_callback_script****Prototype:**

```
static int remove_default_callback_script(int callback_id,  
    const char* callback_script,  
    const char* callback_script_type);
```

Description:

Deprecated; use
[diguyApp::remove_default_scenario_callback_script\(\)](#)
instead.

5.74.540 function diguyScenario::get_character_type_class**Prototype:**

```
const char* get_character_type_class(const char* character_type);
```

Description:

Deprecated as of 10.0.0; use [get_character_type_map\(\)](#) instead, and then call

[diguyCharacterTypeMap::get_field_value\(DIGUY_CHARACTER_TYPE_MAP_FIELD](#)

5.74.541 function diguyScenario::merge_asset**Prototype:**

```
int merge_asset(const char* string);
```

Description:

Deprecated as of 10.5.1; use [merge_object\(\)](#) instead.

5.74.542 function diguyScenario::print_to_log**Prototype:**

```
void print_to_log(int notify_level, const char* string);
```

Description:

Deprecated as of 12.0.0; use `bdi_log_print()` from libbdilog.h instead.

VegaPrime Helper Functions

5.74.543 function**diguyScenario::create_pending_reflected_character****Prototype:**

```
diguyCharacter* create_pending_reflected_character(const char* name,  
    const char* character_type,  
    const char* appearance = NULL);
```

5.74.544 function diguyScenario::set_network_translation**Prototype:**

```
int set_network_translation(float x, float y, float z);
```

5.74.545 function diguyScenario::get_network_translation**Prototype:**

```
int get_network_translation(float* x, float* y, float* z);
```

5.74.546 function diguyScenario::set_project_filename**Prototype:**

```
void set_project_filename(const char* project_filename);
```

Description:

Deprecated as of 12.0.0; use [save_as](#)() instead.

5.74.547 function diguyScenario::get_project_filename**Prototype:**

```
const char* get_project_filename();
```

Description:

Deprecated as of 12.0.0; use [get_filename\(\)](#) or [get_filename_without_directory\(\)](#) instead.

5.75 diguyScenarioMergeSettings

diguyScenarioMergeSettings.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Offset Functions](#)
- ◆ [Merge Flags](#)
- ◆ [Misc Settings](#)

Link against: **libdiguy**

5.75.1 class diguyScenarioMergeSettings

```
class BDI_DECLSPEC_diguy diguyScenarioMergeSettings
{
public:
    int set_time_offset(float t);
    float get_time_offset();
    int set_position_offset(float tx, float ty, float tz);
    int get_position_offset(float* tx, float* ty, float* tz);
    int set_orientation_offset(float rz, float rx, float ry);
    int get_orientation_offset(float* rz, float* rx, float* ry);
    int set_transformation_reference_point(float tx, float ty, float tz);
    int get_transformation_reference_point(float* tx, float* ty, float* tz);
    int set_reclamp_waypoints_flag(int reclamp_waypoints_flag);
    int get_reclamp_waypoints_flag();
    void set_all_merge_flags_to_dont_merge();
    int set_camera_settings_merge_flag(diguyScenarioMergeFlag merge_flag);
    diguyScenarioMergeFlag get_camera_settings_merge_flag();
    int set_chain_settings_merge_flag(diguyScenarioMergeFlag merge_flag);
    diguyScenarioMergeFlag get_chain_settings_merge_flag();
    int set_character_merge_flag(diguyScenarioMergeFlag merge_flag);
    diguyScenarioMergeFlag get_character_merge_flag();
    int set_crowd_merge_flag(diguyScenarioMergeFlag merge_flag);
    diguyScenarioMergeFlag get_crowd_merge_flag();
    int set_crowd_profile_merge_flag(diguyScenarioMergeFlag merge_flag);
    diguyScenarioMergeFlag get_crowd_profile_merge_flag();
    int set_decision_merge_flag(diguyScenarioMergeFlag merge_flag);
    diguyScenarioMergeFlag get_decision_merge_flag();
    int set_face_exp_merge_flag(diguyScenarioMergeFlag merge_flag);
    diguyScenarioMergeFlag get_face_exp_merge_flag();
    int set_fog_settings_merge_flag(diguyScenarioMergeFlag merge_flag);
    diguyScenarioMergeFlag get_fog_settings_merge_flag();
    int set_formation_merge_flag(diguyScenarioMergeFlag merge_flag);
    diguyScenarioMergeFlag get_formation_merge_flag();
    int set_group_merge_flag(diguyScenarioMergeFlag merge_flag);
    diguyScenarioMergeFlag get_group_merge_flag();
    int set_guide_merge_flag(diguyScenarioMergeFlag merge_flag);
    diguyScenarioMergeFlag get_guide_merge_flag();
};
```

```

int set_info_popup_merge_flag(diguyScenarioMergeFlag merge_flag);
diguyScenarioMergeFlag get_info_popup_merge_flag();
int set_library_function_merge_flag(diguyScenarioMergeFlag merge_flag);
diguyScenarioMergeFlag get_library_function_merge_flag();
int set_light_settings_merge_flag(diguyScenarioMergeFlag merge_flag);
diguyScenarioMergeFlag get_light_settings_merge_flag();
int set_particle_system_merge_flag(diguyScenarioMergeFlag merge_flag);
diguyScenarioMergeFlag get_particle_system_merge_flag();
int set_path_shape_merge_flag(diguyScenarioMergeFlag merge_flag);
diguyScenarioMergeFlag get_path_shape_merge_flag();
int set_region_merge_flag(diguyScenarioMergeFlag merge_flag);
diguyScenarioMergeFlag get_region_merge_flag();
int set_scenario_settings_merge_flag(diguyScenarioMergeFlag merge_flag);
diguyScenarioMergeFlag get_scenario_settings_merge_flag();
int set_scene_object_merge_flag(diguyScenarioMergeFlag merge_flag);
diguyScenarioMergeFlag get_scene_object_merge_flag();
int set_scene_object_grid_merge_flag(diguyScenarioMergeFlag merge_flag);
diguyScenarioMergeFlag get_scene_object_grid_merge_flag();
int set_script_merge_flag(diguyScenarioMergeFlag merge_flag);
diguyScenarioMergeFlag get_script_merge_flag();
int set_sensor_region_merge_flag(diguyScenarioMergeFlag merge_flag);
diguyScenarioMergeFlag get_sensor_region_merge_flag();
int set_signal_merge_flag(diguyScenarioMergeFlag merge_flag);
diguyScenarioMergeFlag get_signal_merge_flag();
int set_sound_merge_flag(diguyScenarioMergeFlag merge_flag);
diguyScenarioMergeFlag get_sound_merge_flag();
int set_variable_merge_flag(diguyScenarioMergeFlag merge_flag);
diguyScenarioMergeFlag get_variable_merge_flag();
int set_reset_flag(diguyScenarioMergeResetFlag reset_flag);
diguyScenarioMergeResetFlag get_reset_flag();
int set_rename_prefix(const char* rename_prefix);
const char* get_rename_prefix();
int set_rename_suffix(const char* rename_suffix);
const char* get_rename_suffix();
int set_character_group(const char* group_name);
const char* get_character_group();

```

Offset Functions

The following functions allow characters and other objects in the merged scenario to be offset in time and/or space.

5.75.2 function diguyScenarioMergeSettings::set_time_offset

Prototype:

```
int set_time_offset(float t);
```

Description:

This function sets the time offset, in seconds, of the merged scenario from the existing scenario.

The default value is 0, or no time offset.

Arguments:

t	time offset in seconds; must be ≥ 0

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.75.3 function diguyScenarioMergeSettings::get_time_offset

Prototype:

```
float get_time_offset();
```

Returns:

time offset as set by [set_time_offset\(\)](#)

Callable From:

◇ C++
◇ Script

5.75.4 function diguyScenarioMergeSettings::set_position_offset**Prototype:**

```
int set_position_offset(float tx, float ty, float tz);
```

Description:

This function sets the position offset, in meters, of the merged scenario from its original position.

The default values are 0,0,0, or no position change.

Arguments:

tx, ty,	position offset in meters
tz	

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.5 function diguyScenarioMergeSettings::get_position_offset

Prototype:

```
int get_position_offset(float* tx, float* ty, float* tz);
```

Returns:

position offset as set by [set_position_offset\(\)](#)

Arguments:

tx, ty, position offset in meters
tz

Pass NULL for any values that are not needed.

Callable From:

- ◇ C++
- ◇ Script

5.75.6 function diguyScenarioMergeSettings::set_orientation_offset

Prototype:

```
int set_orientation_offset(float rz, float rx, float ry);
```

Description:

This function sets the orientation offset, in degrees, of the merged scenario from its original orientation. The rotation is done about the point set by the function [set_transformation_reference_point\(\)](#) below.

The default values are 0,0,0, or no orientation change.

Arguments:

rz, rx, orientation offset in degrees
ry

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.7 function diguyScenarioMergeSettings::get_orientation_offset

Prototype:

```
int get_orientation_offset(float* rz, float* rx, float* ry);
```

Returns:

orientation offset as set by [set_orientation_offset\(\)](#)

Arguments:

rz, rx, ry	orientations in degrees counter-clockwise from the positive X axis

Pass NULL for any values that are not needed.

Callable From:

- ◇ C++
- ◇ Script

5.75.8 function `diguyScenarioMergeSettings::set_transformation_reference_poin`

Prototype:

```
int set_transformation_reference_point(float tx, float ty, float tz);
```

Description:

This function sets the point about which the orientation offset rotation will be done.

The default values are 0,0,0, or the origin.

Arguments:

<code>tx, ty,</code>	reference point location in meters from the origin
<code>tz</code>	

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.9 function**diguyScenarioMergeSettings::get_transformation_reference_point****Prototype:**

```
int get_transformation_reference_point(float* tx, float* ty, float* tz);
```

Returns:

reference point location as set by [set_position_offset\(\)](#)

Arguments:

tx, ty, tz reference point location in meters from the origin

Pass NULL for any values that are not needed.

Callable From:

- ◇ C++
- ◇ Script

5.75.10 function**diguyScenarioMergeSettings::set_reclamp_waypoints_flag****Prototype:**

```
int set_reclamp_waypoints_flag(int reclamp_waypoints_flag);
```

Description:

This function sets whether waypoints of any paths or path shapes in the merged scenario are ground clamped after they have had either a position or orientation offset applied.

The default value is 1, re-clamp waypoints.

Arguments:

reclamp_waypoints_flag pass 1 to re-clamp waypoints; pass 0 to not do so

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.11 function diguyScenarioMergeSettings::get_reclamp_waypoints_flag

Prototype:

```
int get_reclamp_waypoints_flag();
```

Returns:

whether to re-clamp waypoints, as set by
[set_reclamp_waypoints_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

Merge Flags

The following functions specify whether objects of specific types are merged at all, and if so, what to do if there is a conflict with an existing object.

The default values for these flags are determined by the `initial_settings` argument passed to the `diguyScenario::create_merge_settings()` function. See the documentation for this function for what the default values are.

5.75.12 function

`diguyScenarioMergeSettings::set_all_merge_flags_to_dont_merge()`

Prototype:

```
void set_all_merge_flags_to_dont_merge();
```

Description:

This function sets all merge flags to not merge. Individual flags can be modified to enable merges of specific types.

Callable From:

- ◇ C++
- ◇ Script

5.75.13 function diguyScenarioMergeSettings::set_camera_settings_merge_flag

Prototype:

```
int set_camera_settings_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged. Since there are a set number of camera settings the options for the merge_flag argument are limited.

Arguments:

merge_flag can be set only to
DIGUY_MERGE_FLAG_DONT_MERGE or
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.14 function**diguyScenarioMergeSettings::get_camera_settings_merge_flag****Prototype:**

```
diguyScenarioMergeFlag get_camera_settings_merge_flag();
```

Returns:

most recent value set by [set_camera_settings_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.15 function**diguyScenarioMergeSettings::set_chain_settings_merge_flag****Prototype:**

```
int set_chain_settings_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.75.16 function `diguyScenarioMergeSettings::get_chain_settings_merge_flag`

Prototype:

```
diguyScenarioMergeFlag get_chain_settings_merge_flag();
```

Returns:

most recent value set by `set_chain_settings_merge_flag()`

Callable From:

◇ C++
◇ Script

5.75.17 function diguyScenarioMergeSettings::set_character_merge_flag

Prototype:

```
int set_character_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.18 function diguyScenarioMergeSettings::get_character_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_character_merge_flag();
```

Returns:

most recent value set by [set_character_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.19 function diguyScenarioMergeSettings::set_crowd_merge_flag

Prototype:

```
int set_crowd_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.20 function diguyScenarioMergeSettings::get_crowd_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_crowd_merge_flag();
```

Returns:

most recent value set by [set_crowd_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.21 function

diguyScenarioMergeSettings::set_crowd_profile_merge_flag

Prototype:

```
int set_crowd_profile_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.22 function

diguyScenarioMergeSettings::get_crowd_profile_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_crowd_profile_merge_flag();
```

Returns:

most recent value set by [set_crowd_profile_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.23 function

diguyScenarioMergeSettings::set_decision_merge_flag

Prototype:

```
int set_decision_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.24 function

diguyScenarioMergeSettings::get_decision_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_decision_merge_flag();
```

Returns:

most recent value set by [set_decision_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.25 function

diguyScenarioMergeSettings::set_face_exp_merge_flag

Prototype:

```
int set_face_exp_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.26 function

diguyScenarioMergeSettings::get_face_exp_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_face_exp_merge_flag();
```

Returns:

most recent value set by [set_face_exp_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.27 function**diguyScenarioMergeSettings::set_fog_settings_merge_flag****Prototype:**

```
int set_fog_settings_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged. Since there are a set number of fog settings the options for the merge_flag argument are limited.

Arguments:

merge_flag can be set only to
DIGUY_MERGE_FLAG_DONT_MERGE or
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.28 function

diguyScenarioMergeSettings::get_fog_settings_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_fog_settings_merge_flag();
```

Returns:

most recent value set by [set_fog_settings_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.29 function

diguyScenarioMergeSettings::set_formation_merge_flag

Prototype:

```
int set_formation_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.75.30 function `diguyScenarioMergeSettings::get_formation_merge_flag`

Prototype:

```
diguyScenarioMergeFlag get_formation_merge_flag();
```

Returns:

most recent value set by `set_formation_merge_flag()`

Callable From:

◇ C++
◇ Script

5.75.31 function

diguyScenarioMergeSettings::set_group_merge_flag

Prototype:

```
int set_group_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.32 function

diguyScenarioMergeSettings::get_group_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_group_merge_flag();
```

Returns:

most recent value set by [set_group_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.33 function diguyScenarioMergeSettings::set_guide_merge_flag

Prototype:

```
int set_guide_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.34 function diguyScenarioMergeSettings::get_guide_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_guide_merge_flag();
```

Returns:

most recent value set by [set_guide_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.35 function**diguyScenarioMergeSettings::set_info_popup_merge_flag****Prototype:**

```
int set_info_popup_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.36 function**diguyScenarioMergeSettings::get_info_popup_merge_flag****Prototype:**

```
diguyScenarioMergeFlag get_info_popup_merge_flag();
```

Returns:

most recent value set by [set_info_popup_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.37 function**diguyScenarioMergeSettings::set_library_function_merge_flag****Prototype:**

```
int set_library_function_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.38 function**diguyScenarioMergeSettings::get_library_function_merge_flag****Prototype:**

```
diguyScenarioMergeFlag get_library_function_merge_flag();
```

Returns:

most recent value set by [set_library_function_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.39 function

diguyScenarioMergeSettings::set_light_settings_merge_flag

Prototype:

```
int set_light_settings_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged. Since there are a set number of light settings the options for the `merge_flag` argument are limited.

Arguments:

`merge_flag` can be set only to
DIGUY_MERGE_FLAG_DONT_MERGE or
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.40 function**diguyScenarioMergeSettings::get_light_settings_merge_flag****Prototype:**

```
diguyScenarioMergeFlag get_light_settings_merge_flag();
```

Returns:

most recent value set by [set_light_settings_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.41 function**diguyScenarioMergeSettings::set_particle_system_merge_flag****Prototype:**

```
int set_particle_system_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.75.42 function `diguyScenarioMergeSettings::get_particle_system_merge_flag`

Prototype:

```
diguyScenarioMergeFlag get_particle_system_merge_flag();
```

Returns:

most recent value set by `set_particle_system_merge_flag()`

Callable From:

◇ C++
◇ Script

5.75.43 function

diguyScenarioMergeSettings::set_path_shape_merge_flag

Prototype:

```
int set_path_shape_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.44 function

diguyScenarioMergeSettings::get_path_shape_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_path_shape_merge_flag();
```

Returns:

most recent value set by [set_path_shape_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.45 function

diguyScenarioMergeSettings::set_region_merge_flag

Prototype:

```
int set_region_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.46 function

diguyScenarioMergeSettings::get_region_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_region_merge_flag();
```

Returns:

most recent value set by [set_region_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.47 function**diguyScenarioMergeSettings::set_scenario_settings_merge_flag****Prototype:**

```
int set_scenario_settings_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how scenario settings should be merged. Since there is only one set of scenario settings per scenario, the options for the `merge_flag` argument are limited.

Arguments:

`merge_flag` can be set only to
 DIGUY_MERGE_FLAG_DONT_MERGE or
 DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.48 function

diguyScenarioMergeSettings::get_scenario_settings_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_scenario_settings_merge_flag();
```

Returns:

most recent value set by [set_scenario_settings_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.49 function

diguyScenarioMergeSettings::set_scene_object_merge_flag

Prototype:

```
int set_scene_object_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.75.50 function `diguyScenarioMergeSettings::get_scene_object_merge_flag`

Prototype:

```
diguyScenarioMergeFlag get_scene_object_merge_flag();
```

Returns:

most recent value set by `set_scene_object_merge_flag()`

Callable From:

◇ C++
◇ Script

5.75.51 function diguyScenarioMergeSettings::set_scene_object_grid_merge_flag

Prototype:

```
int set_scene_object_grid_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.52 function diguyScenarioMergeSettings::get_scene_object_grid_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_scene_object_grid_merge_flag();
```

Returns:

most recent value set by [set_scene_object_grid_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.53 function

diguyScenarioMergeSettings::set_script_merge_flag

Prototype:

```
int set_script_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.54 function

diguyScenarioMergeSettings::get_script_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_script_merge_flag();
```

Returns:

most recent value set by [set_script_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.55 function

diguyScenarioMergeSettings::set_sensor_region_merge_flag

Prototype:

```
int set_sensor_region_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.56 function

diguyScenarioMergeSettings::get_sensor_region_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_sensor_region_merge_flag();
```

Returns:

most recent value set by [set_sensor_region_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.57 function

diguyScenarioMergeSettings::set_signal_merge_flag

Prototype:

```
int set_signal_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.58 function

diguyScenarioMergeSettings::get_signal_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_signal_merge_flag();
```

Returns:

most recent value set by [set_signal_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.59 function

diguyScenarioMergeSettings::set_sound_merge_flag

Prototype:

```
int set_sound_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.60 function

diguyScenarioMergeSettings::get_sound_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_sound_merge_flag();
```

Returns:

most recent value set by [set_sound_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.61 function

diguyScenarioMergeSettings::set_variable_merge_flag

Prototype:

```
int set_variable_merge_flag(diguyScenarioMergeFlag merge_flag);
```

Description:

This function sets how objects of the specified type should be merged.

Arguments:

`merge_flag` can be set to any merge flag value except
DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.62 function

diguyScenarioMergeSettings::get_variable_merge_flag

Prototype:

```
diguyScenarioMergeFlag get_variable_merge_flag();
```

Returns:

most recent value set by [set_variable_merge_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

Misc Settings

5.75.63 function diguyScenarioMergeSettings::set_reset_flag

Prototype:

```
int set_reset_flag(diguyScenarioMergeResetFlag reset_flag);
```

Description:

This function sets what scenario elements will be reset after the merge is complete.

Arguments:

`reset_flag` DIGUY_MERGE_RESET_NONE, (reset nothing) or DIGUY_MERGE_RESET_MERGED, (reset only merged elements) or DIGUY_MERGE_RESET_ALL (reset entire scenario); default is DIGUY_MERGE_RESET_ALL

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.64 function diguyScenarioMergeSettings::get_reset_flag

Prototype:

```
diguyScenarioMergeResetFlag get_reset_flag();
```

Returns:

most recent value set by [set_reset_flag\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.75.65 function `diguyScenarioMergeSettings::set_rename_prefix`

Prototype:

```
int set_rename_prefix(const char* rename_prefix);
```

Description:

This function sets the prefix that will be added to merged object names if they are renamed.

The default value is the empty string, or no rename prefix.

If neither a prefix nor suffix are specified, a prefix will be derived and used to resolve name conflicts.

Arguments:

`rename_prefix` string to be prepended to object names

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.66 function diguyScenarioMergeSettings::get_rename_prefix

Prototype:

```
const char* get_rename_prefix();
```

Returns:

prefix to be prepended to object names as set by [set_rename_prefix\(\)](#);
this pointer will never be NULL, but may be the empty string

Callable From:

- ◇ C++
- ◇ Script

5.75.67 function diguyScenarioMergeSettings::set_rename_suffix

Prototype:

```
int set_rename_suffix(const char* rename_suffix);
```

Description:

This function sets the suffix that will be added to merged object names if they are renamed.

The default value is the empty string, or no rename suffix.

If neither a prefix nor suffix are specified, a prefix will be derived and used to resolve name conflicts.

Arguments:

`rename_suffix` string to be appended to object names

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.68 function `diguyScenarioMergeSettings::get_rename_suffix`

Prototype:

```
const char* get_rename_suffix();
```

Returns:

prefix to be appended to object names as set by `set_rename_suffix()`;
this pointer will never be NULL, but may be the empty string

Callable From:

- ◇ C++
- ◇ Script

5.75.69 function diguyScenarioMergeSettings::set_character_group

Prototype:

```
int set_character_group(const char* group_name);
```

Description:

Sets the group characters in the merged scenario should be added to. This group will be created if it doesn't already exist.

By default merged characters will not be added to any group they are not already in the merged scenario.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.75.70 function diguyScenarioMergeSettings::get_character_group

Prototype:

```
const char* get_character_group();
```

Returns:

the most recent setting made by [set_character_group\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.76 diguySceneObject

diguySceneObject.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h | [diguy_constants.h](#)

Contents:

- ◆ [Shader Functions](#)
- ◆ [Deprecated Functions](#)
- ◆ [Experimental Functions](#)

Link against: **libdiguy**

5.76.1 class diguySceneObject

```
class BDI_DECLSPEC_diguy diguySceneObject
{
public:
    const char* get_name();
    int set_name(const char* name);
    long get_uid();
    const char* get_filename();
    const char* get_nodename();
    const char* get_alternate_filename();
    const char* get_fully_resolved_filename();
    int show();
    int hide();
    int add_vertex_to_user_defined(float x, float y, float z);
    int add_triangle_to_user_defined(int index0, int index1, int index2);
#ifdef CPLUSPLUS_ONLY
    int add_vertex_array_to_user_defined(const float* const vertices,
        int num_vertices, int skipBytes = 12);
    int add_triangle_array_to_user_defined(const int* const indices, int num_indices);
    int add_triangle_to_user_defined(const float* const pos0,
        const float* const pos1, const float* const pos2);
#endif
    int build_user_defined();
    int set_render_mode_shader(const char* shader_name,
        const char* render_mode = NULL);
    void reset_render_mode_shader_to_default(const char* render_mode = NULL);
    const char* get_render_mode_shader(const char* render_mode = NULL);
    diguyGraphicsShaderProgram* get_shader_program();
    diguyGraphicsShaderTechnique* get_shader_technique();
    int set_temporary_current_shader(const char* shader_name);
    void reset_current_shader_to_default();
    diguyGraphicsShaderInstance* get_shader_instance();
    void reset_shader_instance_variables_to_initial_values();
    void reset_shader_instance_variables_to_program_defaults();
    int set_switch_state(const char* switch_name,
        int state);
    int get_switch_state(const char* switch_name);
};
```

```

int set_dof_state(const char* dof_name,
float tx, float ty, float tz,
float rz, float rx, float ry);
int set_dof_state_flt(const char* dof_name,
float tx, float ty, float tz,
float rx, float ry, float rz);
int get_dof_state(const char* dof_name,
float* tx, float* ty, float* tz,
float* rz, float* rx, float* ry);
int get_dof_state_flt(const char* dof_name,
float* tx, float* ty, float* tz,
float* rx, float* ry, float* rz);
int add_switch_target(const char* switch_name,
int state,
float t);
int abort_all_switch_targets(const char* switch_name);
int add_dof_target(const char* dof_name,
float tx, float ty, float tz,
float rz, float rx, float ry,
float blend_tin, float blend_tout,
const char* blend_function_name = 0);
int add_dof_target_flt(const char* dof_name,
float tx, float ty, float tz,
float rx, float ry, float rz,
float blend_tin, float blend_tout,
const char* blend_function_name = 0);
int abort_all_dof_targets(const char* dof_name);
int set_shader_program(const char* shader_name);
int set_position(float x, float y, float z);
int get_position(float* x, float* y, float* z);
int set_orientation(float rz, float rx, float ry);
int get_orientation(float* rz, float* rx, float* ry);
int set_scale(float sx, float sy, float sz);
int get_scale(float* sx, float* sy, float* sz);
int set_filename(const char* file_name);
int get_is_sky_box();
int set_is_sky_box(int skybox);
static int set_texture_load_behavior(diguyGraphicsTextureLoadingBehavior mode);
static diguyGraphicsTextureLoadingBehavior get_texture_load_behavior();
#ifdef CPLUSPLUS_ONLY
int draw_visual();
#endif
int get_include_in_octtree();
int set_include_in_octtree(int val);
int set_include_in_physics_sim_scene(int include_in_simulation_terrain);
int get_include_in_physics_sim_scene();
int set_physics_collision_group(diguyPhysicsCollisionGroup collision_group);
diguyPhysicsCollisionGroup get_physics_collision_group();
int get_min(float* x, float* y, float* z);
int get_max(float* x, float* y, float* z);
void set_scene_graph_mask(unsigned int mask);
unsigned int get_scene_graph_mask();
int get_enabled();
int set_enabled(int enabled);
int get_visible();
int set_visible(int visible);

```

5.76.2 function diguySceneObject::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the scene object. This pointer will never be NULL.

Returns:

name of the scene object

Callable From:

- ◇ C++
- ◇ Script

5.76.3 function diguySceneObject::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.76.4 function diguySceneObject::get_uid

Prototype:

```
long get_uid();
```

Description:

All scene objects are assigned a unique identifier, or uid. This function returns this scene object's uid.

Note: unique identifiers will change between DI-Guy runs!

Returns:

unique identifier of object

Callable From:

- ◇ C++
- ◇ Script

5.76.5 function diguySceneObject::get_filename

Prototype:

```
const char* get_filename();
```

Description:

Returns the filename of the scene object. This pointer will never be NULL.

The returned filename may be relative to another directory, such as \$DIGUYgeometrysets. Call [get_fully_resolved_filename\(\)](#) to have DI-Guy search for the fully resolved filename.

Returns:

filename of the scene object

Callable From:

- ◇ C++
- ◇ Script

5.76.6 function diguySceneObject::get_nodename

Prototype:

```
const char* get_nodename();
```

Returns:

node within the scene object file; returned string is 0 length if contents of entire file should be used

Callable From:

- ◇ C++
- ◇ Script

5.76.7 function diguySceneObject::get_alternate_filename

Prototype:

```
const char* get_alternate_filename();
```

Returns:

alternate filename of the scene object

Callable From:

- ◇ C++
- ◇ Script

5.76.8 function diguySceneObject::get_fully_resolved_filename**Prototype:**

```
const char* get_fully_resolved_filename();
```

Description:

Returns the fully resolved filename of the scene object. Returns NULL if the file cannot be found.

Returns:

fully resolved filename of the scene object; NULL if not found

Callable From:

- ◇ C++
- ◇ Script

5.76.9 function diguySceneObject::show**Prototype:**

```
int show();
```

Description:

Attempts to load the file returned by [get_filename\(\)](#) and construct an object that can be rendered in the current graphics environment by the call [draw_visual\(\)](#).

Returns:

0 on success, -1 on failure

OpenGL Version:

Function will create OpenGL display lists.

Direct3D Version:

Function will have no effect. User must load and display object.

Callable From:

- ◇ C++
- ◇ Script

5.76.10 function diguySceneObject::hide**Prototype:**

```
int hide();
```

Description:

Hides the scene object. See [show\(\)](#) for notes on which platforms do not yet display scene objects.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.76.11 function diguySceneObject::add_vertex_to_user_defined**Prototype:**

```
int add_vertex_to_user_defined(float x, float y, float z);
```

Description:

Adds a vertex to a user defined scene object.

Returns:

0 on success, -1 on failure

5.76.12 function diguySceneObject::add_triangle_to_user_defined**Prototype:**

```
int add_triangle_to_user_defined(int index0, int index1, int index2);
```

Description:

Adds a triangle to a user defined scene object. Indices refer to vertices added with a `add_vertex_*` call.

Returns:

0 on success, -1 on failure

5.76.13 function diguySceneObject::add_vertex_array_to_user_defined

Prototype:

```
int add_vertex_array_to_user_defined(const float* const vertices,
                                     int num_vertices, int skipBytes = 12);
```

Description:

Adds an array of vertices to a user defined scene object.

Arguments:

skipBytes bytes between vertex position information

Returns:

0 on success, -1 on failure

5.76.14 function diguySceneObject::add_triangle_array_to_user_defined

Prototype:

```
int add_triangle_array_to_user_defined(const int* const indices, int num_indices);
```

Description:

Adds an array of triangles to a user defined scene object. The array should contain indices to vertices added with add_vertex_* calls.

Arguments:

num_indices Number of indices (3 indices per triangle).

Returns:

0 on success, -1 on failure

5.76.15 function diguySceneObject::add_triangle_to_user_defined

Prototype:

```
int add_triangle_to_user_defined(const float* const pos0,  
                                const float* const pos1, const float* const pos2);
```

Description:

Adds a triangle to a user defined object. Positions are expected to be float[3].

Returns:

0 on success, -1 on failure

5.76.16 function diguySceneObject::build_user_defined

Prototype:

```
int build_user_defined();
```

Description:

Builds the geometry of the user defined scene object from data passed via `add_vertex_` and `add_triangle_` calls.

Returns:

0 on success, -1 on failure

Shader Functions

5.76.17 function diguySceneObject::set_render_mode_shader

Prototype:

```
int set_render_mode_shader(const char* shader_name,  
                           const char* render_mode = NULL);
```

Description:

This function is essentially the same as the function [diguyCharacter::set_render_mode_shader \(\)](#), but applies to this scene object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.76.18 function diguySceneObject::reset_render_mode_shader_to_default

Prototype:

```
void reset_render_mode_shader_to_default(const char* render_mode = NULL);
```

Description:

This function is essentially the same as the function [diguyCharacter::reset_render_mode_shader_to_default\(\)](#), but applies to this scene object.

Callable From:

- ◇ C++
- ◇ Script

5.76.19 function diguySceneObject::get_render_mode_shader

Prototype:

```
const char* get_render_mode_shader(const char* render_mode = NULL);
```

Description:

This function is essentially the same as the function [diguyCharacter::get_render_mode_shader\(\)](#), but applies to this to this scene object.

Returns:

name of shader program; NULL if none

Callable From:

- ◇ C++
- ◇ Script

5.76.20 function diguySceneObject::get_shader_program**Prototype:**

```
diguyGraphicsShaderProgram* get_shader_program();
```

Description:

This function is essentially the same as the function `diguyCharacter::get_shader_program()`, but applies to this to this scene object. It's here for backwards compatibility DI-Guy 12 now uses shader techniques as wrappers for shader programs.

Returns:

object of type `diguyGraphicsShaderProgram`; NULL if none

Callable From:

- ◇ C++
- ◇ Script

5.76.21 function diguySceneObject::get_shader_technique**Prototype:**

```
diguyGraphicsShaderTechnique* get_shader_technique();
```

Description:

This function is essentially the same as the function `diguyCharacter::get_shader_technique()`, but applies to this to this scene object.

Returns:

object of type `diguyGraphicsShaderTechnique`; NULL if none

Callable From:

- ◇ C++
- ◇ Script

5.76.22 function

diguySceneObject::set_temporary_current_shader

Prototype:

```
int set_temporary_current_shader(const char* shader_name);
```

Description:

This function is essentially the same as the function `diguyCharacter::set_temporary_current_shader_program()`, but applies to this to this scene object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.76.23 function

diguySceneObject::reset_current_shader_to_default

Prototype:

```
void reset_current_shader_to_default();
```

Description:

This function is essentially the same as the function `diguyCharacter::reset_current_shader_to_default()`, but applies to this to this scene object.

Callable From:

- ◇ C++
- ◇ Script

5.76.24 function `diguySceneObject::get_shader_instance`

Prototype:

```
diguyGraphicsShaderInstance* get_shader_instance();
```

Description:

This function is essentially the same as the function `diguyCharacter::get_shader_instance()`, but applies to this to this scene object.

Note that the shader instance of a scene object can change over the lifetime of the scene object. Therefore the returned pointer should not be stored, but should be re-read each time it is needed.

Returns:

object of type `diguyGraphicsShaderInstance`; NULL if none

Callable From:

- ◇ C++
- ◇ Script

5.76.25 function `diguySceneObject::reset_shader_instance_variables_to_initial_va`

Prototype:

```
void reset_shader_instance_variables_to_initial_values();
```

Description:

This function is essentially the same as the function `diguyCharacter::reset_shader_instance_variables_to_initial_values()`, but applies to this scene object.

Callable From:

- ◇ C++
- ◇ Script

5.76.26 function**diguySceneObject::reset_shader_instance_variables_to_program****Prototype:**

```
void reset_shader_instance_variables_to_program_defaults();
```

Description:

This function is essentially the same as the function

[diguyCharacter::reset_shader_instance_variables_to_program_defaults\(\)](#)

but applies to this scene object.

Callable From:

◇ C++

◇ Script

Deprecated Functions

5.76.27 function diguySceneObject::set_switch_state**Prototype:**

```
int set_switch_state(const char* switch_name,  
                    int state);
```

Description:

Set the state of a switch belonging to this scene object. Currently only supported on OpenGL platform.

Arguments:

<code>switch_name</code>	the name of the switch element
<code>state</code>	the state, i.e. active sub-element, of the switch

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.76.28 function diguySceneObject::get_switch_state**Prototype:**

```
int get_switch_state(const char* switch_name);
```

Description:

Get the state of a switch belonging to this scene object. Currently only supported on OpenGL platform.

Arguments:

switch_name the name of the switch element

Returns:

The state of the switch, -1 if fails.

Callable From:

- ◇ C++
- ◇ Script

5.76.29 function diguySceneObject::set_dof_state**Prototype:**

```
int set_dof_state(const char* dof_name,
float tx, float ty, float tz,
float rz, float rx, float ry);
```

Description:

Set the state of a DOF element belonging to this scene object. Currently only supported on OpenGL platform.

Arguments:

dof_name	the name of the DOF element
tx, ty, tz	the x, y, and z displacements of the DOF
rz, rx, ry	the rotation of the DOF about the z, x, and y axes

The DOF will be rotated in ZXY order. For rotation in XYZ order, use [set_dof_state_flt\(\)](#).

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.76.30 function diguySceneObject::set_dof_state_flt

Prototype:

```
int set_dof_state_flt(const char* dof_name,
    float tx, float ty, float tz,
    float rx, float ry, float rz);
```

Description:

Set the state of a DOF element belonging to this scene object. Currently only supported on OpenGL platform.

Arguments:

dof_name	the name of the DOF element
tx, ty, tz	the x, y, and z displacements of the DOF
rz, rx, ry	the rotation of the DOF about the z, x, and y axes

The DOF will be rotated in XYZ order. For rotation in ZXY order, use `set_dof_state()`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.76.31 function diguySceneObject::get_dof_state

Prototype:

```
int get_dof_state(const char* dof_name,
                 float* tx, float* ty, float* tz,
                 float* rz, float* rx, float* ry);
```

Description:

Get the state of a DOF element belonging to this scene object. Currently only supported on OpenGL platform.

Arguments:

dof_name	the name of the DOF element
tx, ty, tz	the x, y, and z displacements of the DOF
rz, rx, ry	the rotation of the DOF about the z-, x-, and y-axes

The returned rotation values assume ZXY order. For XYZ order, use [get_dof_state_flt\(\)](#).

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.76.32 function diguySceneObject::get_dof_state_flt

Prototype:

```
int get_dof_state_flt(const char* dof_name,
    float* tx, float* ty, float* tz,
    float* rx, float* ry, float* rz);
```

Description:

Get the state of a DOF element belonging to this scene object. Currently only supported on OpenGL platform.

Arguments:

dof_name	the name of the DOF element
tx, ty, tz	the x, y, and z displacements of the DOF
rx, ry, rz	the rotation of the DOF about the z-, x-, and y-axes

The returned rotation values assume ZXY order. For XYZ order, use [get_dof_state_float\(\)](#).

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.76.33 function diguySceneObject::add_switch_target

Prototype:

```
int add_switch_target(const char* switch_name,
                     int state,
                     float t);
```

Description:

Set a trigger to change the state of a switch at a given time. Currently only supported on OpenGL platform.

Arguments:

<code>switch_name</code>	the name of the switch to be triggered
<code>state</code>	the state the switch should be in after time t
<code>t</code>	the time at which the switch should switch

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.76.34 function diguySceneObject::abort_all_switch_targets**Prototype:**

```
int abort_all_switch_targets(const char* switch_name);
```

Description:

Abort all the targets states for a given switch which have been set either using the DI-Guy function, [add_dof_target \(\)](#) or using a behavior file. Currently only supported on OpenGL platform.

Arguments:

`switch_name` the name of the switch whose targets are to be aborted

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
 ◇ Script

5.76.35 function diguySceneObject::add_dof_target

Prototype:

```
int add_dof_target(const char* dof_name,
                  float tx, float ty, float tz,
                  float rz, float rx, float ry,
                  float blend_tin, float blend_tout,
                  const char* blend_function_name = 0);
```

Description:

Set a trigger to change the state of a DOF at a given time. Currently only supported on OpenGL platform.

Arguments:

dof_name	the name of the DOF element
tx, ty, tz	the x, y, and z displacements of the DOF
rz, rx, ry	the rotation of the DOF about the z-, x-, and y-axes
blend_tin	

	the time at which the DOF should begin changing states
<code>blend_tout</code>	the time at which the DOF should stop changing states
<code>blend_function_name</code>	the type of interpolation to be used between the current and desired DOF states

Currently, valid function names are "Linear", "HalfSine", and "QuarterSine".

The DOF will be rotated in ZXY order. For rotation in XYZ order, use `set_dof_state_flt()`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.76.36 function `diguySceneObject::add_dof_target_flt`

Prototype:

```
int add_dof_target_flt(const char* dof_name,
    float tx, float ty, float tz,
    float rx, float ry, float rz,
    float blend_tin, float blend_tout,
    const char* blend_function_name = 0);
```

Description:

Set a trigger to change the state of a DOF at a given time. Currently only supported on OpenGL platform.

Arguments:

<code>dof_name</code>	the name of the DOF element
<code>tx, ty, tz</code>	the x, y, and z displacements of the DOF
<code>rx, ry, rz</code>	the rotation of the DOF about the z-, x-, and y-axes The DOF will be rotated in XYZ order. For rotation in ZXY order, use

	<code>add_dof_target()</code> .
<code>blend_tin</code>	the time at which the DOF should begin changing states
<code>blend_tout</code>	the time at which the DOF should stop changing states
<code>blend_function_name</code>	the type of interpolation to be used between the current and desired DOF states

Currently, valid function names are "Linear", "HalfSine", and "QuarterSine".

The DOF will be rotated in XYZ order. For rotation in ZXY order, use `set_dof_state_flt()`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.76.37 function `diguySceneObject::abort_all_dof_targets`

Prototype:

```
int abort_all_dof_targets(const char* dof_name);
```

Description:

Abort all the targets states for a given DOF which have been set either using the DI-Guy function, `add_dof_target()` or using a behavior file. Currently only supported on OpenGL platform.

Arguments:

`dof_name` the name of the DOF whose targets are to be aborted

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.76.38 function diguySceneObject::set_shader_program

Prototype:

```
int set_shader_program(const char* shader_name);
```

Deprecated as of 10.5.2. Use `set_shader_program_name( )` instead.

Experimental Functions

The following functions are experimental and may change or disappear with no warning.

5.76.39 function diguySceneObject::set_position

Prototype:

```
int set_position(float x, float y, float z);
```

Description:

This function allows the a master flight file xrefs with large offsets from the origin to be untranslated. The offset can then be added to the XYZ position of the Scene Object. This is sometimes necessary to overcome far position rendering issues.

5.76.40 function diguySceneObject::get_position**Prototype:**

```
int get_position(float* x, float* y, float* z);
```

5.76.41 function diguySceneObject::set_orientation**Prototype:**

```
int set_orientation(float rz, float rx, float ry);
```


5.76.42 function diguySceneObject::get_orientation**Prototype:**

```
int get_orientation(float* rz, float* rx, float* ry);
```

5.76.43 function diguySceneObject::set_scale**Prototype:**

```
int set_scale(float sx, float sy, float sz);
```

5.76.44 function diguySceneObject::get_scale**Prototype:**

```
int get_scale(float* sx, float* sy, float* sz);
```

5.76.45 function diguySceneObject::set_filename**Prototype:**

```
int set_filename(const char* file_name);
```

5.76.46 function diguySceneObject::get_is_sky_box**Prototype:**

```
int get_is_sky_box();
```

5.76.47 function diguySceneObject::set_is_sky_box**Prototype:**

```
int set_is_sky_box(int skybox);
```

5.76.48 function diguySceneObject::set_texture_load_behavior**Prototype:**

```
static int set_texture_load_behavior(diguyGraphicsTextureLoadingBehavior mode);
```

5.76.49 function diguySceneObject::get_texture_load_behavior**Prototype:**

```
static diguyGraphicsTextureLoadingBehavior get_texture_load_behavior();
```

5.76.50 function diguySceneObject::draw_visual**Prototype:**

```
int draw_visual();
```

Description:

Draws the scene object.

Currently OpenGL only.

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.76.51 function diguySceneObject::get_include_in_octtree**Prototype:**

```
int get_include_in_octtree();
```

5.76.52 function diguySceneObject::set_include_in_octtree**Prototype:**

```
int set_include_in_octtree(int val);
```

**5.76.53 function
diguySceneObject::set_include_in_physics_sim_scene****Prototype:**

```
int set_include_in_physics_sim_scene(int include_in_simulation_terrain);
```

5.76.54 function `diguySceneObject::get_include_in_physics_sim_scene`

Prototype:

```
int get_include_in_physics_sim_scene();
```

5.76.55 function `diguySceneObject::set_physics_collision_group`

Prototype:

```
int set_physics_collision_group(diguyPhysicsCollisionGroup collision_group);
```

5.76.56 function diguySceneObject::get_physics_collision_group

Prototype:

```
diguyPhysicsCollisionGroup get_physics_collision_group();
```

5.76.57 function diguySceneObject::get_min

Prototype:

```
int get_min(float* x, float* y, float* z);
```


5.76.58 function diguySceneObject::get_max**Prototype:**

```
int get_max(float* x, float* y, float* z);
```

5.76.59 function diguySceneObject::set_scene_graph_mask**Prototype:**

```
void set_scene_graph_mask(unsigned int mask);
```

5.76.60 function diguySceneObject::get_enabled**Prototype:**

```
int get_enabled();
```

5.76.61 function diguySceneObject::set_enabled**Prototype:**

```
int set_enabled(int enabled);
```

5.76.62 function diguySceneObject::get_visible**Prototype:**

```
int get_visible();
```

5.76.63 function diguySceneObject::set_visible**Prototype:**

```
int set_visible(int visible);
```

5.77 diguySensorRegion

diguySensorRegion.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [diguy_typedefs.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Autosense Functions](#)
- ◆ [Callback Functions](#)
- ◆ [Event Handler Functions](#)

Link against: **libdiguy**

5.77.1 class diguySensorRegion

```
class BDI_DECLSPEC_diguy diguySensorRegion
{
public:
    const char* get_name();
    int set_name(const char* name);
    const char* get_type_name();
    int contains_point(float tx, float ty, float tz);
    int contains_character(const char* character_name);
    int contains_character_by_type(const char* character_type);
    int contains_character_by_type_and_appearance(const char* character_type,
        const char* appearance);
    int contains_character_from_group(const char* group_name);
    int contains_character_link(const char* character_name,
        const char* link_name,
        float offset_tx, float offset_ty, float offset_tz);
    int add_characters_in_region_to_group(diguyCharacterGroup* group);
    int add_candidate_characters_in_region_to_group(diguyCharacterGroup* candidates_group,
        diguyCharacterGroup* group);
    int set_autosense_candidates_group(diguyCharacterGroup* autosense_candidates_group);
    diguyCharacterGroup* get_autosense_candidates_group();
    int set_autosense_members_group(diguyCharacterGroup* autosense_members_group);
    diguyCharacterGroup* get_autosense_members_group();
    diguyCharacter* get_autosense_callback_candidate_entered();
    diguyCharacter* get_autosense_callback_candidate_left();
    enum {
        CALLBACK_ID_CREATE = 1,
        CALLBACK_ID_DESTROY,
        CALLBACK_ID_DETONATION,
        CALLBACK_ID_AUTOSENSE_CHARACTER_ENTERED,
        CALLBACK_ID_AUTOSENSE_CHARACTER_LEFT
    };
#ifdef CPLUSPLUS_ONLY
    int add_callback(int callback_id,
        diguySensorRegionCallback* callback,
        void* callback_params = 0,
        void* callback_user_data = 0);
#endif
};
```

```

int remove_callback(int callback_id,
    diguySensorRegionCallback* callback);
int remove_callback_with_user_data(int callback_id,
    void* callback_user_data);
#endif
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int remove_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int map_event_handler_to_callback_id(int callback_id,
    const char* handler_name);
int unmap_event_handler_from_callback_id(int callback_id,
    const char* handler_name,
    int unmap_all_matches = 0);

```

5.77.2 function diguySensorRegion::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the object

Callable From:

- ◇ C++
- ◇ Script

5.77.3 function diguySensorRegion::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.77.4 function diguySensorRegion::get_type_name

Prototype:

```
const char* get_type_name();
```

Description:

Returns the type name of the object. This pointer will never be NULL.

Returns:

type name of the object

Callable From:

- ◇ C++
- ◇ Script

5.77.5 function diguySensorRegion::contains_point

Prototype:

```
int contains_point(float tx, float ty, float tz);
```

Description:

This function is checks whether the passed x, y, z position is in the sensor region.

Arguments:

tx, ty, tz point to check

Returns:

1 if the passed point is in the sensor region; 0 otherwise.

5.77.6 function diguySensorRegion::contains_character

Prototype:

```
int contains_character(const char* character_name);
```

Description:

This function checks to see whether the passed character is within this sensor region. The actual point that is checked is the point returned by [diguyCharacter::get_position\(\)](#).

Note that this point is typically by the character's feet, so it is possible for the upper half of a character to pass through a sensor region without this function detecting this.

To check the upper portion of a character, use [contains_character_link\(\)](#) instead, and specify "cervical" or "back" as the link.

Arguments:

character character to check

Returns:

1 if the passed character is in the sensor region; 0 otherwise.

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.77.7 function diguySensorRegion::contains_character_by_type

Prototype:

```
int contains_character_by_type(const char* character_type);
```

Description:

This function checks to see whether a character with the specified type is within this sensor region. The actual point that is checked is the point returned by [diguyCharacter::get_position\(\)](#).

Note that this point is typically by the character's feet, so it is possible for the upper half of a character to pass through a sensor region without this function detecting this.

Arguments:

character_type character type to check

Returns:

1 if a character with the passed type is in the sensor region; 0 otherwise.

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.77.8 function `diguySensorRegion::contains_character_by_type_and_appearance`

Prototype:

```
int contains_character_by_type_and_appearance(const char* character_type,
const char* appearance);
```

Description:

This function checks to see whether a character with the specified type and appearance is within this sensor region. The actual point that is checked is the point returned by `diguyCharacter::get_position()`.

Note that this point is typically by the character's feet, so it is possible for the upper half of a character to pass through a sensor region without this function detecting this.

Arguments:

<code>character_type</code>	character type to check
<code>appearance</code>	appearance to check

Returns:

1 if a character with the passed type is in the sensor region; 0 otherwise.

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.77.9 function diguySensorRegion::contains_character_from_group

Prototype:

```
int contains_character_from_group(const char* group_name);
```

Description:

This function checks to see whether a character from the specified group is within this sensor region. The actual point that is checked is the point returned by [diguyCharacter::get_position\(\)](#).

Note that this point is typically by the character's feet, so it is possible for the upper half of a character to pass through a sensor region without this function detecting this.

Arguments:

group_name group of characters to check

Returns:

1 if a character from the group is in the sensor region; 0 otherwise.

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.77.10 function diguySensorRegion::contains_character_link

Prototype:

```
int contains_character_link(const char* character_name,
                           const char* link_name,
                           float offset_tx, float offset_ty, float offset_tz);
```

Description:

This function is similar to [contains_character\(\)](#), but rather than checking whether the character's ideal position is in the sensor region, this function checks whether the specified link is in the sensor region.

Arguments:

<code>character_name</code>	name of the character
<code>link_name</code>	name of the link
<code>offset_tx,</code> <code>offset_ty,</code> <code>offset_tz</code>	offset in link

Returns:

1 if the passed character link is in the sensor region; 0 otherwise.

5.77.11 function diguySensorRegion::add_characters_in_region_to_group

Prototype:

```
int add_characters_in_region_to_group(diguyCharacterGroup* group);
```

Description:

This function checks all characters in the scenario for whether they are in this sensor region. Each character that is in the sensor region is put into the passed character group. Note that the group is first cleared of all members.

A subset of characters can be checked by calling
[add_candidate_characters_in_region_to_group\(\)](#).

Arguments:

group	group to which characters in sensor region should be added

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.77.12 function diguySensorRegion::add_candidate_characters_in_region_to_group

Prototype:

```
int add_candidate_characters_in_region_to_group(diguyCharacterGroup* candidates_group,
diguyCharacterGroup* group);
```

Description:

This function checks all characters in the candidates_group for whether they are in this sensor region. Each character that is in the sensor region is put into the passed character group. Note that the group is first cleared of all members.

Arguments:

candidates_group	group whose members should be checked
group	group to which characters in sensor region should be added

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

◇ Script

Autosense Functions

The autosense functions make it easy to add callbacks and event logic for when specific characters enter or leave a sensor region.

In each scenario update, the characters in the sensor region's autosense candidates group are checked for whether they are inside the sensor region. Candidate characters that are inside the sensor region are put into the autosense members group. Candidates that are not in the sensor region are removed from the members group.

The members group is always a subset of the candidates group. Characters should not be manually added or removed from the members group.

When a candidate character enters the sensor region for the first time, callbacks and events with id `CALLBACK_ID_AUTOSENSE_CHARACTER_ENTERED` are called.

When a candidate character leaves the sensor region, callbacks and events with id `CALLBACK_ID_AUTOSENSE_CHARACTER_LEFT` are called.

5.77.13 function

diguySensorRegion::set_autosense_candidates_group

Prototype:

```
int set_autosense_candidates_group(diguyCharacterGroup* autosense_candidates_group),
```

Description:

This function sets the group that will be used as autosense candidates.

The passed group must be created by the DI-Guy Scenario UI or via a call to [diguyScenario::create_group\(\)](#).

The members of the candidates group should be set using the DI-Guy Scenario UI or via calls to [diguyCharacterGroup::add_member\(\)](#).

Arguments:

<code>autosense_candidates_group</code>	group of characters to use as candidates

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.77.14 function diguySensorRegion::get_autosense_candidates_group

Prototype:

```
diguyCharacterGroup* get_autosense_candidates_group();
```

Returns:

the most recent setting of [set_autosense_candidates_group\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.77.15 function diguySensorRegion::set_autosense_members_group

Prototype:

```
int set_autosense_members_group(diguyCharacterGroup* autosense_members_group);
```

Description:

This function sets the group that will be used to hold autosense candidate characters that are in this sensor region.

The passed group must be created by the DI-Guy Scenario UI or via a call to [diguyScenario::create_group\(\)](#).

The members group is always a subset of the candidates group. Characters should not be manually added or removed from the members group.

Arguments:

`autosense_members_group` group that holds which candidate characters are in this sensor region

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.77.16 function diguySensorRegion::get_autosense_members_group

Prototype:

```
diguyCharacterGroup* get_autosense_members_group();
```

Returns:

the most recent setting of `set_autosense_members_group()`

Callable From:

- ◇ C++
- ◇ Script

5.77.17 function diguySensorRegion::get_autosense_callback_candidate_entered

Prototype:

```
diguyCharacter* get_autosense_callback_candidate_entered();
```

Description:

This function returns the character of the autosense candidates group that has entered the sensor region, triggering a CALLBACK_ID_AUTOSENSE_CHARACTER_ENTERED callback. This function will return NULL when not in the above callback.

Returns:

autosense candidate that has entered this region; NULL if not in a CALLBACK_ID_AUTOSENSE_CHARACTER_ENTERED callback

Callable From:

- ◇ C++
- ◇ Script

5.77.18 function diguySensorRegion::get_autosense_callback_candidate_left

Prototype:

```
diguyCharacter* get_autosense_callback_candidate_left();
```

Description:

This function returns the character of the autosense candidates group that has left the sensor region, triggering a CALLBACK_ID_AUTOSENSE_CHARACTER_LEFT callback. This function will return NULL when not in the above callback.

Returns:

autosense candidate that has left this region; NULL if not in a CALLBACK_ID_AUTOSENSE_CHARACTER_LEFT callback

Callable From:

- ◇ C++

◇ Script

Callback Functions

5.77.19 enumeration

```
enum {  
    CALLBACK_ID_CREATE = 1,  
    CALLBACK_ID_DESTROY,  
    CALLBACK_ID_DETONATION,  
    CALLBACK_ID_AUTOSENSE_CHARACTER_ENTERED,  
    CALLBACK_ID_AUTOSENSE_CHARACTER_LEFT  
};
```

Description:

This is an enumeration of the different callbacks that can be registered with [add_callback\(\)](#) and [add_callback_script\(\)](#).

Usable From:

◇ C++
◇ Script

5.77.20 function diguySensorRegion::add_callback

Prototype:

```
int add_callback(int callback_id,
                diguySensorRegionCallback* callback,
                void* callback_params = 0,
                void* callback_user_data = 0);
```

Description:

This function adds a user callback. Callbacks can be removed with [remove_callback\(\)](#) or [remove_callback_with_user_data\(\)](#).

Arguments:

callback	pointer to function with prototype diguySensorRegionCallback (typedefed above)
callback_id	integer id of when this callback is to be called
callback_params	not currently used; pass NULL
callback_user_data	pointer for user's own use; DI-Guy will do nothing to the contents of this pointer beyond passing it back when the callback is invoked

callback_id should be one of the following values:

CALLBACK_ID_CREATE

This callback will be called when a new sensor region is created.

Note that this callback can only be added by calling [diguyScenario::add_default_sensor_region_callback\(\)](#); adding it with [diguySensorRegion::add_callback\(\)](#) will have no effect, as by that time the sensor region has already been created.

CALLBACK_ID_DESTROY

This callback will be called when a sensor region is destroyed.

CALLBACK_ID_DETONATION

This callback will be called when a detonation occurs within the sensor region.

CALLBACK_ID_AUTOSENSE_CHARACTER_ENTERED

This callback will be called when an autosense candidate character first enters the sensor region.

CALLBACK_ID_AUTOSENSE_CHARACTER_LEFT

This callback will be called when an autosense candidate character first leaves the sensor region.

Callbacks return a value of type `diguyCallbackReturn`, which will be `DIGUY_CALLBACK_STOP` or `DIGUY_CALLBACK_CONTINUE`. If the callback returns `DIGUY_CALLBACK_STOP`, the default handler of the function will not be called; the callback is asserting that it has done everything necessary for the function call. If the callback returns `DIGUY_CALLBACK_CONTINUE`, the default handler for the function will be called after the callback.

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.77.21 function `diguySensorRegion::remove_callback`

Prototype:

```
int remove_callback(int callback_id,
    diguySensorRegionCallback* callback);
```

Description:

This function removes a user callback. All callbacks matching the specified `callback_id` and callback function will be removed.

Arguments:

<code>callback_id</code>	integer id of when this callback is to be called
<code>callback</code>	pointer to function with prototype <code>diguySensorRegionCallback</code> (typedefed above)

Returns:

0 on success, -1 on failure

5.77.22 function `diguySensorRegion::remove_callback_with_user_data`

Prototype:

```
int remove_callback_with_user_data(int callback_id,
    void* callback_user_data);
```

Description:

This function removes a user callback. All callbacks matching the specified `callback_id` and `callback_user_data` pointer will be removed.

Arguments:

<code>callback_id</code>	integer id of when this callback is to be called
<code>callback_user_data</code>	pointer for user's own use

Returns:

0 on success, -1 on failure

5.77.23 function `diguySensorRegion::add_callback_script`

Prototype:

```
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function adds a user callback script. Callback scripts can be removed with [remove_callback_script\(\)](#).

See [diguyCharacter::add_callback_script\(\)](#) for an example of use.

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback to be added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

lua specific:

When the script is called, the object for which it is being called will be in the `callback_object` global.

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.77.24 function `diguySensorRegion::remove_callback_script`

Prototype:

```
int remove_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function removes a user callback script previously added with [add_callback_script\(\)](#).

See [diguyCharacter::remove_callback_script\(\)](#) for an example of use.

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback previously added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script`, all callback scripts whose ids match `callback_id` and whose types match `callback_script_type` will be removed.

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

Lua specific:

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Event Handler Functions

5.77.25 function**diguySensorRegion::map_event_handler_to_callback_id****Prototype:**

```
int map_event_handler_to_callback_id(int callback_id,
    const char* handler_name);
```

Description:

This function maps the event handler with the given name to a callback id. This mapping will be saved in the .dss file and restored when the .dss file is loaded.

Mappings can also be made via the DI-Guy Scenario UI.

The event handler is one of the following:

- ◇ a scene object callback function registered by
`diguyScenario::register_sensor_region_event_handler()`
or
`diguyScenario::register_sensor_region_event_handler_from_library()`
- ◇ a script registered by
`diguyScenario::register_sensor_region_event_handler_script()`
- ◇ a Script, Decision, or Library Function in the scenario whose "Event Type" is "Sensor Region"

Arguments:

`callback_id` integer id of callback

`handler_name` name of the event handler to map

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.77.26 function**diguySensorRegion::unmap_event_handler_from_callback_id****Prototype:**

```
int unmap_event_handler_from_callback_id(int callback_id,
    const char* handler_name,
    int unmap_all_matches = 0);
```

Description:

This function unmaps the event handler with the given name from a callback id.

Arguments:

callback_id	integer id of callback
handler_name	name of the event handler to map
unmap_all_matches	pass 0 to unmap only the first match, pass 1 to unmap all matches

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.78 diguySignal

diguySignal.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [diguy_typedefs.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Callback Functions](#)
- ◆ [Event Handler Functions](#)

Link against: **libdiguy**

5.78.1 class diguySignal

```
class BDI_DECLSPEC_diguy diguySignal
{
public:
    const char* get_name();
    int set_name(const char* name);
    const char* get_type_name();
    int reset();
    int trigger();
    int triggered();
    int triggered_at_least_n_times(int n);
    int signal_palette_show();
    int signal_palette_hide();
    int get_is_visible_in_signal_palette();
    const char* get_info_text();
    void set_info_text(const char* info_text);
    enum {
        CALLBACK_ID_PRE_TRIGGER = 1,
        CALLBACK_ID_POST_TRIGGER
    };
#ifdef CPLUSPLUS_ONLY
    int add_callback(int callback_id,
        diguySignalCallback* callback,
        void* callback_params = 0,
        void* callback_user_data = 0);
    int remove_callback(int callback_id,
        diguySignalCallback* callback);
    int remove_callback_with_user_data(int callback_id,
        void* callback_user_data);
    int add_callback_script(int callback_id,
        const char* callback_script,
        const char* callback_script_type = NULL);
    int remove_callback_script(int callback_id,
        const char* callback_script,
        const char* callback_script_type = NULL);
    int map_event_handler_to_callback_id(int callback_id,
        const char* handler_name);
    int unmap_event_handler_from_callback_id(int callback_id,
```

```
const char* handler_name,  
int unmap_all_matches = 0);
```

5.78.2 function diguySignal::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the signal

Callable From:

- ◇ C++
- ◇ Script

5.78.3 function diguySignal::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.78.4 function diguySignal::get_type_name

Prototype:

```
const char* get_type_name();
```

Description:

Returns the type name of the object. This pointer will never be NULL.

Returns:

type name of the object

Callable From:

- ◇ C++
- ◇ Script

5.78.5 function diguySignal::reset

Prototype:

```
int reset();
```

Description:

This function resets the signal. The number of times the signal has been triggered will be set to 0.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.78.6 function diguySignal::trigger

Prototype:

```
int trigger();
```

Description:

This function triggers the signal. If a sound is associated with the signal, it will be played. The number of times the signal has been triggered will be incremented by one.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.78.7 function diguySignal::triggered

Prototype:

```
int triggered();
```

Returns:

Returns the number of times this signal has been triggered since the scenario started or since the last call to `reset()`.

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.78.8 function diguySignal::triggered_at_least_n_times

Prototype:

```
int triggered_at_least_n_times(int n);
```

Returns:

Returns 1 if the signal has been triggered at least n times, else returns 0.

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.78.9 function diguySignal::signal_palette_show

Prototype:

```
int signal_palette_show();
```

Description:

This function causes the button for this signal to become visible in the DI-Guy Scenario Signal Palette.

This function is for DI-Guy Scenario only.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.78.10 function diguySignal::signal_palette_hide

Prototype:

```
int signal_palette_hide();
```

Description:

This function causes the button for this signal to become hidden in the DI-Guy Scenario Signal Palette.

This function is for DI-Guy Scenario only.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.78.11 function diguySignal::get_is_visible_in_signal_palette**Prototype:**

```
int get_is_visible_in_signal_palette();
```

Description:

This function returns a flag specifying whether the signal's button is visible in the DI-Guy Scenario Signal Palette.

This function is for DI-Guy Scenario only.

Returns:

1 if the signal's button is visible, 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.78.12 function diguySignal::get_info_text**Prototype:**

```
const char* get_info_text();
```

Description:

This function returns an optional signal description buffer.

Returns:

pointer to info text

Callable From:

- ◇ C++
- ◇ Script

5.78.13 function diguySignal::set_info_text

Prototype:

```
void set_info_text(const char* info_text);
```

Description:

This function sets an optional signal description buffer.

Arguments:

info_text text of info

Callable From:

- ◇ C++
- ◇ Script

Callback Functions

5.78.14 enumeration

```
enum {
    CALLBACK_ID_PRE_TRIGGER = 1,
    CALLBACK_ID_POST_TRIGGER
};
```

Description:

This is an enumeration of the different callbacks that can be registered with [add_callback\(\)](#) and [add_callback_script\(\)](#).

Usable From:

- ◇ C++
- ◇ Script

5.78.15 function diguySignal::add_callback

Prototype:

```
int add_callback(int callback_id,
                diguySignalCallback* callback,
                void* callback_params = 0,
                void* callback_user_data = 0);
```

Description:

This function adds a user callback.

Arguments:

callback	pointer to function with prototype diguySignalCallback (typedefed above)
callback_id	integer id of when this callback is to be called
callback_params	not currently used; pass NULL
callback_user_data	pointer for user's own use; DI-Guy will do nothing to the contents of this pointer beyond passing it back when the callback is invoked

callback_id should be one of the following values:

CALLBACK_ID_PRE_TRIGGER

This callback will be called whenever the `trigger()` function is called, whether internally by DI-Guy or explicitly by the user in C++ or in a script. It will be called before the default handler of the function.

There are no parameters for this callback.

CALLBACK_ID_POST_TRIGGER

This callback will be called whenever the `trigger()` function is called, whether internally by DI-Guy or explicitly by the user in C++ or in a script. It will be called after the default handler of the function.

There are no parameters for this callback.

Callback Return Values:

Callbacks return a value of type `diguyCallbackReturn`, which will be `DIGUY_CALLBACK_STOP` or `DIGUY_CALLBACK_CONTINUE`. If the callback returns `DIGUY_CALLBACK_STOP`, the default handler of the function will not be called; the callback is asserting that it has done everything necessary for the function call. If the callback returns `DIGUY_CALLBACK_CONTINUE`, the default handler for the function will

be called after the callback.

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.78.16 function diguySignal::remove_callback

Prototype:

```
int remove_callback(int callback_id,
    diguySignalCallback* callback);
```

Description:

This function removes a user callback. All callbacks matching the specified callback_id and callback function will be removed.

Arguments:

callback_id	integer id of when this callback is to be called
callback	pointer to function with prototype diguySignalCallback (typedefed above)

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.78.17 function diguySignal::remove_callback_with_user_data

Prototype:

```
int remove_callback_with_user_data(int callback_id,
    void* callback_user_data);
```

Description:

This function removes a user callback. All callbacks matching the specified callback_id and callback_user_data pointer will be removed.

Arguments:

callback_id	integer id of when this callback is to be called
callback_user_data	pointer for user's own use

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.78.18 function diguySignal::add_callback_script

Prototype:

```
int add_callback_script(int callback_id,  
    const char* callback_script,  
    const char* callback_script_type = NULL);
```

Description:

This function adds a user callback script. Callback scripts can be removed with [remove_callback_script\(\)](#).

See [diguyCharacter::add_callback_script\(\)](#) for an example of use.

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback to be added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

Lua specific:

When the script is called, the object for which it is being called will be in the `callback_object` global.

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.78.19 function `diguySignal::remove_callback_script`

Prototype:

```
int remove_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function removes a user callback script previously added with [add_callback_script\(\)](#).

See [diguyCharacter::remove_callback_script\(\)](#) for an example of use.

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback previously added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script`, all callback scripts whose ids match `callback_id` and whose types match `callback_script_type` will be removed.

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

Lua specific:

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Event Handler Functions

5.78.20 function**diguySignal::map_event_handler_to_callback_id****Prototype:**

```
int map_event_handler_to_callback_id(int callback_id,
    const char* handler_name);
```

Description:

This function maps the event handler with the given name to a callback id. This mapping will be saved in the .dss file and restored when the .dss file is loaded.

Mappings can also be made via the DI-Guy Scenario UI.

The event handler is one of the following:

- ◇ a signal callback function registered by
`diguyScenario::register_signal_event_handler()`
or
`diguyScenario::register_signal_event_handler_from_library()`
- ◇ a script registered by
`diguyScenario::register_signal_event_handler_script()`
- ◇ a Script, Decision, or Library Function in the scenario whose "Event Type" is "Signal"

Arguments:

`callback_id` integer id of callback
`handler_name` name of the event handler to map

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.78.21 function**diguySignal::unmap_event_handler_from_callback_id****Prototype:**

```
int unmap_event_handler_from_callback_id(int callback_id,
    const char* handler_name,
    int unmap_all_matches = 0);
```

Description:

This function unmaps the event handler with the given name from a callback id.

Arguments:

callback_id	integer id of callback
handler_name	name of the event handler to map
unmap_all_matches	pass 0 to unmap only the first match, pass 1 to unmap all matches

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.79 diguySound

diguySound.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

This class represents a sound file on disk.

All Function in this class are callable from:

- ◆ C++
- ◆ Script

Unless otherwise noted.

5.79.1 class diguySound

```
class BDI_DECLSPEC_diguy diguySound
{
public:
    const char* get_name();
    const char* get_type_name();
    int set_sound_filename(const char* sound_filename);
    const char* get_sound_filename();
    int set_phoneme_filename(const char* phoneme_filename);
    const char* get_phoneme_filename();
    float get_duration();
    int get_samples_per_second();
    int get_number_of_channels();
    int get_bit_depth();
```

5.79.2 function diguySound::get_name

Prototype:

```
const char* get_name();
```

Returns:

name of this object

5.79.3 function diguySound::get_type_name

Prototype:

```
const char* get_type_name();
```

Returns:

type name of the object; this value will never be NULL

5.79.4 function diguySound::set_sound_filename**Prototype:**

```
int set_sound_filename(const char* sound_filename);
```

Description:

This function sets the name of the file that contains the waveform data for the sound. The file should typically be located in the \$(DIGUY)/sfx or \$(DIGUY)*customsfx* directory.

Arguments:

`sound_filename` name of file containing waveform data

Returns:

0 on success, -1 on failure

5.79.5 function diguySound::get_sound_filename**Prototype:**

```
const char* get_sound_filename();
```

Returns:

name of file containing waveform data for sound as set by [set_sound_filename](#)(); will never be NULL, but may be the empty string ""

5.79.6 function diguySound::set_phoneme_filename

Prototype:

```
int set_phoneme_filename(const char* phoneme_filename);
```

Description:

This function sets the name of the file that contains the phoneme data for the sound. The file should typically be located in the same place as its waveform counterpart.

Phoneme files are optional.

Arguments:

`phoneme_filename` name of file containing phoneme data

Returns:

0 on success, -1 on failure

5.79.7 function diguySound::get_phoneme_filename

Prototype:

```
const char* get_phoneme_filename();
```

Returns:

name of file containing phoneme data as set by
[set_phoneme_filename\(\)](#)

5.79.8 function diguySound::get_duration

Prototype:

```
float get_duration();
```

Returns:

the length of the sound in seconds

5.79.9 function diguySound::get_samples_per_second

Prototype:

```
int get_samples_per_second();
```

Returns:

the number of samples per second

5.79.10 function diguySound::get_number_of_channels**Prototype:**

```
int get_number_of_channels();
```

Returns:

the number of sound channels either 1 (mono) or 2 (stereo)

5.79.11 function diguySound::get_bit_depth**Prototype:**

```
int get_bit_depth();
```

Returns:

the bit depth of a sound, either 8 or 16

5.80 diguySoundInstance

diguySoundInstance.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

5.80.1 class diguySoundInstance

```
class BDI_DECLSPEC_diguy diguySoundInstance
{
public:
    const char* get_name();
    const char* get_type_name();
    int play(double offset = 0.0);
    int get_is_playing();
    int get_is_stopping();
    int stop(float rampdown_time = 0.0f);
    int set_automatic_destroy_flag();
    void set_is_3d_sound_flag(int is_3d_sound_flag);
    int get_is_3d_sound_flag();
};
```

5.80.2 function diguySoundInstance::get_name

Prototype:

```
const char* get_name();
```

Returns:

name of the sound used by the sound instance.

Callable From:

- ◇ C++
- ◇ Script

5.80.3 function diguySoundInstance::get_type_name

Prototype:

```
const char* get_type_name();
```

Returns:

type name of the object; this value will never be NULL

Callable From:

- ◇ C++
- ◇ Script

5.80.4 function diguySoundInstance::play

Prototype:

```
int play(double offset = 0.0);
```

Description:

This function causes the sound to play immediately, starting from the specified time offset. The sound will continue until it has completed, or it is stopped.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.80.5 function diguySoundInstance::get_is_playing

Prototype:

```
int get_is_playing();
```

Returns:

1 if the sound is playing; 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.80.6 function diguySoundInstance::get_is_stopping

Prototype:

```
int get_is_stopping();
```

Returns:

1 if the sound is stopping; 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.80.7 function `diguySoundInstance::stop`

Prototype:

```
int stop(float rampdown_time = 0.0f);
```

Description:

This function causes this sound to stop. It does not destroy the sound instance. The sound instance can be played again if desired.

Arguments

`rampdown_time` how long to take fading out the sound

*b Returns:

0 on success, -1 on failure

Callable From:

◇ C++

◇ Script

5.80.8 function `diguySoundInstance::set_automatic_destroy_flag`

Prototype:

```
int set_automatic_destroy_flag();
```

Description:

This function causes the sound instance to be automatically deleted when the sound has completed. No further calls through the `diguySoundInstance` object can or should be made, so this function should only be called once all durations, reps, etc. have been set.

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

◇ Script

5.80.9 function diguySoundInstance::set_is_3d_sound_flag**Prototype:**

```
void set_is_3d_sound_flag(int is_3d_sound_flag);
```

Description:

This function sets whether the sound is "ambient" (it will be equally audible from everywhere within the world), or "3d" (it will move with the character).

Arguments

`is_3d_sound_flag` pass 1 to make the sound 3d; pass 0 to make the sound ambient

Callable From:

- ◇ C++
- ◇ Script

5.80.10 function diguySoundInstance::get_is_3d_sound_flag**Prototype:**

```
int get_is_3d_sound_flag();
```

Returns:

1 if the sound is 3d; 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.81 diguyVariable

diguyVariable.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [diguy_typedefs.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Callback Functions](#)
- ◆ [Event Handler Functions](#)

Link against: **libdiguy**

5.81.1 class diguyVariable

```
class BDI_DECLSPEC_diguy diguyVariable
{
public:
    const char* get_name();
    int set_name(const char* name);
    const char* get_type_name();
    diguyScenario* get_scenario();
    diguyCharacter* get_character();
    int copy(diguyVariable* rhs);
    int get_value_as_int();
    float get_value_as_float();
    const char* get_value_as_string();
    int set_value_as_int(int value);
    int set_value_as_float(float value);
    int set_value_as_string(const char* value);
    int increment_value_as_int(int increment_amount = 1);
    int decrement_value_as_int(int decrement_amount = 1);
    int equal_to_int(int value);
    int greater_than_int(int value);
    int less_than_int(int value);
    int equal_to_float(float value);
    int greater_than_float(float value);
    int less_than_float(float value);
    int equal_to_string(const char* value);
    int greater_than_string(const char* value);
    int less_than_string(const char* value);
    int get_initial_value_as_int();
    float get_initial_value_as_float();
    const char* get_initial_value_as_string();
    int set_initial_value_as_int(int initial_value);
    int set_initial_value_as_float(float initial_value);
    int set_initial_value_as_string(const char* initial_value);
    enum {
        CALLBACK_ID_VALUE_CHANGED = 1
    };
#ifdef CPLUSPLUS_ONLY
    int add_callback(int callback_id,
```

```

    diguyVariableCallback* callback,
    void* callback_params = 0,
    void* callback_user_data = 0);
int remove_callback(int callback_id,
    diguyVariableCallback* callback);
int remove_callback_with_user_data(int callback_id,
    void* callback_user_data);
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int remove_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int map_event_handler_to_callback_id(int callback_id,
    const char* handler_name);
int unmap_event_handler_from_callback_id(int callback_id,
    const char* handler_name,
    int unmap_all_matches = 0);

```

5.81.2 function diguyVariable::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the variable. This pointer will never be NULL.

Returns:

name of the variable

Callable From:

- ◇ C++
- ◇ Script

5.81.3 function diguyVariable::set_name

Prototype:

```
int set_name(const char* name);
```

Description:

This function sets the name of this object.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.81.4 function diguyVariable::get_type_name

Prototype:

```
const char* get_type_name();
```

Description:

Returns the type name of the object. This pointer will never be NULL.

Returns:

type name of the object

Callable From:

- ◇ C++
- ◇ Script

5.81.5 function diguyVariable::get_scenario

Prototype:

```
diguyScenario* get_scenario();
```

Description:

Returns the scenario that this variable belongs to. This pointer will never be NULL. Note that if this variable belongs to a character, a pointer to the scenario that the character belongs to will be returned.

Returns:

pointer of type [diguyScenario](#)

Callable From:

- ◇ C++
- ◇ Script

5.81.6 function diguyVariable::get_character

Prototype:

```
diguyCharacter* get_character();
```

Description:

Returns the character that this variable belongs to, if any. This pointer might be NULL if this variable is not a character variable.

Returns:

pointer of type [diguyCharacter](#); NULL if this variable is not a character variable

Callable From:

- ◇ C++
- ◇ Script

5.81.7 function diguyVariable::copy

Prototype:

```
int copy(diguyVariable* rhs);
```

Description:

Copies the contents of the passed variable.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.81.8 function diguyVariable::get_value_as_int

Prototype:

```
int get_value_as_int();
```

Description:

Returns the value of the variable as an integer.

Returns:

value of the variable

Callable From:

- ◇ C++
- ◇ Script

5.81.9 function diguyVariable::get_value_as_float

Prototype:

```
float get_value_as_float();
```

Description:

Returns the value of the variable as a floating point number.

Returns:

value of the variable

Callable From:

- ◇ C++
- ◇ Script

5.81.10 function diguyVariable::get_value_as_string

Prototype:

```
const char* get_value_as_string();
```

Description:

Returns the value of the variable as a string. This pointer will never be NULL.

Returns:

value of the variable

Callable From:

- ◇ C++
- ◇ Script

5.81.11 function diguyVariable::set_value_as_int**Prototype:**

```
int set_value_as_int(int value);
```

Description:

This function sets the value as an integer.

Arguments:

value	new value of variable

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.81.12 function diguyVariable::set_value_as_float**Prototype:**

```
int set_value_as_float(float value);
```

Description:

This function sets the value as a floating point number.

Arguments:

value	new value of variable

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

◇ Decision Bead

5.81.13 function diguyVariable::set_value_as_string

Prototype:

```
int set_value_as_string(const char* value);
```

Description:

This function sets the value as a string.

Arguments:

va lue	new value of variable

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

◇ Decision Bead

5.81.14 function diguyVariable::increment_value_as_int

Prototype:

```
int increment_value_as_int(int increment_amount = 1);
```

Description:

This function increments the value as an int.

Arguments:

`increment_amount` the value by which to increment

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

◇ Decision Bead

5.81.15 function diguyVariable::decrement_value_as_int

Prototype:

```
int decrement_value_as_int(int decrement_amount = 1);
```

Description:

This function decrements the value as an int.

Arguments:

`decrement_amount` the value by which to decrement

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

◇ Decision Bead

5.81.16 function diguyVariable::equal_to_int

Prototype:

```
int equal_to_int(int value);
```

Returns:

1 if the variable as an int is equal to the passed value; 0 if not

Arguments:

value the value to compare against

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

◇ Decision Bead

5.81.17 function diguyVariable::greater_than_int

Prototype:

```
int greater_than_int(int value);
```

Returns:

1 if the variable as an int is greater than the passed value; 0 if not

Arguments:

value the value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.81.18 function diguyVariable::less_than_int**Prototype:**

```
int less_than_int(int value);
```

Returns:

1 if the variable as an int is less than the passed value; 0 if not

Arguments:

value the value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.81.19 function diguyVariable::equal_to_float**Prototype:**

```
int equal_to_float(float value);
```

Returns:

1 if the variable as a float is equal to the passed value; 0 if not

Arguments:

value the value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.81.20 function diguyVariable::greater_than_float**Prototype:**

```
int greater_than_float(float value);
```

Returns:

1 if the variable as a float is greater than the passed value; 0 if not

Arguments:

value the value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.81.21 function diguyVariable::less_than_float**Prototype:**

```
int less_than_float(float value);
```

Returns:

1 if the variable as a float is less than the passed value; 0 if not

Arguments:

value the value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.81.22 function diguyVariable::equal_to_string**Prototype:**

```
int equal_to_string(const char* value);
```

Returns:

1 if the variable as a string is equal to the passed value; 0 if not

Arguments:

value the value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.81.23 function diguyVariable::greater_than_string**Prototype:**

```
int greater_than_string(const char* value);
```

Returns:

1 if the variable as a string is alphabetically greater than the passed value; 0 if not

Arguments:

value the value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.81.24 function diguyVariable::less_than_string**Prototype:**

```
int less_than_string(const char* value);
```

Returns:

1 if the variable as a string is alphabetically less than the passed value; 0 if not

Arguments:

value the value to compare against

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.81.25 function diguyVariable::get_initial_value_as_int**Prototype:**

```
int get_initial_value_as_int();
```

Description:

Returns the initial value of the variable as an integer.

Returns:

initial value of the variable

Callable From:

- ◇ C++
- ◇ Script

5.81.26 function diguyVariable::get_initial_value_as_float**Prototype:**

```
float get_initial_value_as_float();
```

Description:

Returns the initial value of the variable as a floating point number.

Returns:

initial value of the variable

Callable From:

- ◇ C++
- ◇ Script

5.81.27 function diguyVariable::get_initial_value_as_string**Prototype:**

```
const char* get_initial_value_as_string();
```

Description:

Returns the initial value of the variable as a string. This pointer will never be NULL.

Returns:

initial value of the variable

Callable From:

- ◇ C++
- ◇ Script

5.81.28 function diguyVariable::set_initial_value_as_int**Prototype:**

```
int set_initial_value_as_int(int initial_value);
```

Description:

This function sets the initial value as an integer. The current value will be set to the initial value when the scenario containing this variable is reset.

Arguments:

<code>initial</code>	new initial value of variable
<code>value</code>	

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.81.29 function diguyVariable::set_initial_value_as_float**Prototype:**

```
int set_initial_value_as_float(float initial_value);
```

Description:

This function sets the initial value as a floating point number. The current value will be set to the initial value when the scenario containing this variable is reset.

Arguments:

<code>initial</code>	new initial value of variable
<code>value</code>	

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.81.30 function diguyVariable::set_initial_value_as_string

Prototype:

```
int set_initial_value_as_string(const char* initial_value);
```

Description:

This function sets the initial value as a string. The current value will be set to the initial value when the scenario containing this variable is reset.

Arguments:

<code>initial</code>	new initial value of variable
<code>value</code>	

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

Callback Functions

5.81.31 enumeration

```
enum {  
    CALLBACK_ID_VALUE_CHANGED = 1  
};
```

Description:

This is an enumeration of the different callbacks that can be registered with [add_callback\(\)](#) and [add_callback_script\(\)](#).

Usable From:

◇ C++
◇ Script

5.81.32 function diguyVariable::add_callback

Prototype:

```
int add_callback(int callback_id,
                 diguyVariableCallback* callback,
                 void* callback_params = 0,
                 void* callback_user_data = 0);
```

Description:

This function adds a user callback.

Arguments:

callback	pointer to function with prototype diguyVariableCallback (typedefed above)
callback_id	integer id of when this callback is to be called
callback_params	not currently used; pass NULL
callback_user_data	pointer for user's own use; DI-Guy will do nothing to the contents of this pointer beyond passing it back when the callback is invoked

callback_id should be one of the following values:

CALLBACK_ID_VALUE_CHANGED

This callback will be called whenever the value of the variable changes.

There are no parameters for this callback.

Callback Return Values:

Callbacks return a value of type [diguyCallbackReturn](#), which will be [DIGUY_CALLBACK_STOP](#) or [DIGUY_CALLBACK_CONTINUE](#). If the callback returns [DIGUY_CALLBACK_STOP](#), the default handler of the function will not be called; the callback is asserting that it has done everything necessary for the function call. If the callback returns [DIGUY_CALLBACK_CONTINUE](#), the default handler for the function will be called after the callback.

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.81.33 function diguyVariable::remove_callback

Prototype:

```
int remove_callback(int callback_id,  
    diguyVariableCallback* callback);
```

Description:

This function removes a user callback. All callbacks matching the specified `callback_id` and callback function will be removed.

Arguments:

<code>callback_id</code>	integer id of when this callback is to be called
<code>callback</code>	pointer to function with prototype <code>diguyVariableCallback</code> (typedefed above)

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.81.34 function diguyVariable::remove_callback_with_user_data**Prototype:**

```
int remove_callback_with_user_data(int callback_id,
    void* callback_user_data);
```

Description:

This function removes a user callback. All callbacks matching the specified `callback_id` and `callback_user_data` pointer will be removed.

Arguments:

<code>callback_id</code>	integer id of when this callback is to be called
<code>callback_user_data</code>	pointer for user's own use

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.81.35 function diguyVariable::add_callback_script**Prototype:**

```
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function adds a user callback script. Callback scripts can be removed with [remove_callback_script\(\)](#).

See [diguyCharacter::add_callback_script\(\)](#) for an example of use.

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback to be added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

Lua specific:

When the script is called, the object for which it is being called will be in the `callback_object` global.

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.81.36 function `diguyVariable::remove_callback_script`

Prototype:

```
int remove_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function removes a user callback script previously added with [add_callback_script\(\)](#).

See [diguyCharacter::remove_callback_script\(\)](#) for an example of use.

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback previously added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script`, all callback scripts whose ids match `callback_id` and whose types match `callback_script_type` will be removed.

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

Lua specific:

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Event Handler Functions

5.81.37 function**diguyVariable::map_event_handler_to_callback_id****Prototype:**

```
int map_event_handler_to_callback_id(int callback_id,
    const char* handler_name);
```

Description:

This function maps the event handler with the given name to a callback id. This mapping will be saved in the .dss file and restored when the .dss file is loaded.

Mappings can also be made via the DI-Guy Scenario UI.

The event handler is one of the following:

- ◇ a variable callback function registered by
`diguyScenario::register_variable_event_handler()`
or
`diguyScenario::register_variable_event_handler_from_library()`
- ◇ a script registered by
`diguyScenario::register_variable_event_handler_script()`
- ◇ a Script, Decision, or Library Function in the scenario whose "Event Type" is "Variable"

Arguments:

`callback_id` integer id of callback

`handler_name` name of the event handler to map

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.81.38 function**diguyVariable::unmap_event_handler_from_callback_id****Prototype:**

```
int unmap_event_handler_from_callback_id(int callback_id,
    const char* handler_name,
    int unmap_all_matches = 0);
```

Description:

This function unmaps the event handler with the given name from a callback id.

Arguments:

callback_id	integer id of callback
handler_name	name of the event handler to map
unmap_all_matches	pass 0 to unmap only the first match, pass 1 to unmap all matches

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.82 diguyVehicleController

diguyVehicleController.h

DI-Guy SDK Version 12.5.2

Includes: [declspec_diguy.h](#) | [diguy_constants.h](#)

Contents:

- ◆ [Character Functions](#)
- ◆ [Speed Functions](#)
- ◆ [Steering Functions](#)

5.82.1 class diguyVehicleController

```
class BDI_DECLSPEC_diguy diguyVehicleController
{
public:
    diguyCharacter* get_vehicle_character();
    void set_speed(float speed,
        diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
    float get_speed(diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
    void set_desired_speed_with_acceleration(float desired_speed,
        float acceleration,
        diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
    void set_desired_speed_with_acceleration_time(float desired_speed,
        float acceleration_time, // seconds
        diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
    float get_desired_speed(diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
    void set_default_acceleration(float acceleration,
        diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
    float get_default_acceleration(diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
    void set_default_deceleration(float deceleration,
        diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
    float get_default_deceleration(diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
    void set_vehicle_simulation_mode(diguyVehicleSimulationMode vsim_mode);
    diguyVehicleSimulationMode get_vehicle_simulation_mode();
    void set_front_wheels_steer_angle(float angle);
    float get_front_wheels_steer_angle();
    void set_desired_front_wheels_steer_angle(float desired_angle,
        float angle_change_rate = DIGUY_DEFAULT_FLOAT);
    float get_desired_front_wheels_steer_angle();
```

Character Functions

5.82.2 function diguyVehicleController::get_vehicle_character

Prototype:

```
diguyCharacter* get_vehicle_character();
```

Speed Functions

These functions should only be used in free action mode.

5.82.3 function diguyVehicleController::set_speed

Prototype:

```
void set_speed(float speed,
               diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
```

Description:

This function sets the vehicle's speed. The speed becomes effective immediately; no acceleration or deceleration is applied.

Arguments:

speed	new speed of vehicle
units	units of speed

Callable From:

- ◇ C++
- ◇ Script

5.82.4 function diguyVehicleController::get_speed

Prototype:

```
float get_speed(diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
```

5.82.5 function diguyVehicleController::set_desired_speed_with_acceleration

Prototype:

```
void set_desired_speed_with_acceleration(float desired_speed,  
    float acceleration,  
    diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
```

5.82.6 function**diguyVehicleController::set_desired_speed_with_acceleration_time****Prototype:**

```
void set_desired_speed_with_acceleration_time(float desired_speed,  
float acceleration_time,    // seconds  
diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
```

5.82.7 function diguyVehicleController::get_desired_speed**Prototype:**

```
float get_desired_speed(diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
```

5.82.8 function diguyVehicleController::set_default_acceleration**Prototype:**

```
void set_default_acceleration(float acceleration,  
    diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
```

5.82.9 function diguyVehicleController::get_default_acceleration**Prototype:**

```
float get_default_acceleration(diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
```

5.82.10 function `diguyVehicleController::set_default_deceleration`

Prototype:

```
void set_default_deceleration(float deceleration,  
    diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
```

5.82.11 function `diguyVehicleController::get_default_deceleration`

Prototype:

```
float get_default_deceleration(diguySpeedUnits units = DIGUY_SPEED_UNITS_METERS_PER_SECOND);
```

Steering Functions

These functions should only be used in free action mode.

5.82.12 function `diguyVehicleController::set_vehicle_simulation_mode`

Prototype:

```
void set_vehicle_simulation_mode(diguyVehicleSimulationMode vsim_mode);
```


5.82.13 function

diguyVehicleController::get_vehicle_simulation_mode

Prototype:

```
diguyVehicleSimulationMode get_vehicle_simulation_mode();
```

5.82.14 function

diguyVehicleController::set_front_wheels_steer_angle

Prototype:

```
void set_front_wheels_steer_angle(float angle);
```

5.82.15 function**diguyVehicleController::get_front_wheels_steer_angle****Prototype:**

```
float get_front_wheels_steer_angle();
```

5.82.16 function**diguyVehicleController::set_desired_front_wheels_steer_angle****Prototype:**

```
void set_desired_front_wheels_steer_angle(float desired_angle,  
float angle_change_rate = DIGUY_DEFAULT_FLOAT);
```

5.82.17 function

diguyVehicleController::get_desired_front_wheels_steer_angle

Prototype:

```
float get_desired_front_wheels_steer_angle();
```

5.83 diguyView

diguyView.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [diguy_typedefs.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Size and Position Functions](#)
- ◆ [Callback Functions](#)

Link against: **libdiguy**

5.83.1 class diguyView

```
class BDI_DECLSPEC_diguy diguyView
{
public:
    const char* get_name();
    const char* get_type_name();
    int show();
    int hide();
    int is_showing();
    const char* get_window_title();
    int set_window_title(const char* window_title);
    diguyViewCamera* get_camera();
    diguyViewFog* get_fog();
    diguyViewLight* get_light(int number = 0);
    int set_width(int width);
    int get_width();
    int set_height(int height);
    int get_height();
    int set_x(int x);
    int get_x();
    int set_y(int y);
    int get_y();
    enum {
        CALLBACK_ID_SHOW = 1,
        CALLBACK_ID_HIDE,
        CALLBACK_ID_WINDOW_TITLE_CHANGED,
        CALLBACK_ID_WINDOW_GEOMETRY_CHANGED
    };
#ifdef CPLUSPLUS_ONLY
    int add_callback(int callback_id,
        diguyViewCallback* callback,
        void* callback_params = 0,
        void* callback_user_data = 0);
    int remove_callback(int callback_id,
        diguyViewCallback* callback);
    int remove_callback_with_user_data(int callback_id,
        void* callback_user_data);
    int add_callback_script(int callback_id,
```

```
const char* callback_script,  
const char* callback_script_type = NULL);  
int remove_callback_script(int callback_id,  
const char* callback_script,  
const char* callback_script_type = NULL);
```

5.83.2 function diguyView::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the view

Callable From:

- ◇ C++
- ◇ Script

5.83.3 function diguyView::get_type_name

Prototype:

```
const char* get_type_name();
```

Description:

Returns the type name of the object. This pointer will never be NULL.

Returns:

type name of the object

Callable From:

- ◇ C++
- ◇ Script

5.83.4 function diguyView::show

Prototype:

```
int show();
```

Description:

Shows the view.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.83.5 function diguyView::hide

Prototype:

```
int hide();
```

Description:

Hides the view.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.83.6 function diguyView::is_showing

Prototype:

```
int is_showing();
```

Returns:

Returns 1 if the view is showing, 0 if not.

Callable From:

- ◇ C++
- ◇ Script
- ◇ Decision Bead

5.83.7 function diguyView::get_window_title**Prototype:**

```
const char* get_window_title();
```

Description:

Returns the window title of the view. This pointer will never be NULL.

Returns:

window title of the view

Callable From:

- ◇ C++
- ◇ Script

5.83.8 function diguyView::set_window_title**Prototype:**

```
int set_window_title(const char* window_title);
```

Description:

This function sets the window title of the view.

Arguments:

`window_title` new window title

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.83.9 function diguyView::get_camera

Prototype:

```
diguyViewCamera* get_camera();
```

Description:

This function returns a pointer to the view camera.

Callable From:

- ◇ C++
- ◇ Script

Returns:

pointer of type [diguyViewCamera](#); should never be NULL

5.83.10 function diguyView::get_fog

Prototype:

```
diguyViewFog* get_fog();
```

Description:

This function returns a pointer to the view fog.

Returns:

pointer of type [diguyViewFog](#); should never be NULL

Callable From:

- ◇ C++
- ◇ Script

5.83.11 function diguyView::get_light

Prototype:

```
diguyViewLight* get_light(int number = 0);
```

Description:

This function returns a pointer to the view lights.

Arguments:

number	number of light to be found

Returns:

pointer of type `diguyViewLight`; can be NULL if light at specified index doesn't exist

Callable From:

- ◇ C++
- ◇ Script

Size and Position Functions

5.83.12 function diguyView::set_width

Prototype:

```
int set_width(int width);
```

Description:

Sets the width of the view, in pixels.

Arguments:

width	new width of view, in pixels

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.83.13 function diguyView::get_width

Prototype:

```
int get_width();
```

Returns:

width in pixels of the view

Callable From:

- ◇ C++
- ◇ Script

5.83.14 function diguyView::set_height

Prototype:

```
int set_height(int height);
```

Description:

Sets the height of the view, in pixels. Increasing height value moves the view window bottom edge lower.

Arguments:

width	new height of view, in pixels

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.83.15 function diguyView::get_height

Prototype:

```
int get_height();
```

Returns:

height in pixels of the view

Callable From:

- ◇ C++
- ◇ Script

5.83.16 function diguyView::set_x**Prototype:**

```
int set_x(int x);
```

Description:

Sets the current x position of the view window relative to the left-hand edge of the screen.

Arguments:

x	new x coordinate of view

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.83.17 function diguyView::get_x**Prototype:**

```
int get_x();
```

Returns:

current x coordinate of the view window; see [set_x\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.83.18 function diguyView::set_y

Prototype:

```
int set_y(int y);
```

Description:

Sets the current y position of the view window relative to the upper edge of the screen. Increasing y moves the view window lower.

Arguments:

y	new y coordinate of view

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.83.19 function diguyView::get_y

Prototype:

```
int get_y();
```

Returns:

current y coordinate of the view window; see [set_y\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

Callback Functions

5.83.20 enumeration

```
enum {
    CALLBACK_ID_SHOW = 1,
    CALLBACK_ID_HIDE,
    CALLBACK_ID_WINDOW_TITLE_CHANGED,
    CALLBACK_ID_WINDOW_GEOMETRY_CHANGED
};
```

Description:

This is an enumeration of the different callbacks that can be registered with [add_callback\(\)](#) and [add_callback_script\(\)](#).

Usable From:

- ◇ C++
- ◇ Script

5.83.21 function diguyView::add_callback

Prototype:

```
int add_callback(int callback_id,
    diguyViewCallback* callback,
    void* callback_params = 0,
    void* callback_user_data = 0);
```

Description:

This function adds a user callback.

Arguments:

callback	pointer to function with prototype diguyViewCallback (typedefed above)
callback_id	integer id of when this callback is to be called
callback_params	not currently used; pass NULL
callback_user_data	pointer for user's own use; DI-Guy will do nothing to the contents of this pointer beyond passing it back when the callback is invoked

callback_id should be one of the following values:

CALLBACK_ID_SHOW

This callback will be called whenever the view is to be shown, whether due to a call to the `diguyView::show()` function above or due to the result of a decision or script.

There are no parameters for this callback.

CALLBACK_ID_HIDE

This callback will be called whenever the view is to be hidden, whether due to a call to the `diguyView::hide()` function above or due to the result of a decision or script.

There are no parameters for this callback.

CALLBACK_ID_WINDOW_TITLE_CHANGED

This callback will be called whenever the view's window title has been changed.

There are no parameters for this callback.

Callback Return Values:

Callbacks return a value of type `diguyCallbackReturn`, which will be `DIGUY_CALLBACK_STOP` or `DIGUY_CALLBACK_CONTINUE`. If the callback returns `DIGUY_CALLBACK_STOP`, the default handler of the function will not be called; the callback is asserting that it has done everything necessary for the function call. If the callback returns `DIGUY_CALLBACK_CONTINUE`, the default handler for the function will be called after the callback.

Returns:

0 on success, -1 on failure

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.83.22 function diguyView::remove_callback**Prototype:**

```
int remove_callback(int callback_id,
    diguyViewCallback* callback);
```

Description:

This function removes a user callback. All callbacks matching the specified callback_id and callback function will be removed.

Arguments:

callback_id	integer id of when this callback is to be called
callback	pointer to function with prototype diguyCharacterCallback (typedefed above)

Returns:

0 on success, -1 on failure

5.83.23 function diguyView::remove_callback_with_user_data**Prototype:**

```
int remove_callback_with_user_data(int callback_id,
    void* callback_user_data);
```

Description:

This function removes a user callback. All callbacks matching the specified callback_id and callback_user_data pointer will be removed.

Arguments:

callback_id	integer id of when this callback is to be called
callback_user_data	pointer for user's own use

Returns:

0 on success, -1 on failure

5.83.24 function diguyView::add_callback_script**Prototype:**

```
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function adds a user callback script. Callback scripts can be removed with [remove_callback_script\(\)](#).

See [diguyCharacter::add_callback_script\(\)](#) for an example of use.

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback to be added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

Lua specific:

When the script is called, the object for which it is being called will be in the `callback_object` global.

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.83.25 function `diguyView::remove_callback_script`

Prototype:

```
int remove_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function removes a user callback script previously added with `add_callback_script()`.

See `diguyCharacter::remove_callback_script()` for an example of use.

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback previously added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script`, all callback scripts whose ids match `callback_id` and whose types match `callback_script_type` will be removed.

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

Lua specific:

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.84 diguyViewButtonPanel

diguyViewButtonPanel.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h | stdlib.h

Contents:

Link against: libdiguy

5.84.1 class diguyViewButtonPanel

```
class BDI_DECLSPEC_diguy diguyViewButtonPanel
{
public:
    static diguyViewButtonPanel* find_or_create_instance(const char* name,
        int horizontal = 0,
        int title = 1);
    static diguyViewButtonPanel* make_instance(const char* name,
        int horizontal = 0,
        int title = 1);
    static diguyViewButtonPanel* get_instance(const char* name);
    static int destroy_instance(const char* name);
    static int destroy_all();
    const char* get_name();
    int get_position(int* ret_x, int* ret_y);
    int get_width();
    int get_height();
    int set_visible(int visible);
    int set_minimized_state(int minimized);
    int set_position(int x, int y);
    int set_dimensions(int w, int h);
    int set_depth(int z);
    int set_justifications(int h, int v);
    int set_spacing(int spacing);
    int set_title(const char* title);
    int add_button(const char* name,
        const char* lua_object_name,
        const char* lua_function_name,
        const char* style_sheet = NULL);
    int add_button_with_script(const char* name, const char* script,
        const char* style_sheet = NULL);
    int set_button_name(int index, const char* name);
    int set_allow_toggle(int index, int can_toggle);
    int set_selected_button(int index);
    int set_selected_button_by_name(const char* name);
    int add_int_slider(const char* name,
        const char* lua_object_name,
        const char* lua_function_name,
        int init_val,
        int min_val,
        int max_val,
        const char* style_sheet = NULL);
```

```

int add_float_slider(const char* name,
    const char* lua_object_name,
    const char* lua_function_name,
    float init_val,
    float min_val,
    float max_val,
    const char* style_sheet = NULL);
int set_slider_value(const char* name, float value);
int add_spacer(int pixels);
void set_item_visibility(int index, bool visible);
void remove_all_items();
void set_open_script(const char* lua_script);
void set_minimize_script(const char* lua_script);
int set_style_sheet(const char* sheet);

```

5.84.2 function diguyViewButtonPanel::find_or_create_instance

Prototype:

```

static diguyViewButtonPanel* find_or_create_instance(const char* name,
    int horizontal = 0,
    int title = 1);

```

Description:

Returns the named button panel or creates it.

Arguments:

name	name of panel (must be unique)
horizontal	if 1, make a horizontal panel
title	if 1, show panel title

Returns:

pointer to the button panel

Callable From:

- ◇ C++
- ◇ Script

5.84.3 function diguyViewButtonPanel::make_instance

Prototype:

```
static diguyViewButtonPanel* make_instance(const char* name,  
int horizontal = 0,  
int title = 1);
```

Description:

Makes a button panel with the given name and attributes.

Arguments:

name	name of panel (must be unique)
horizontal	if 1, make a horizontal panel
title	if 1, show panel title

Returns:

pointer to the button panel

Callable From:

- ◇ C++
- ◇ Script

5.84.4 function diguyViewButtonPanel::get_instance

Prototype:

```
static diguyViewButtonPanel* get_instance(const char* name);
```

Description:

Returns a pointer to the named button panel, if it exists.

Arguments:

name	name of panel

Returns:

pointer to the button panel or NULL

Callable From:

- ◇ C++
- ◇ Script

5.84.5 function diguyViewButtonPanel::destroy_instance

Prototype:

```
static int destroy_instance(const char* name);
```

Description:

Destroys named button panel, if found.

Arguments:

name	name of panel

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.84.6 function diguyViewButtonPanel::destroy_all

Prototype:

```
static int destroy_all();
```

Description:

Destroys all button panels.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.84.7 function diguyViewButtonPanel::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the object

Callable From:

- ◇ C++
- ◇ Script

5.84.8 function diguyViewButtonPanel::get_position

Prototype:

```
int get_position(int* ret_x, int* ret_y);
```

Description:

Returns the x and y coordinate of the panel.

Callable From:

- ◇ C++
- ◇ Script

5.84.9 function diguyViewButtonPanel::get_width

Prototype:

```
int get_width();
```

Description:

Returns the width of the panel

Callable From:

- ◇ C++
- ◇ Script

5.84.10 function diguyViewButtonPanel::get_height

Prototype:

```
int get_height();
```

Description:

Returns the height of the panel

Callable From:

- ◇ C++
- ◇ Script

5.84.11 function diguyViewButtonPanel::set_visible**Prototype:**

```
int set_visible(int visible);
```

Description:

Makes button panel visible or invisible.

Arguments:

visible pass 1 to make visible, 0 to make invisible

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.84.12 function diguyViewButtonPanel::set_minimized_state**Prototype:**

```
int set_minimized_state(int minimized);
```

Description:

Makes button panel minimized (or un-minimized).

Arguments:

visible pass 1 to make minimized, 0 to make full size

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.84.13 function diguyViewButtonPanel::set_position**Prototype:**

```
int set_position(int x, int y);
```

Description:

Sets the position of the panel in its parent window. Coordinates work differently if panel is *rightbottom justified on its xy* (see `set_justification( )`). In that case, the coordinate will imply the position of the panel's *rightbottom edge relative to the rightbottom* edge of the containing window.

Arguments:

x	x position, in pixels
y	y position, in pixels

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.84.14 function diguyViewButtonPanel::set_dimensions**Prototype:**

```
int set_dimensions(int w, int h);
```

Description:

Sets the desired dimensions of the panel.

Arguments:

w	width, in pixels
h	height, in pixels

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.84.15 function diguyViewButtonPanel::set_depth**Prototype:**

```
int set_depth(int z);
```

Description:

Sets the desired z depth of the panel.

Arguments:

z	depth

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.84.16 function diguyViewButtonPanel::set_justifications

Prototype:

```
int set_justifications(int h, int v);
```

Description:

Sets the horizontal/vertical justifications of the panel (see description in [set_position\(\)](#)).

Arguments:

h	horizontal justification (> 0 for left, 0 for center, < 0 for right)
v	vertical justification (> 0 for top, 0 for center, < 0 for bottom)

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.84.17 function diguyViewButtonPanel::set_spacing**Prototype:**

```
int set_spacing(int spacing);
```

Description:

Sets the spacing between items in panel.

Arguments:

spacing	spacing in pixels

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.84.18 function diguyViewButtonPanel::set_title**Prototype:**

```
int set_title(const char* title);
```

Description:

Sets the display title of the panel.

Arguments:

title	text to show

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.84.19 function diguyViewButtonPanel::add_button**Prototype:**

```
int add_button(const char* name,
               const char* lua_object_name,
               const char* lua_function_name,
               const char* style_sheet = NULL);
```

Description:

Adds a button to the panel and specifies the lua script that should be run upon press.

Arguments:

name	text of button
lua_object_name	name of lua table containing function to run
lua_function_name	name of lua function to run
style_sheet	optional css to set on button

An example style sheet string: "QPushButton {background-color : rgba(192, 210, 192, 192); }"

Returns:

index of the new button (zero-indexed)

Callable From:

- ◇ C++
- ◇ Script

5.84.20 function diguyViewButtonPanel::add_button_with_script**Prototype:**

```
int    add_button_with_script(const char* name, const char* script,
                             const char* style_sheet = NULL);
```

Description:

Adds a button to the panel and specifies the lua script that should be run upon press.

Arguments:

<code>name</code>	text of button
<code>script</code>	Lua script to run (the script itself)
<code>style_sheet</code>	optional css to set on button

An example style sheet string: "QPushButton {background-color : rgba(192, 210, 192, 192); }"

Returns:

index of the new button (zero-indexed)

Callable From:

- ◇ C++
- ◇ Script

5.84.21 function diguyViewButtonPanel::set_button_name**Prototype:**

```
int set_button_name(int index, const char* name);
```

Description:

Sets the name (text) of button at index n (zero-indexed).

Arguments:

index index of button (starting from 0)

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
◇ Script

5.84.22 function diguyViewButtonPanel::set_allow_toggle**Prototype:**

```
int set_allow_toggle(int index, int can_toggle);
```

Description:

Sets whether or not the specified button will be a toggle button. A toggle button can be clicked back and forth between an on and off state, whereas a regular button provides only a once-off action.

Arguments:

index index of button (starting from 0)
can_toggle true if toggle button

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.84.23 function diguyViewButtonPanel::set_selected_button

Prototype:

```
int set_selected_button(int index);
```

Description:

Sets the specified button to be the "selected" one within the panel. The selected button may have a special appearance depending on style sheet. See notes for [set_style_sheet\(\)](#).

Arguments:

index	index of button (starting from 0)

Returns:

0 on success, -1 on failure

Callable From:

◇ C++
 ◇ Script

5.84.24 function diguyViewButtonPanel::set_selected_button_by_name

Prototype:

```
int set_selected_button_by_name(const char* name);
```

Description:

Similar to [set_selected_button\(\)](#), but using button name (text).

Arguments:

name	name (text) of button

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

◇ Script

5.84.25 function diguyViewButtonPanel::add_int_slider**Prototype:**

```
int add_int_slider(const char* name,  
                  const char* lua_object_name,  
                  const char* lua_function_name,  
                  int init_val,  
                  int min_val,  
                  int max_val,  
                  const char* style_sheet = NULL);
```

Description:

Adds an integer slider to the panel and specifies the lua script that should be run upon the slider changing value. lua function should be:

```
function lua_object_name:lua_function_name(value)
```

Arguments:

name	text of label description
lua_object_name	name of lua table containing function to run

`lua_function_name` name of lua function to run

Returns:

index of the new slider (zero-indexed)

Callable From:

- ◇ C++
- ◇ Script

5.84.26 function diguyViewButtonPanel::add_float_slider

Prototype:

```
int add_float_slider(const char* name,
                    const char* lua_object_name,
                    const char* lua_function_name,
                    float init_val,
                    float min_val,
                    float max_val,
                    const char* style_sheet = NULL);
```

Description:

Adds a continuous floating slider to the panel and specifies the lua script that should be run upon the slider changing value. lua function should be:

```
function lua_object_name:lua_function_name(value)
```

Arguments:

<code>name</code>	text of label description
<code>lua_object_name</code>	name of lua table containing function to run

`lua_function_name` name of lua function to run

Returns:

index of the new slider (zero-indexed)

Callable From:

- ◇ C++
- ◇ Script

5.84.27 function diguyViewButtonPanel::set_slider_value

Prototype:

```
int set_slider_value(const char* name, float value);
```

Description:

Updates specified slider with the value specified.

Arguments:

<code>name</code>	name of slider
<code>value</code>	value of slider

5.84.28 function diguyViewButtonPanel::add_spacer**Prototype:**

```
int add_spacer(int pixels);
```

Description:

Adds a vertical or horizontal spacer element.

5.84.29 function diguyViewButtonPanel::set_item_visibility**Prototype:**

```
void set_item_visibility(int index, bool visible);
```

Description:

Change visibility of item specified by index

5.84.30 function diguyViewButtonPanel::remove_all_items**Prototype:**

```
void remove_all_items();
```

Description:

Clears all items out of panel

5.84.31 function diguyViewButtonPanel::set_open_script**Prototype:**

```
void set_open_script(const char* lua_script);
```

Description:

Sets a Lua script to be called when panel is opened (un-minimized).

Arguments:

`lua_script` the entire script

5.84.32 function diguyViewButtonPanel::set_minimize_script**Prototype:**

```
void set_minimize_script(const char* lua_script);
```

Description:

Sets a Lua script to be called when panel is opened (un-minimized).

Arguments:

`lua_script` the entire script

5.84.33 function diguyViewButtonPanel::set_style_sheet**Prototype:**

```
int set_style_sheet(const char* sheet);
```

Description:

Passes a Qt style sheet to the panel.

Sample one (user must replace "MAIN_BG_COLOR" and other tags with meaningful values):

Note:

"isSelected" used to set visual properties of selected button

```
basic_style = [[
* {
    background-color : rgba( 192, 192, 192, 192 );
    color : white;
```

```

    }
    QFrame#m_button_frame {
        background-color : MAIN_BG_COLOR;
    }
    QLabel#m_title_label {
        color : white;
    }
    QPushButton {
        color : BUTTON_TEXT_COLOR;
        background-color : BUTTON_BG_COLOR;
    }
    QPushButton[isSelected="true"] {
        color : BUTTON_SELECTED_TEXT_COLOR;
        background-color : BUTTON_SELECTED_BG_COLOR;
    }
}
];

```

Sample two (user must replace "MAIN_BG_COLOR" and other tags with meaningful values):

Note:

"isToggled" used to set visual properties on button that is toggled on

```

    basic_style = [
        * {
            background-color : rgba( 192, 192, 192, 192 );
            color : white;
        }
    ]
    QFrame#m_button_frame {
        background-color : MAIN_BG_COLOR;
    }
    QLabel#m_title_label {
        color : white;
    }
    QPushButton {
        color : BUTTON_TEXT_COLOR;
        background-color : BUTTON_BG_COLOR;
    }
    QPushButton[isToggled="true"] {
        color : BUTTON_TOGGLED_TEXT_COLOR;
        background-color : BUTTON_TOGGLED_BG_COLOR;
    }
}
];

```

Arguments:

sheet the style sheet

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.85 diguyViewCamera

diguyViewCamera.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [diguy_typedefs.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [General Functions](#)
- ◆ [Load and Apply Functions](#)
- ◆ [Far Position Camera Functions](#)
- ◆ [Look At and Look From Functions](#)
- ◆ [C++ Callback Functions](#)
- ◆ [C++/Script Callback Functions](#)
- ◆ [Experimental Functions](#)

5.85.1 class diguyViewCamera

```
class BDI_DECLSPEC_diguy diguyViewCamera
{
public:
    const char* get_name();
    const char* get_type_name();
    void set_near_plane(float near_plane);
    float get_near_plane();
    void set_far_plane(float far_plane);
    float get_far_plane();
    void set_fov(float fov);
    void set_max_fov(float fov);
    void set_min_fov(float fov);
    float get_fov();
    float get_min_fov();
    float get_max_fov();
    void set_aspect(float aspect);
    float get_aspect();
    void set_position(float x, float y, float z);
    float get_position_x();
    float get_position_y();
    float get_position_z();
    void set_desired_position(float x, float y, float z, bool preserve_fix = false);
    void set_orientation(float yaw, float roll, float pitch);
    void set_orientation_roll(float roll);
    float get_orientation_yaw();
    float get_orientation_roll();
    float get_orientation_pitch();
    void set_pitch_lower_limit(float limit);
    float get_pitch_lower_limit();
    void set_pitch_upper_limit(float limit);
    float get_pitch_upper_limit();
    int get_pitch_has_limits();
    void set_distance(float distance);
    void set_max_track_distance(float distance);
```

```

void set_min_track_distance(float distance);
float get_distance();
float get_min_track_distance();
float get_max_track_distance();
void set_fix(float x, float y, float z);
float get_fix_x();
float get_fix_y();
float get_fix_z();
void set_desired_fix(float x, float y, float z);
int set_move_mode(diguyViewCameraMoveMode move_mode);
diguyViewCameraMoveMode get_move_mode();
int set_speed(float speed);
float get_speed();
int increase_speed();
int decrease_speed();
int set_projection_mode(diguyViewCameraProjectionMode projection_mode);
diguyViewCameraProjectionMode get_projection_mode();
int load_settings(const char* settings_name);
int load_settings(diguyViewCameraSettings* settings);
void apply_settings();
void set_position_double(double x, double y, double z);
void get_position_double(double* x, double* y, double* z);
void get_segment_origin(float* x, float* y, float* z);
void get_position_segment_offset(float* x, float* y, float* z);
void get_fix_segment_offset(float* x, float* y, float* z);
void set_fix_double(double x, double y, double z);
void set_track_name(const char* track_name);
const char* get_track_name();
void set_track_group_name(const char* track_name);
const char* get_track_group_name();
void set_track_link_name(const char* track_link_name);
const char* get_track_link_name();
void set_track_offset(float x, float y, float z);
float get_track_offset_x();
float get_track_offset_y();
float get_track_offset_z();
void set_track_offset_is_in_world_coordinates(int value);
int get_track_offset_is_in_world_coordinates();
void set_dolly_name(const char* dolly_name);
const char* get_dolly_name();
void set_dolly_offset(float x, float y, float z);
float get_dolly_offset_x();
float get_dolly_offset_y();
float get_dolly_offset_z();
void set_dolly_link_name(const char* dolly_link_name);
const char* get_dolly_link_name();
void set_dolly_offset_is_in_world_coordinates(int value);
int get_dolly_offset_is_in_world_coordinates();
int get_symbolic_view();
enum {
    CALLBACK_ID_SETTINGS_CHANGED = 1
};
#ifdef CPLUSPLUS_ONLY
int add_callback(int callback_id,
    diguyViewCameraCallback* callback,
    void* callback_params = 0,
    void* callback_user_data = 0);
int remove_callback(int callback_id,
    diguyViewCameraCallback* callback);
int remove_callback_with_user_data(int callback_id,
    void* callback_user_data);

```

```

#endif
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int remove_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
void set_speed_fov_degrees_per_wheel_click(float fov_change);
float get_speed_fov_degrees_per_wheel_click();
void set_speed_meters_per_second(float speed);
float get_speed_meters_per_second();
void set_require_smooth_servo(int require);
int get_require_smooth_servo();
void set_servo_time_modifier( float modifier );
float get_servo_time_modifier();
void set_track_camera_avoids_obstacles(int avoid);
int get_track_camera_avoids_obstacles();
void set_ignore_mouse drags(int ignore);
int get_ignore_mouse drags();
#ifdef CPLUSPLUS_ONLY
    void update_frustum(float* proj_matrix, float* model_matrix);
#endif

```

General Functions

All functions in this section are callable from:

- ◆ C++
- ◆ Script

5.85.2 function diguyViewCamera::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the object

5.85.3 function diguyViewCamera::get_type_name

Prototype:

```
const char* get_type_name();
```

Description:

Returns the type name of the object. This pointer will never be NULL.

Returns:

type name of the object

5.85.4 function diguyViewCamera::set_near_plane

Prototype:

```
void set_near_plane(float near_plane);
```

Description:

This sets the distance from the camera to the near plane.

Arguments:

`near_plane` distance to near plane, in meters

5.85.5 function diguyViewCamera::get_near_plane

Prototype:

```
float get_near_plane();
```

Returns:

distance from the camera to the near plane, in meters

5.85.6 function diguyViewCamera::set_far_plane

Prototype:

```
void set_far_plane(float far_plane);
```

Description:

This sets the distance from the camera to the far plane.

Arguments:

`far_plane` distance to far plane, in meters

5.85.7 function diguyViewCamera::get_far_plane

Prototype:

```
float get_far_plane();
```

Returns:

distance from the camera to the far plane, in meters

5.85.8 function diguyViewCamera::set_fov

Prototype:

```
void set_fov(float fov);
```

Description:

This sets the field-of-view of the current camera.

Arguments:

fov	field-of-view, in degrees

5.85.9 function diguyViewCamera::set_max_fov

Prototype:

```
void set_max_fov(float fov);
```

Description:

This sets the maximum field-of-view of the current camera. This affects the FOV that can be set via set_fov.

Arguments:

fov	max field-of-view, in degrees

5.85.10 function diguyViewCamera::set_min_fov**Prototype:**

```
void set_min_fov(float fov);
```

Description:

This sets the minimum field-of-view of the current camera. This affects the FOV that can be set via set_fov.

Arguments:

fov	min field-of-view, in degrees

5.85.11 function diguyViewCamera::get_fov**Prototype:**

```
float get_fov();
```

Returns:

field-of-view of the camera, in degrees

5.85.12 function diguyViewCamera::get_min_fov**Prototype:**

```
float get_min_fov();
```

Returns:

min field-of-view of the camera, in degrees

5.85.13 function diguyViewCamera::get_max_fov**Prototype:**

```
float get_max_fov();
```

Returns:

max field-of-view of the camera, in degrees

5.85.14 function diguyViewCamera::set_aspect

Prototype:

```
void set_aspect(float aspect);
```

Description:

This function sets the aspect ratio of the camera.

Arguments:

aspect	aspect ratio; 1.3333 (4/3) is the default

5.85.15 function diguyViewCamera::get_aspect

Prototype:

```
float get_aspect();
```

Returns:

aspect ratio of the camera

5.85.16 function diguyViewCamera::set_position

Prototype:

```
void set_position(float x, float y, float z);
```

Description:

This function sets the position of the camera.

Arguments:

x, y, z position in meters from the origin

5.85.17 function diguyViewCamera::get_position_x

Prototype:

```
float get_position_x();
```

Returns:

x position, in meters from the origin

5.85.18 function diguyViewCamera::get_position_y**Prototype:**

```
float get_position_y();
```

Returns:

y position, in meters from the origin

5.85.19 function diguyViewCamera::get_position_z**Prototype:**

```
float get_position_z();
```

Returns:

z position, in meters from the origin

5.85.20 function diguyViewCamera::set_desired_position**Prototype:**

```
void set_desired_position(float x, float y, float z, bool preserve_fix = false);
```

Description:

This function sets the desired position of the camera. The camera will interpolate over time to this position.

Arguments:

`x, y, z` position in meters from the origin

5.85.21 function diguyViewCamera::set_orientation**Prototype:**

```
void set_orientation(float yaw, float roll, float pitch);
```

Description:

This function sets the orientation of the camera. Note that this affects the fix.

NOTE: Pitch angles are positive for looking down, not up!

Arguments:

<code>yaw</code>	rotation about z axis, in degrees
<code>roll</code>	rotation about x axis, in degrees
<code>pitch</code>	rotation about y axis, in degrees

5.85.22 function diguyViewCamera::set_orientation_roll**Prototype:**

```
void set_orientation_roll(float roll);
```

Description:

This function sets the roll of the camera.

Arguments:

roll	rotation about x axis, in radians

5.85.23 function diguyViewCamera::get_orientation_yaw**Prototype:**

```
float get_orientation_yaw();
```

Returns:

rotation about z axis, in degrees

5.85.24 function diguyViewCamera::get_orientation_roll**Prototype:**

```
float get_orientation_roll();
```

Returns:

rotation about x axis, in degrees

5.85.25 function diguyViewCamera::get_orientation_pitch**Prototype:**

```
float get_orientation_pitch();
```

Returns:

rotation about y axis, in degrees

5.85.26 function diguyViewCamera::set_pitch_lower_limit**Prototype:**

```
void set_pitch_lower_limit(float limit);
```

Description:

Sets the lower limit of the camera pitch **up**. Default is very close to -90 degrees.

NOTE: Pitch angles are positive for looking down, not up!

Arguments:

limit	rotation about y axis, in degrees

5.85.27 function diguyViewCamera::get_pitch_lower_limit**Prototype:**

```
float get_pitch_lower_limit();
```

Returns:

lower limit of the camera pitch (up/down)

5.85.28 function diguyViewCamera::set_pitch_upper_limit**Prototype:**

```
void set_pitch_upper_limit(float limit);
```

Description:

Sets the upper limit of the camera pitch **down**. Default is very close to 90 degrees.

NOTE: Pitch angles are positive for looking down, not up!

Arguments:

limit	rotation about y axis, in degrees

5.85.29 function diguyViewCamera::get_pitch_upper_limit**Prototype:**

```
float get_pitch_upper_limit();
```

Returns:

upper limit of the camera pitch (up/down)

5.85.30 function diguyViewCamera::get_pitch_has_limits**Prototype:**

```
int get_pitch_has_limits();
```

Returns:

1 if pitch has limits, 0 if not

5.85.31 function diguyViewCamera::set_distance**Prototype:**

```
void set_distance(float distance);
```

Description:

This function sets the distance from the position to the fix point. Note that this affects the fix values.

Arguments:

distance	distance in meters

5.85.32 function diguyViewCamera::set_max_track_distance**Prototype:**

```
void set_max_track_distance(float distance);
```

Description:

This function sets the maximum allowed distance from the position to the fix point.

Arguments:

distance distance in meters; set to -1 if no max to be used

5.85.33 function diguyViewCamera::set_min_track_distance**Prototype:**

```
void set_min_track_distance(float distance);
```

Description:

This function sets the minimum allowed distance from the position to the fix point.

Arguments:

distance distance in meters; set to -1 if no min to be used

5.85.34 function diguyViewCamera::get_distance**Prototype:**

```
float get_distance();
```

Returns:

distance from position to fix point, in meters

5.85.35 function diguyViewCamera::get_min_track_distance**Prototype:**

```
float get_min_track_distance();
```

Returns:

minimum allowed tracking distance, in meters; will be less than zero if not applicable

5.85.36 function diguyViewCamera::get_max_track_distance**Prototype:**

```
float get_max_track_distance();
```

Returns:

maximum allowed tracking distance, in meters; will be less than zero if not applicable

5.85.37 function diguyViewCamera::set_fix**Prototype:**

```
void set_fix(float x, float y, float z);
```

Description:

This function sets the fix point of the camera. Note that this affects the orientation.

Arguments:

`x, y, z` position in meters from the origin

5.85.38 function diguyViewCamera::get_fix_x**Prototype:**

```
float get_fix_x();
```

Returns:

fix point in x, in meters

5.85.39 function diguyViewCamera::get_fix_y**Prototype:**

```
float get_fix_y();
```

Returns:

fix point in y, in meters

5.85.40 function diguyViewCamera::get_fix_z**Prototype:**

```
float get_fix_z();
```

Returns:

fix point in z, in meters

5.85.41 function diguyViewCamera::set_desired_fix**Prototype:**

```
void set_desired_fix(float x, float y, float z);
```

Description:

This function sets the desired fix point of the camera. The camera will interpolate to there from its current fix over time.

Arguments:

x, y, z position in meters from the origin

5.85.42 function diguyViewCamera::set_move_mode**Prototype:**

```
int set_move_mode(diguyViewCameraMoveMode move_mode);
```

Description:

This function sets the movement mode of the camera.

In DI-Guy Scenario in Camera Input Mode, the movement mode of the camera determines what will happen when mouse buttons are pressed in the view window.

Arguments:

move_mode	new movement mode of the camera; enumeration of type diguyViewCameraMoveMode
DIGUY_CAMERA_MOVE_FORWARD	left mouse press moves camera forward, right mouse press moves camera back
DIGUY_CAMERA_MOVE_SIDEWAYS	left mouse press moves camera left, right mouse press moves camera right
DIGUY_CAMERA_MOVE_VERTICAL	left mouse press moves camera up, right mouse press moves camera down

Returns:

0 on success, -1 on failure

5.85.43 function diguyViewCamera::get_move_mode**Prototype:**

```
diguyViewCameraMoveMode get_move_mode();
```

Returns:

the most recent setting made by `set_move_mode()`

5.85.44 function diguyViewCamera::set_speed**Prototype:**

```
int set_speed(float speed);
```

Description:

This function sets the movement speed of the camera when the camera is translating due to user input.

In DI-Guy Scenario in Camera Input Mode, the movement speed of the camera determines how fast the camera will move when mouse buttons are pressed in the view window.

Arguments:

speed	speed of camera movement is meters per second

Returns:

0 on success, -1 on failure

5.85.45 function diguyViewCamera::get_speed**Prototype:**

```
float get_speed();
```

Returns:

the most recent setting made by [set_speed\(\)](#)

5.85.46 function diguyViewCamera::increase_speed**Prototype:**

```
int increase_speed();
```

Description:

This function increases the movement speed of the camera when the camera is translating due to user input. The increase is typically by a factor of 2.

Returns:

0 on success, -1 on failure

5.85.47 function diguyViewCamera::decrease_speed**Prototype:**

```
int decrease_speed();
```

Description:

This function decreases the movement speed of the camera when the camera is translating due to user input. The decrease is typically by a factor of 2.

Returns:

0 on success, -1 on failure

5.85.48 function diguyViewCamera::set_projection_mode**Prototype:**

```
int set_projection_mode(diguyViewCameraProjectionMode projection_mode);
```

Description:

This function sets the projection mode of the camera.

In DI-Guy Scenario in Camera Input Mode, the movement mode of the camera determines what will happen when mouse buttons are pressed in the view window.

Arguments:

projection_mode	new projection mode of the camera; enumeration of type diguyViewCameraProjectionMode
DIGUY_CAMERA_PROJECTION_PERSPECTIVE	standard perspective camera, can move and rotate freely

DIGUY_CAMERA_PROJECTION_PLANVIEW_XY always views in negative Z direction

DIGUY_CAMERA_PROJECTION_PLANVIEW_XZ always views in negative Y direction

DIGUY_CAMERA_PROJECTION_PLANVIEW_YZ always views in negative X direction

5.85.49 function diguyViewCamera::get_projection_mode

Prototype:

```
diguyViewCameraProjectionMode get_projection_mode();
```

Returns:

the most recent setting made by [set_projection_mode\(\)](#)

Load and Apply Functions

All functions in this section are callable from:

- ◆ C++
- ◆ Script

5.85.50 function diguyViewCamera::load_settings

Prototype:

```
int load_settings(const char* settings_name);
```

Description:

Loads the specified settings into this camera.

Returns:

0 on success, -1 on failure

5.85.51 function diguyViewCamera::load_settings**Prototype:**

```
int load_settings(diguyViewCameraSettings* settings);
```

Description:

Loads the specified settings into this camera.

Returns:

0 on success, -1 on failure

5.85.52 function diguyViewCamera::apply_settings**Prototype:**

```
void apply_settings();
```

Description:

This function applies this camera's settings.

OpenGL Version:

This function sets the current viewpoint in these environments based on information in this camera.

Direct3D Version:

This function has no effect. The viewpoint in these environments must be set manually.

Far Position Camera Functions

The following functions are for working with a "far position", or high precision, cameras.

One approach for dealing with high-magnitude floating point numbers is to simply use doubles instead of floats. However, many rendering pipelines use single-precision floats internally, making the use of doubles to store transformations only a partial solution.

Another approach for rendering objects that are extremely far from the origin without having floating point precision problems is to move the camera and all objects it is observing closer to the origin. This avoids placing large magnitude numbers in transformation matrices, which with most graphics environments causes precision loss. This is the approach DI-Guy uses.

DI-Guy internally stores high precision positions as a "far position": a segment (integer x and y coordinates), a segment size (a floating point number), and a floating point offset into the segment (float x, y, and z coordinates). This effectively compartmentalizes x and y coordinates into a grid where each cell, or segment, of the grid is of size (segment size x segment size) meters. (Currently z coordinates are stored "as is", in single precision floats.)

The world coordinate position of a far position can be computed using the following formula:

$$\text{world position} = (\text{segment} * \text{segment\_size}) + \text{segment\_offset}$$

If done using doubles, the above formula provides a very accurate position, even at very high magnitudes. This is the calculation the `get_position_double()` function uses.

Assuming the camera and an object are in the same segment, the local (camera-relative) position of the object is simply its offset.

If a `diguyViewCamera` is being used to control the graphics environment viewpoint, the viewpoint matrix can be calculated from the position segment offset and the fix segment offset.

Example:

```
float pos_x, pos_y, pos_z;
scenario_camera->get_position_segment_offset(

float fix_x, fix_y, fix_z;
scenario_camera->get_fix_segment_offset(

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(pos_x, pos_y, pos_z,
          fix_x, fix_y, fix_z,
          0.0, 0.0, 1.0);
```

To render non-DI-Guy objects correctly, the segment origin translation must be subtracted from the position of the objects. This is done automatically for DI-Guy characters in the OpenGL version of DI-Guy if the `diguy_ogl_set_use_far_positions(1)` call is

made.

Example:

Far position cameras are used in the `diguy_oglvview_far_settingsview_far_settings.cpp` example.

All functions in this section are callable from:

- ◆ C++
- ◆ Script

5.85.53 function `diguyViewCamera::set_position_double`

Prototype:

```
void set_position_double(double x, double y, double z);
```

Description:

This function sets the position of this camera using double precision floating point numbers. The numbers will be converted internally into a DI-Guy far position.

Arguments:

x	x position, in meters
y	y position, in meters
z	z position, in meters

5.85.54 function diguyViewCamera::get_position_double**Prototype:**

```
void get_position_double(double* x, double* y, double* z);
```

Description:

This function returns the world-space position of this camera. These values are computed from the DI-Guy far position of the camera. The results are placed into the passed pointers.

Arguments:

x	x position, in meters
y	y position, in meters
z	z position, in meters

5.85.55 function diguyViewCamera::get_segment_origin**Prototype:**

```
void get_segment_origin(float* x, float* y, float* z);
```

Description:

This function returns the world-space origin of the far position segment the camera is in. This segment origin plus the position segment offset returned by [get_position_segment_offset\(\)](#) yields the final world-space position of the camera.

The results are placed into the passed pointers.

Arguments:

x	x component of origin, in meters
y	y component of origin, in meters
z	z component of origin, in meters

5.85.56 function diguyViewCamera::get_position_segment_offset

Prototype:

```
void get_position_segment_offset(float* x, float* y, float* z);
```

Description:

This function returns the far position segment offset part of the camera's position. The results are placed into the passed pointers.

Arguments:

x	x component of segment offset, in meters
y	y component of segment offset, in meters
z	z component of segment offset, in meters

5.85.57 function diguyViewCamera::get_fix_segment_offset

Prototype:

```
void get_fix_segment_offset(float* x, float* y, float* z);
```

Description:

This function returns the far position offset portion of the camera's fix point. The results are placed into the passed pointers.

Arguments:

x	x component of fix segment offset, in meters
y	y component of fix segment offset, in meters
z	z component of fix segment offset, in meters

5.85.58 function diguyViewCamera::set_fix_double**Prototype:**

```
void set_fix_double(double x, double y, double z);
```

Description:

This function sets the fix point of the camera using double precision floating point numbers. The numbers will be converted internally into a DI-Guy far position. Note that this affects the orientation.

Arguments:

x	x position, in meters
y	y position, in meters
z	z position, in meters

Look At and Look From Functions

All functions in this section are callable from:

- ◆ C++
- ◆ Script

5.85.59 function diguyViewCamera::set_track_name**Prototype:**

```
void set_track_name(const char* track_name);
```

Description:

This function sets the name of the character the camera should look at, if such a character can be found. This will cause fix values to change over time.

By default the camera will look at the idealized position of the character (a spot on the ground typically under the center of gravity for human characters). To look at a particular part of a character, call [set_track_link_name\(\)](#).

The spot looked at on a link can be offset by calling [set_track_offset\(\)](#). Whether the offset is in world coordinates or character-local coordinates is controlled by a call to [set_track_offset_is_in_world_coordinates\(\)](#).

Arguments:

`track_name` name of character to look at, or NULL if no character is to be tracked

C++ Example

The following example shows how to use some of the pose functions.

```
//
// Get a pointer to the primary view's camera, and
// to the first character.
//
diguyViewCamera* camera = scenario->get_scenario_camera();
diguyCharacter* character = scenario->get_character_at_index(0);

//
// Look just a little bit above the character's head.
//
camera->set_track_name(character->get_name());
camera->set_track_link_name("cervical");
camera->set_track_offset(0.0f, 0.0f, 0.1f);
```


5.85.60 function diguyViewCamera::get_track_name**Prototype:**

```
const char* get_track_name();
```

Returns:

name of character to be looked at; see [set_track_name\(\)](#)

5.85.61 function diguyViewCamera::set_track_group_name**Prototype:**

```
void set_track_group_name(const char* track_name);
```

Description:

This function sets the name of the group the camera will track.

Arguments:

`track_name` name of group to look at; pass NULL to un-set track group

5.85.62 function diguyViewCamera::get_track_group_name**Prototype:**

```
const char* get_track_group_name();
```

Returns:

name of group to be looked at; see [set_track_group_name\(\)](#)

5.85.63 function diguyViewCamera::set_track_link_name**Prototype:**

```
void set_track_link_name(const char* track_link_name);
```

Description:

This function sets the name of the link to be tracked on the character specified by [set_track_name\(\)](#). See [set_track_name\(\)](#) for more information.

Arguments:

`track_link_name` name of link to look at, or NULL if the base link is to be tracked

5.85.64 function diguyViewCamera::get_track_link_name**Prototype:**

```
const char* get_track_link_name();
```

Returns:

name of link to be looked at; see [set_track_link_name\(\)](#)

5.85.65 function diguyViewCamera::set_track_offset**Prototype:**

```
void set_track_offset(float x, float y, float z);
```

Description:

This function sets the offset of the track position on the character specified by [set_track_name\(\)](#). See [set_track_name\(\)](#) for more information.

Whether the offset is in world coordinates or character-local coordinates is controlled by a call to [set_track_offset_is_in_world_coordinates\(\)](#).

Arguments:

`x, y, z` offset, in meters, of track point

5.85.66 function diguyViewCamera::get_track_offset_x**Prototype:**

```
float get_track_offset_x();
```

Returns:

x component of track offset; see [set_track_offset\(\)](#)

5.85.67 function diguyViewCamera::get_track_offset_y**Prototype:**

```
float get_track_offset_y();
```

Returns:

y component of track offset; see [set_track_offset\(\)](#)

5.85.68 function diguyViewCamera::get_track_offset_z**Prototype:**

```
float get_track_offset_z();
```

Returns:

z component of track offset; see [set_track_offset\(\)](#)

5.85.69 function diguyViewCamera::set_track_offset_is_in_world_coordinates**Prototype:**

```
void set_track_offset_is_in_world_coordinates(int value);
```

Description:

This function sets whether the offset of the track position is character-relative (pass 0) or in world coordinates (pass 1).

Default value is 0.

Arguments:

value	pass 0 for character-relative, 0 for world coordinates

5.85.70 function diguyViewCamera::get_track_offset_is_in_world_coordinates

Prototype:

```
int get_track_offset_is_in_world_coordinates();
```

Returns:

the most recent setting made by
`set_track_offset_is_in_world_coordinates()`.

5.85.71 function diguyViewCamera::set_dolly_name

Prototype:

```
void set_dolly_name(const char* dolly_name);
```

Description:

This function sets the name of the entity this camera should look from, if such an entity can be found. This will cause position values to change over time.

Arguments:

`dolly_name` name of entity to look from, or NULL if no entity is to influence camera position

5.85.72 function diguyViewCamera::get_dolly_name**Prototype:**

```
const char* get_dolly_name();
```

Description:

This function sets the name of the character the camera should look from, if such an character can be found. This will cause position values to change over time.

Arguments:

`dolly_name` name of character to look from, or NULL if no character is to influence camera position

5.85.73 function diguyViewCamera::set_dolly_offset**Prototype:**

```
void set_dolly_offset(float x, float y, float z);
```

Description:

This function sets the offset of the dolly position on the character specified by `set_dolly_name()`. See `set_dolly_name()` for more information.

Whether the offset is in world coordinates or character-local coordinates is controlled by a call to `set_dolly_offset_is_in_world_coordinates()`.

Arguments:

`x, y, z` offset, in meters, of dolly point

5.85.74 function diguyViewCamera::get_dolly_offset_x**Prototype:**

```
float get_dolly_offset_x();
```

Returns:

x component of track offset; see [set_dolly_offset\(\)](#)

5.85.75 function diguyViewCamera::get_dolly_offset_y**Prototype:**

```
float get_dolly_offset_y();
```

Returns:

y component of track offset; see [set_dolly_offset\(\)](#)

5.85.76 function diguyViewCamera::get_dolly_offset_z**Prototype:**

```
float get_dolly_offset_z();
```

Returns:

z component of track offset; see [set_dolly_offset\(\)](#)

5.85.77 function diguyViewCamera::set_dolly_link_name**Prototype:**

```
void set_dolly_link_name(const char* dolly_link_name);
```

Description:

This function sets the name of the link to be dollyed on the character specified by [set_dolly_name\(\)](#).

Arguments:

dolly_link_name name of link to look from, or NULL if the position link is to influence camera position

5.85.78 function diguyViewCamera::get_dolly_link_name**Prototype:**

```
const char* get_dolly_link_name();
```

Returns:

name of link to be looked from; see [set_dolly_link_name\(\)](#)

5.85.79 function diguyViewCamera::set_dolly_offset_is_in_world_coordinates**Prototype:**

```
void set_dolly_offset_is_in_world_coordinates(int value);
```

Description:

This function sets whether the offset of the dolly position is character-relative (pass 0) or in world coordinates (pass 1).

Default value is 0.

Arguments:

<code>value</code>	pass 0 for character-relative, 0 for world coordinates

5.85.80 function
diguyViewCamera::get_dolly_offset_is_in_world_coordinates**Prototype:**

```
int get_dolly_offset_is_in_world_coordinates();
```

Returns:

the most recent setting made by
`set_dolly_offset_is_in_world_coordinates()`.

5.85.81 function diguyViewCamera::get_symbolic_view**Prototype:**

```
int get_symbolic_view();
```

Returns:

if DI-Guy Scenario is rendering simplified 2D representations of characters.

C++ Callback Functions

5.85.82 enumeration

```
enum {
    CALLBACK_ID_SETTINGS_CHANGED = 1
};
```

Description:

This is an enumeration of the different callbacks that can be registered with [add_callback\(\)](#) and [add_callback_script\(\)](#).

Usable From:

- ◇ C++
- ◇ Script

5.85.83 function diguyViewCamera::add_callback

Prototype:

```
int add_callback(int callback_id,
    diguyViewCameraCallback* callback,
    void* callback_params = 0,
    void* callback_user_data = 0);
```

Description:

This function adds a user callback.

Arguments:

callback	pointer to function with prototype diguyViewCameraCallback (typedefed above)
callback_id	integer id of when this callback is to be called
callback_params	struct containing additional parameters needed by some callbacks; actual type depends on callback_id
callback_user_data	

pointer for user's own use; DI-Guy will do nothing to the contents of this pointer beyond passing it back when the callback is invoked

callback_id should be one of the following values:

CALLBACK_ID_SETTINGS_CHANGED

This callback will be called if the current settings of this camera change

Callbacks return a value of type `diguyCallbackReturn`, which will be `DIGUY_CALLBACK_STOP` or `DIGUY_CALLBACK_CONTINUE`. If the callback returns `DIGUY_CALLBACK_STOP`, the default handler of the function will not be called; the callback is asserting that it has done everything necessary for the function call. If the callback returns `DIGUY_CALLBACK_CONTINUE`, the default handler for the function will be called after the callback.

Returns:

0 on success, -1 on failure

5.85.84 function `diguyViewCamera::remove_callback`

Prototype:

```
int remove_callback(int callback_id,
                    diguyViewCameraCallback* callback);
```

Description:

This function removes a user callback. All callbacks matching the specified `callback_id` and callback function will be removed.

Arguments:

<code>callback_id</code>	integer id of when this callback is to be called
<code>callback</code>	pointer to function with prototype <code>diguyViewCameraCallback</code> (typedefed above)

Returns:

0 on success, -1 on failure

5.85.85 function**diguyViewCamera::remove_callback_with_user_data****Prototype:**

```
int remove_callback_with_user_data(int callback_id,  
void* callback_user_data);
```

Description:

This function removes a user callback. All callbacks matching the specified callback_id and callback_user_data pointer will be removed.

Arguments:

callback_id	integer id of when this callback is to be called
callback_user_data	pointer for user's own use

Returns:

0 on success, -1 on failure

C++/Script Callback Functions

5.85.86 function diguyViewCamera::add_callback_script**Prototype:**

```
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function adds a user callback script. Callback scripts can be removed with [remove_callback_script\(\)](#).

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback to be added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

lua specific:

When the script is called, the object for which it is being called will be in the `callback_object` local.

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

5.85.87 function diguyViewCamera::remove_callback_script**Prototype:**

```
int remove_callback_script(int callback_id,
                           const char* callback_script,
                           const char* callback_script_type = NULL);
```

Description:

This function removes a user callback script previously added with [add_callback_script\(\)](#).

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback previously added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script`, all callback scripts whose ids match `callback_id` and whose types match `callback_script_type` will be removed.

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

lua specific:

To pass NULL when calling from a lua script, use nil.

Returns:

0 on success, -1 on failure

Experimental Functions

The following functions are experimental and may change or disappear with no warning.

Unless otherwise noted all are callable from:

- ◆ C++
- ◆ Script

5.85.88 function `diguyViewCamera::set_speed_fov_degrees_per_wheel_click`

Prototype:

```
void set_speed_fov_degrees_per_wheel_click(float fov_change);
```

Description:

Sets number of degrees by which the camera FOV should change whenever the user moves the mouse wheel.

Arguments:

`fov_change` change to FOV in degrees

5.85.89 function

diguyViewCamera::get_speed_fov_degrees_per_wheel_click

Prototype:

```
float get_speed_fov_degrees_per_wheel_click();
```

Description:

Gets the number of degrees by which the camera FOV should change whenever the user moves the mouse wheel.

Returns:

change to FOV, in degrees

5.85.90 function

diguyViewCamera::set_speed_meters_per_second

Prototype:

```
void set_speed_meters_per_second(float speed);
```

Description:

Sets how fast the camera should move in various flyaround modes.

Arguments:

speed	flying speed in meters per second

5.85.91 function diguyViewCamera::get_speed_meters_per_second

Prototype:

```
float get_speed_meters_per_second();
```

Description:

Gets how fast the camera should move in various flyaround modes.

Returns:

speed in meters per second

5.85.92 function diguyViewCamera::set_require_smooth_servo

Prototype:

```
void set_require_smooth_servo(int require);
```

Description:

Sets whether the camera should smoothly servo to its new desired position/heading (as opposed to instantly popping there).

Arguments:

`require` 0 for smooth servoing

5.85.93 function diguyViewCamera::get_require_smooth_servo**Prototype:**

```
int get_require_smooth_servo();
```

Description:

Gets whether the camera should smoothly servo to its new desired position/heading (as opposed to instantly popping there).

Returns:

1 if smooth servoing is required; 0 if not

5.85.94 function diguyViewCamera::set_servo_time_modifier**Prototype:**

```
void set_servo_time_modifier( float modifier );
```

Description:

Sets a factor that determines how smoothly the camera should servo. If a high number (e.g. 10.0), the camera will servo quickly. If a low number (e.g. 1.0), it will servo more slowly.

modifier

5.85.95 function diguyViewCamera::get_servo_time_modifier**Prototype:**

```
float get_servo_time_modifier();
```

Returns:

The servo modifier time that determines how smoothly the camera should servo.

5.85.96 function diguyViewCamera::set_track_camera_avoids_obstacles**Prototype:**

```
void set_track_camera_avoids_obstacles(int avoid);
```

Description:

Sets if the camera mover should make an effort to avoid obstacles that block visibility while a character or group is being tracked.

Arguments:

<code>avoids</code>	true for avoidance

5.85.97 function diguyViewCamera::get_track_camera_avoids_obstacles

Prototype:

```
int get_track_camera_avoids_obstacles();
```

Description:

Gets whether the camera mover should make an effort to avoid obstacles that block visibility while a character or group is being tracked.

Returns:

1 if obstacle avoidance should be attempted

5.85.98 function diguyViewCamera::set_ignore_mouse_drag

Prototype:

```
void set_ignore_mouse_drag(int ignore);
```

Description:

Sets if the camera should not respond to mouse drags. Responding is the default.

Arguments:

avoids	1 to ignore mouse drags; 0 if not

5.85.99 function diguyViewCamera::get_ignore_mouse_drag**Prototype:**

```
int get_ignore_mouse_drag();
```

Description:

Gets whether the camera should ignore mouse drags.

Returns:

1 if mouse drags are to be ignored; 0 if not

5.85.100 function diguyViewCamera::update_frustum**Prototype:**

```
void update_frustum(float* proj_matrix, float* model_matrix);
```

Description:

Allows the diguy camera to build an internal frustum needed for doing character and scene object culling. `projection_matrix` and `model_matrix` should be pointers to 16 floats in OpenGL style matrix format.

This function is not callable from lua.

5.86 diguyViewCameraSettings

diguyViewCameraSettings.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [General Functions](#)
- ◆ [Look At and Look From Functions](#)
- ◆ [Experimental Functions](#)

The [diguyViewCameraSettings](#) class holds camera settings data that can be loaded into a [diguyViewCamera](#) using the [diguyViewCamera::load_settings\(\)](#) call.

The settings of a [diguyViewCameraSettings](#) object cannot be directly applied to a graphics environment viewpoint; the settings must be loaded into a [diguyViewCamera](#) to be applied.

The [diguyViewCameraSettings](#) API closely mirrors the [diguyViewCamera](#) API, with the exception of load, apply, and callback functions.

5.86.1 class diguyViewCameraSettings

```
class BDI_DECLSPEC_diguy diguyViewCameraSettings
{
public:
    const char* get_name();
    const char* get_type_name();
    void set_near_plane(float near_plane);
    float get_near_plane();
    void set_far_plane(float far_plane);
    float get_far_plane();
    void set_fov(float fov);
    float get_fov();
    void set_min_fov(float fov);
    void set_max_fov(float fov);
    float get_min_fov();
    float get_max_fov();
    void set_aspect(float aspect);
    float get_aspect();
    void set_position(float x, float y, float z);
    float get_position_x();
    float get_position_y();
    float get_position_z();
    void set_orientation(float yaw, float roll, float pitch);
    void set_orientation_roll(float r);
    float get_orientation_yaw();
    float get_orientation_roll();
    float get_orientation_pitch();
};
```



```

void set_pitch_lower_limit(float limit);
void set_pitch_upper_limit(float limit);
float get_pitch_lower_limit();
float get_pitch_upper_limit();
int get_pitch_has_limits();
void set_distance(float distance);
float get_distance();
void set_fix(float x, float y, float z);
float get_fix_x();
float get_fix_y();
float get_fix_z();
int set_move_mode(diguyViewCameraMoveMode move_mode);
diguyViewCameraMoveMode get_move_mode();
int set_projection_mode(diguyViewCameraProjectionMode projection_mode);
diguyViewCameraProjectionMode get_projection_mode();
int set_speed(float speed);
float get_speed();
void set_track_name(const char* track_name);
const char* get_track_name();
void set_track_link_name(const char* track_link_name);
const char* get_track_link_name();
void set_track_offset(float x, float y, float z);
float get_track_offset_x();
float get_track_offset_y();
float get_track_offset_z();
void set_track_offset_is_in_world_coordinates(int value);
int get_track_offset_is_in_world_coordinates();
void set_dolly_name(const char* dolly_name);
const char* get_dolly_name();
void set_dolly_link_name(const char* dolly_link_name);
const char* get_dolly_link_name();
void set_dolly_offset(float x, float y, float z);
float get_dolly_offset_x();
float get_dolly_offset_y();
float get_dolly_offset_z();
void set_dolly_offset_is_in_world_coordinates(int value);
int get_dolly_offset_is_in_world_coordinates();
void set_speed_fov_degrees_per_wheel_click(float fov_change);
float get_speed_fov_degrees_per_wheel_click();
void set_speed_meters_per_second(float speed);
float get_speed_meters_per_second();
void set_require_smooth_servo(int require);
int get_require_smooth_servo();
void set_track_camera_avoids_obstacles(int avoid);
int get_track_camera_avoids_obstacles();
void set_ignore_mouse drags(int ignore);
int get_ignore_mouse drags();

```

General Functions

5.86.2 function diguyViewCameraSettings::get_name

Prototype:

```
const char* get_name();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_name()`.

5.86.3 function diguyViewCameraSettings::get_type_name

Prototype:

```
const char* get_type_name();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_type_name()`.

5.86.4 function diguyViewCameraSettings::set_near_plane

Prototype:

```
void set_near_plane(float near_plane);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_near_plane()`.

5.86.5 function diguyViewCameraSettings::get_near_plane

Prototype:

```
float get_near_plane();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_near_plane()`.

5.86.6 function diguyViewCameraSettings::set_far_plane

Prototype:

```
void set_far_plane(float far_plane);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_far_plane()`.

5.86.7 function diguyViewCameraSettings::get_far_plane

Prototype:

```
float get_far_plane();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_far_plane()`.

5.86.8 function diguyViewCameraSettings::set_fov

Prototype:

```
void set_fov(float fov);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_fov()`.

5.86.9 function diguyViewCameraSettings::get_fov

Prototype:

```
float get_fov();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_fov()`.

5.86.10 function diguyViewCameraSettings::set_min_fov

Prototype:

```
void set_min_fov(float fov);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_min_fov()`.

5.86.11 function `diguyViewCameraSettings::set_max_fov`

Prototype:

```
void set_max_fov(float fov);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_max_fov()`.

5.86.12 function `diguyViewCameraSettings::get_min_fov`

Prototype:

```
float get_min_fov();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_min_fov()`.

5.86.13 function diguyViewCameraSettings::get_max_fov**Prototype:**

```
float get_max_fov();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_max_fov()`.

5.86.14 function diguyViewCameraSettings::set_aspect**Prototype:**

```
void set_aspect(float aspect);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_aspect()`.

5.86.15 function diguyViewCameraSettings::get_aspect**Prototype:**

```
float get_aspect();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_aspect()`.

5.86.16 function diguyViewCameraSettings::set_position**Prototype:**

```
void set_position(float x, float y, float z);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_position()`.

5.86.17 function diguyViewCameraSettings::get_position_x**Prototype:**

```
float get_position_x();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_position_x()`.

5.86.18 function diguyViewCameraSettings::get_position_y**Prototype:**

```
float get_position_y();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_position_y()`.

5.86.19 function diguyViewCameraSettings::get_position_z**Prototype:**

```
float get_position_z();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_position_z()`.

5.86.20 function diguyViewCameraSettings::set_orientation**Prototype:**

```
void set_orientation(float yaw, float roll, float pitch);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_orientation()`.

5.86.21 function diguyViewCameraSettings::set_orientation_roll**Prototype:**

```
void set_orientation_roll(float r);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_orientation_roll()`.

5.86.22 function diguyViewCameraSettings::get_orientation_yaw**Prototype:**

```
float get_orientation_yaw();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_orientation_yaw()`.

5.86.23 function diguyViewCameraSettings::get_orientation_roll**Prototype:**

```
float get_orientation_roll();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_orientation_roll()`.

5.86.24 function diguyViewCameraSettings::get_orientation_pitch**Prototype:**

```
float get_orientation_pitch();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_orientation_pitch()`.

5.86.25 function `diguyViewCameraSettings::set_pitch_lower_limit`

Prototype:

```
void set_pitch_lower_limit(float limit);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_pitch_lower_limit()`.

5.86.26 function `diguyViewCameraSettings::set_pitch_upper_limit`

Prototype:

```
void set_pitch_upper_limit(float limit);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_pitch_upper_limit()`.

5.86.27 function `diguyViewCameraSettings::get_pitch_lower_limit`

Prototype:

```
float get_pitch_lower_limit();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_pitch_lower_limit()`.

5.86.28 function `diguyViewCameraSettings::get_pitch_upper_limit`

Prototype:

```
float get_pitch_upper_limit();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_pitch_upper_limit()`.

5.86.29 function diguyViewCameraSettings::get_pitch_has_limits**Prototype:**

```
int get_pitch_has_limits();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_pitch_has_limits()`.

5.86.30 function diguyViewCameraSettings::set_distance**Prototype:**

```
void set_distance(float distance);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_distance()`.

5.86.31 function diguyViewCameraSettings::get_distance**Prototype:**

```
float get_distance();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_distance()`.

5.86.32 function diguyViewCameraSettings::set_fix**Prototype:**

```
void set_fix(float x, float y, float z);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_fix()`.

5.86.33 function diguyViewCameraSettings::get_fix_x**Prototype:**

```
float get_fix_x();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_fix_x()`.

5.86.34 function diguyViewCameraSettings::get_fix_y**Prototype:**

```
float get_fix_y();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_fix_y()`.

5.86.35 function diguyViewCameraSettings::get_fix_z**Prototype:**

```
float get_fix_z();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_fix_z()`.

5.86.36 function diguyViewCameraSettings::set_move_mode**Prototype:**

```
int set_move_mode(diguyViewCameraMoveMode move_mode);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_move_mode()`.

5.86.37 function diguyViewCameraSettings::get_move_mode**Prototype:**

```
diguyViewCameraMoveMode get_move_mode();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_move_mode()`.

5.86.38 function diguyViewCameraSettings::set_projection_mode**Prototype:**

```
int set_projection_mode(diguyViewCameraProjectionMode projection_mode);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_projection_mode()`.

5.86.39 function diguyViewCameraSettings::get_projection_mode

Prototype:

```
diguyViewCameraProjectionMode get_projection_mode();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_projection_mode()`.

5.86.40 function diguyViewCameraSettings::set_speed

Prototype:

```
int set_speed(float speed);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_speed()`.

5.86.41 function diguyViewCameraSettings::get_speed**Prototype:**

```
float get_speed();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_speed()`.

Look At and Look From Functions

5.86.42 function diguyViewCameraSettings::set_track_name**Prototype:**

```
void set_track_name(const char* track_name);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_track_name()`.

5.86.43 function diguyViewCameraSettings::get_track_name**Prototype:**

```
const char* get_track_name();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_track_name()`.

5.86.44 function diguyViewCameraSettings::set_track_link_name**Prototype:**

```
void set_track_link_name(const char* track_link_name);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_track_link_name()`.

5.86.45 function diguyViewCameraSettings::get_track_link_name**Prototype:**

```
const char* get_track_link_name();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_track_link_name()`.

5.86.46 function diguyViewCameraSettings::set_track_offset**Prototype:**

```
void set_track_offset(float x, float y, float z);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_track_offset()`.

5.86.47 function diguyViewCameraSettings::get_track_offset_x**Prototype:**

```
float get_track_offset_x();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_track_offset_x()`.

5.86.48 function diguyViewCameraSettings::get_track_offset_y**Prototype:**

```
float get_track_offset_y();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_track_offset_y()`.

5.86.49 function diguyViewCameraSettings::get_track_offset_z**Prototype:**

```
float get_track_offset_z();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_track_offset_z()`.

5.86.50 function `diguyViewCameraSettings::set_track_offset_is_in_world_coordin`

Prototype:

```
void set_track_offset_is_in_world_coordinates(int value);
```

Description:

The `diguyViewCameraSettings` equivalent of
`diguyViewCamera::set_track_offset_is_in_world_coordinates()`.

5.86.51 function `diguyViewCameraSettings::get_track_offset_is_in_world_coordin`

Prototype:

```
int get_track_offset_is_in_world_coordinates();
```

Description:

The `diguyViewCameraSettings` equivalent of
`diguyViewCamera::get_track_offset_is_in_world_coordinates()`.

5.86.52 function diguyViewCameraSettings::set_dolly_name**Prototype:**

```
void set_dolly_name(const char* dolly_name);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_dolly_name()`.

5.86.53 function diguyViewCameraSettings::get_dolly_name**Prototype:**

```
const char* get_dolly_name();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_dolly_name()`.

5.86.54 function diguyViewCameraSettings::set_dolly_link_name**Prototype:**

```
void set_dolly_link_name(const char* dolly_link_name);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_dolly_link_name()`.

5.86.55 function diguyViewCameraSettings::get_dolly_link_name**Prototype:**

```
const char* get_dolly_link_name();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_dolly_link_name()`.

5.86.56 function diguyViewCameraSettings::set_dolly_offset**Prototype:**

```
void set_dolly_offset(float x, float y, float z);
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::set_dolly_offset()`.

5.86.57 function diguyViewCameraSettings::get_dolly_offset_x**Prototype:**

```
float get_dolly_offset_x();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_dolly_offset_x()`.

5.86.58 function diguyViewCameraSettings::get_dolly_offset_y**Prototype:**

```
float get_dolly_offset_y();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_dolly_offset_y()`.

5.86.59 function diguyViewCameraSettings::get_dolly_offset_z**Prototype:**

```
float get_dolly_offset_z();
```

Description:

The `diguyViewCameraSettings` equivalent of `diguyViewCamera::get_dolly_offset_z()`.

5.86.60 function `diguyViewCameraSettings::set_dolly_offset_is_in_world_coordin`

Prototype:

```
void set_dolly_offset_is_in_world_coordinates(int value);
```

Description:

The `diguyViewCameraSettings` equivalent of
`diguyViewCamera::set_dolly_offset_is_in_world_coordinates()`.

5.86.61 function `diguyViewCameraSettings::get_dolly_offset_is_in_world_coordin`

Prototype:

```
int get_dolly_offset_is_in_world_coordinates();
```

Description:

The `diguyViewCameraSettings` equivalent of
`diguyViewCamera::get_dolly_offset_is_in_world_coordinates()`.

Experimental Functions

The following functions are experimental and may change or disappear with no warning.

5.86.62 function `diguyViewCameraSettings::set_speed_fov_degrees_per_wheel_c`

Prototype:

```
void set_speed_fov_degrees_per_wheel_click(float fov_change);
```

The `diguyViewCameraSettings` equivalent of the `diguyViewCamera` function with the same name.

5.86.63 function `diguyViewCameraSettings::get_speed_fov_degrees_per_wheel_c`

Prototype:

```
float get_speed_fov_degrees_per_wheel_click();
```

The `diguyViewCameraSettings` equivalent of the `diguyViewCamera` function with the same name.

5.86.64 function `diguyViewCameraSettings::set_speed_meters_per_second`

Prototype:

```
void set_speed_meters_per_second(float speed);
```

The `diguyViewCameraSettings` equivalent of the `diguyViewCamera` function with the same name.

5.86.65 function `diguyViewCameraSettings::get_speed_meters_per_second`

Prototype:

```
float get_speed_meters_per_second();
```

The `diguyViewCameraSettings` equivalent of the `diguyViewCamera` function with the same name.

5.86.66 function `diguyViewCameraSettings::set_require_smooth_servo`

Prototype:

```
void set_require_smooth_servo(int require);
```

The `diguyViewCameraSettings` equivalent of the `diguyViewCamera` function with the same name.

5.86.67 function `diguyViewCameraSettings::get_require_smooth_servo`

Prototype:

```
int get_require_smooth_servo();
```

The `diguyViewCameraSettings` equivalent of the `diguyViewCamera` function with the same name.

5.86.68 function `diguyViewCameraSettings::set_track_camera_avoids_obstacles`

Prototype:

```
void set_track_camera_avoids_obstacles(int avoid);
```

The `diguyViewCameraSettings` equivalent of the `diguyViewCamera` function with the same name.

5.86.69 function `diguyViewCameraSettings::get_track_camera_avoids_obstacles`

Prototype:

```
int get_track_camera_avoids_obstacles();
```

The `diguyViewCameraSettings` equivalent of the `diguyViewCamera` function with the same name.

5.86.70 function `diguyViewCameraSettings::set_ignore_mouse_drag`

Prototype:

```
void set_ignore_mouse_drag(int ignore);
```

The `diguyViewCameraSettings` equivalent of the `diguyViewCamera` function with the same name.

5.86.71 function `diguyViewCameraSettings::get_ignore_mouse_drag`s

Prototype:

```
int get_ignore_mouse_drag();
```

The `diguyViewCameraSettings` equivalent of the `diguyViewCamera` function with the same name.

5.87 diguyViewFog

diguyViewFog.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [diguy_typedefs.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [General Functions](#)
- ◆ [Load and Apply Functions](#)
- ◆ [Callback Functions](#)

5.87.1 class diguyViewFog

```
class BDI_DECLSPEC_diguy diguyViewFog
{
public:
    const char* get_name();
    const char* get_type_name();
    void set_density(float density);
    float get_density();
    void set_start(float start);
    float get_start();
    void set_end(float end);
    float get_end();
    void set_color(float r, float g, float b, float a);
    void get_color(float* r, float* g, float* b, float* a);
    void set_fog_mode(int mode);
    int get_fog_mode();
    int load_settings(const char* settings_name);
#ifdef CPLUSPLUS_ONLY
    int load_settings(diguyViewFogSettings* settings);
#endif
    enum {
        CALLBACK_ID_SETTINGS_CHANGED = 1
    };
#ifdef CPLUSPLUS_ONLY
    int add_callback(int callback_id,
        diguyViewFogCallback* callback,
        void* callback_params = 0,
        void* callback_user_data = 0);
    int remove_callback(int callback_id,
        diguyViewFogCallback* callback);
    int remove_callback_with_user_data(int callback_id,
        void* callback_user_data);
#endif
    int add_callback_script(int callback_id,
        const char* callback_script,
        const char* callback_script_type = NULL);
    int remove_callback_script(int callback_id,
        const char* callback_script,
        const char* callback_script_type = NULL);
};
```

General Functions

5.87.2 function diguyViewFog::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the object

Callable From:

- ◇ C++
- ◇ Script

5.87.3 function diguyViewFog::get_type_name

Prototype:

```
const char* get_type_name();
```

Description:

Returns the type name of the object. This pointer will never be NULL.

Returns:

type name of the object

Callable From:

- ◇ C++
- ◇ Script

5.87.4 function diguyViewFog::set_density

Prototype:

```
void set_density(float density);
```

Description:

This sets the density of the fog.

Arguments:

`density` distance of this fog; higher causes more opaque fog

Callable From:

- ◇ C++
- ◇ Script

5.87.5 function diguyViewFog::get_density

Prototype:

```
float get_density();
```

Returns:

density of fog

Callable From:

- ◇ C++
- ◇ Script

5.87.6 function diguyViewFog::set_start

Prototype:

```
void set_start(float start);
```

Description:

This sets the distance at which fog begins to affect the scene. Colors of objects before this distance will not be affected. Colors of objects after this distance will start to be modulated toward to fog color.

Arguments:

start	distance to where fog starts, in meters

Callable From:

- ◇ C++
- ◇ Script

5.87.7 function diguyViewFog::get_start

Prototype:

```
float get_start();
```

Returns:

distance to where fog starts, in meters; see [set_start\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.87.8 function diguyViewFog::set_end

Prototype:

```
void set_end(float end);
```

Description:

This sets the distance at which the fog completely overrides the colors of objects in the scene.

Arguments:

end	distance to where fog ends, in meters

Callable From:

- ◇ C++
- ◇ Script

5.87.9 function diguyViewFog::get_end

Prototype:

```
float get_end();
```

Returns:

distance to where fog ends, in meters; see [set_end\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.87.10 function diguyViewFog::set_color

Prototype:

```
void set_color(float r, float g, float b, float a);
```

Description:

This function sets the color of this fog.

Arguments:

r	red component
g	blue component
b	green component
a	alpha component

Callable From:

- ◇ C++
- ◇ Script

5.87.11 function diguyViewFog::get_color

Prototype:

```
void get_color(float* r, float* g, float* b, float* a);
```

Description:

This function gets the color of this fog. Values are returned via the passed pointers.

Arguments:

r	red component
g	blue component
b	green component
a	alpha component

Callable From:

- ◇ C++
- ◇ Script

5.87.12 function diguyViewFog::set_fog_mode

Prototype:

```
void set_fog_mode(int mode);
```

Description:

This function sets the fog mode.

Arguments:

mode	new mode for fog calculations
DIGUY_FOG_MODE_DISABLED	fog is completely turned off
DIGUY_FOG_MODE_EXP	per-vertex exponential
DIGUY_FOG_MODE_EXP2	per-vertex exponential style 2
DIGUY_FOG_MODE_LINEAR	per-vertex linear

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.87.13 function diguyViewFog::get_fog_mode

Prototype:

```
int get_fog_mode();
```

Returns:

the most recent setting made by [set_fog_mode\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

Load and Apply Functions

5.87.14 function diguyViewFog::load_settings

Prototype:

```
int load_settings(const char* settings_name);
```

Description:

Loads the specified settings into this fog.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.87.15 function diguyViewFog::load_settings**Prototype:**

```
int load_settings(diguyViewFogSettings* settings);
```

Description:

Loads the specified settings into this fog.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Callback Functions

5.87.16 enumeration

```
enum {
    CALLBACK_ID_SETTINGS_CHANGED = 1
};
```

Description:

This is an enumeration of the different callbacks that can be registered with [add_callback\(\)](#) and [add_callback_script\(\)](#).

Usable From:

- ◇ C++
- ◇ Script

5.87.17 function diguyViewFog::add_callback**Prototype:**

```
int add_callback(int callback_id,
                 diguyViewFogCallback* callback,
                 void* callback_params = 0,
                 void* callback_user_data = 0);
```

Description:

This function adds a user callback.

Arguments:

<code>callback</code>	pointer to function with prototype <code>diguyViewFogCallback</code> (typedefed above)
<code>callback_id</code>	integer id of when this callback is to be called
<code>callback_params</code>	struct containing additional parameters needed by some callbacks; actual type depends on <code>callback_id</code>
<code>callback_user_data</code>	pointer for user's own use; DI-Guy will do nothing to the contents of this pointer beyond passing it back when the callback is invoked

`callback_id` should be one of the following values:

CALLBACK_ID_SETTINGS_CHANGED

This callback will be called if the current settings of this fog change

Callbacks return a value of type `diguyCallbackReturn`, which will be `DIGUY_CALLBACK_STOP` or `DIGUY_CALLBACK_CONTINUE`. If the callback returns `DIGUY_CALLBACK_STOP`, the default handler of the function will not be called; the callback is asserting that it has done everything necessary for the function call. If the callback returns `DIGUY_CALLBACK_CONTINUE`, the default handler for the function will be called after the callback.

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.87.18 function diguyViewFog::remove_callback**Prototype:**

```
int remove_callback(int callback_id,  
                    diguyViewFogCallback* callback);
```

Description:

This function removes a user callback. All callbacks matching the specified `callback_id` and callback function will be removed.

Arguments:

<code>callback_id</code>	integer id of when this callback is to be called
<code>callback</code>	pointer to function with prototype <code>diguyViewFogCallback</code> (typedefed above)

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.87.19 function

diguyViewFog::remove_callback_with_user_data

Prototype:

```
int remove_callback_with_user_data(int callback_id,  
void* callback_user_data);
```

Description:

This function removes a user callback. All callbacks matching the specified `callback_id` and `callback_user_data` pointer will be removed.

Arguments:

<code>callback_id</code>	integer id of when this callback is to be called
<code>callback_user_data</code>	pointer for user's own use

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.87.20 function diguyViewFog::add_callback_script**Prototype:**

```
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function adds a user callback script. Callback scripts can be removed with [remove_callback_script\(\)](#).

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback to be added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

lua specific:

When the script is called, the object for which it is being called will be in the `callback_object` global.

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.87.21 function diguyViewFog::remove_callback_script**Prototype:**

```
int remove_callback_script(int callback_id,
                           const char* callback_script,
                           const char* callback_script_type = NULL);
```

Description:

This function removes a user callback script previously added with [add_callback_script\(\)](#).

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback previously added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script`, all callback scripts whose ids match `callback_id` and whose types match `callback_script_type` will be removed.

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

lua specific:

To pass NULL when calling from a lua script, use nil.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.88 diguyViewFogSettings

diguyViewFogSettings.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

The [diguyViewFogSettings](#) class holds camera settings data that can be loaded into a [diguyViewFog](#) using the [diguyViewFog::load_settings\(\)](#) call.

The settings of a [diguyViewFogSettings](#) object cannot be directly applied to a graphics environment; the settings must be loaded into a [diguyViewFog](#) to be applied.

The [diguyViewFogSettings](#) API closely mirrors the [diguyViewFog](#) API, with the exception of load and callback functions.

5.88.1 class diguyViewFogSettings

```
class BDI_DECLSPEC_diguy diguyViewFogSettings
{
public:
    const char* get_name();
    const char* get_type_name();
    void set_density(float density);
    float get_density();
    void set_start(float start);
    float get_start();
    void set_end(float end);
    float get_end();
    void set_color(float r, float g, float b, float a);
    void get_color(float* r, float* g, float* b, float* a);
    void set_fog_mode(int mode);
    int get_fog_mode();
```

5.88.2 function diguyViewFogSettings::get_name

Prototype:

```
const char* get_name();
```

Description:

The `diguyViewLightSettings` equivalent of `diguyViewLight::get_name()`.

5.88.3 function diguyViewFogSettings::get_type_name

Prototype:

```
const char* get_type_name();
```

Description:

The `diguyViewLightSettings` equivalent of `diguyViewLight::get_type_name()`.

5.88.4 function diguyViewFogSettings::set_density

Prototype:

```
void set_density(float density);
```

Description:

The [diguyViewLightSettings](#) equivalent of `diguyViewLight::set_density( )`.

5.88.5 function diguyViewFogSettings::get_density

Prototype:

```
float get_density();
```

Description:

The [diguyViewLightSettings](#) equivalent of `diguyViewLight::get_density( )`.

5.88.6 function diguyViewFogSettings::set_start

Prototype:

```
void set_start(float start);
```

Description:

The [diguyViewLightSettings](#) equivalent of `diguyViewLight::set_start( )`.

5.88.7 function diguyViewFogSettings::get_start

Prototype:

```
float get_start();
```

Description:

The [diguyViewLightSettings](#) equivalent of `diguyViewLight::get_start( )`.

5.88.8 function diguyViewFogSettings::set_end

Prototype:

```
void set_end(float end);
```

Description:

The [diguyViewLightSettings](#) equivalent of `diguyViewLight::set_end( )`.

5.88.9 function diguyViewFogSettings::get_end

Prototype:

```
float get_end();
```

Description:

The [diguyViewLightSettings](#) equivalent of `diguyViewLight::get_end( )`.

5.88.10 function diguyViewFogSettings::set_color

Prototype:

```
void set_color(float r, float g, float b, float a);
```

Description:

The [diguyViewLightSettings](#) equivalent of `diguyViewLight::set_color( )`.

5.88.11 function diguyViewFogSettings::get_color

Prototype:

```
void get_color(float* r, float* g, float* b, float* a);
```

Description:

The [diguyViewLightSettings](#) equivalent of `diguyViewLight::get_color( )`.

5.88.12 function diguyViewFogSettings::set_fog_mode

Prototype:

```
void set_fog_mode(int mode);
```

Description:

The [diguyViewLightSettings](#) equivalent of `diguyViewLight::set_fog_mode( )`.

5.88.13 function diguyViewFogSettings::get_fog_mode

Prototype:

```
int get_fog_mode();
```

Description:

The [diguyViewLightSettings](#) equivalent of `diguyViewLight::get_fog_mode( )`.

5.89 diguyViewLabel

diguyViewLabel.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [diguy_typedefs.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [General Functions](#)
- ◆ [Text Functions](#)
- ◆ [Position Functions](#)
- ◆ [Background Functions](#)
- ◆ [Border Functions](#)
- ◆ [Bar Graph Functions](#)
- ◆ [Misc Functions](#)

5.89.1 typedef diguyViewLabelJustification

```
typedef int diguyViewLabelJustification;
```

5.89.2 enumeration

```
enum {  
    DIGUY_GRAPHICS_LABEL_JUSTIFIED_LEFT    = 0,  
    DIGUY_GRAPHICS_LABEL_JUSTIFIED_BOTTOM = 0,  
    DIGUY_GRAPHICS_LABEL_JUSTIFIED_RIGHT   = 1,  
    DIGUY_GRAPHICS_LABEL_JUSTIFIED_TOP     = 1,  
    DIGUY_GRAPHICS_LABEL_JUSTIFIED_CENTER = 2  
};
```

5.89.3 class diguyViewLabel

```

class BDI_DECLSPEC diguy diguyViewLabel
{
public:
    const char* get_name();
    void set_text(const char* text);
    const char* get_text();
    void set_text_color(float r, float g, float b, float a,
        float interp_time = 0.0f);
    void set_text_shadow_color(float r, float g, float b, float a,
        float interp_time = 0.0f);
    void set_text_shadow_enabled(int text_shadow_enabled);
    void set_text_justification(diguyViewLabelJustification just);
    diguyViewLabelJustification get_text_justification();
    void set_text_offset_x(int offset);
    int get_text_offset_x();
    void set_text_offset_y(int offset);
    int get_text_offset_y();
    void set_x(float screen_x_offset,
        float interp_time = 0.0f);
    float get_x();
    void set_y(float screen_y_offset,
        float interp_time = 0.0f);
    float get_y();
    void set_negative_wraps_x(int wrap);
    int get_negative_wraps_x();
    void set_negative_wraps_y(int wrap);
    int get_negative_wraps_y();
    void set_window_justification_x(diguyViewLabelJustification just);
    diguyViewLabelJustification get_window_justification_x();
    void set_window_justification_y(diguyViewLabelJustification just);
    diguyViewLabelJustification get_window_justification_y();
    void set_minimum_width(int min_x);
    int get_minimum_width();
    void set_minimum_height(int min_y);
    int get_minimum_height();
    int get_width();
    int get_height();
    void set_full_window(int full_window);
    int get_full_window();
    void reset();
    void set_visible(int visible);
    int get_visible();
    void set_background_color(float r, float g, float b, float a,
        float interp_time = 0.0f);
    void set_background_image(const char* filename);
    void set_background_enabled(int background_enabled);
    void set_border_color(float r, float g, float b, float a,
        float interp_time = 0.0f);
    void set_margin_size(int margin_size);
    void set_border_enabled(int border_enabled);
    void enable_bar_graph(int num_lines,
        int lines_are_vertical = 1,
        int line_thickness = 10,
        int space_between_lines = 2,
        int derive_minimum_label_size = 1);
    void disable_bar_graph();
    int get_bar_graph_num_lines();
    int get_bar_graph_lines_are_vertical();

```

```

void set_bar_graph_line_color(int line_number,
    float r, float g, float b, float a);
void set_bar_graph_line_magnitude_ratio(int line_number, float ratio);
void set_bar_graph_line_magnitude_pixels(int line_number, int pixels);
void set_is_temporary(int temporary);
int get_is_temporary();
void set_draw_limit(float draw_limit);
float get_draw_limit();
void set_is_clickable(int is_clickable);
int get_is_clickable();
void set_is_noise(int is_noise);
int get_is_noise();
const char* get_type_name();
enum {
    CALLBACK_ID_USER_SELECTED = 1,
    CALLBACK_ID_USER_UNSELECTED,
    CALLBACK_ID_IGUY_INTERACT
};
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
int remove_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
#ifdef CPLUSPLUS_ONLY
int add_callback(int callback_id,
    diguyViewLabelCallback* callback,
    void* callback_params = NULL,
    void* callback_user_data = NULL);
int remove_callback(int callback_id,
    diguyViewLabelCallback* callback);
int remove_callback_with_user_data(int callback_id,
    void* callback_user_data);

```

General Functions

5.89.4 function diguyViewLabel::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the object

Callable From:

- ◇ C++
- ◇ Script

Text Functions

5.89.5 function diguyViewLabel::set_text

Prototype:

```
void set_text(const char* text);
```

Description:

Sets the text on the label.

Callable From:

- ◇ C++
- ◇ Script

5.89.6 function diguyViewLabel::get_text

Prototype:

```
const char* get_text();
```

Description:

Returns the text on the label.

Returns:

pointer to NULL-terminated string

Callable From:

- ◇ C++
- ◇ Script

5.89.7 function diguyViewLabel::set_text_color

Prototype:

```
void set_text_color(float r, float g, float b, float a,
float interp_time = 0.0f);
```

Description:

Similar to [set_background_color \(\)](#), but sets the color of the label text.

Default color is solid white (1, 1, 1, 1).

Callable From:

- ◇ C++
- ◇ Script

5.89.8 function diguyViewLabel::set_text_shadow_color**Prototype:**

```
void set_text_shadow_color(float r, float g, float b, float a,
    float interp_time = 0.0f);
```

Description:

Similar to [set_background_color \(\)](#), but sets the shadow color of the label text.

Default color is solid black (0, 0, 0, 1).

Callable From:

- ◇ C++
- ◇ Script

5.89.9 function diguyViewLabel::set_text_shadow_enabled**Prototype:**

```
void set_text_shadow_enabled(int text_shadow_enabled);
```

Description:

Enables or disables text shadow. Default is 0 (disabled).

Arguments:

`text_shadow_enabled` pass 1 to enable shadow, 0 to disable

Callable From:

- ◇ C++
- ◇ Script

5.89.10 function diguyViewLabel::set_text_justification**Prototype:**

```
void set_text_justification(diguyViewLabelJustification just);
```

Description:

Sets how the label text is justified, possible values are:

– DIGUY_GRAPHICS_LABEL_JUSTIFIED_LEFT –
 DIGUY_GRAPHICS_LABEL_JUSTIFIED_RIGHT –
 DIGUY_GRAPHICS_LABEL_JUSTIFIED_CENTER

Depending on how the flag is set the text will either draw flowing to the right of the current x. The left of the current x or centered above x.

Callable From:

- ◇ C++
- ◇ Script

5.89.11 function diguyViewLabel::get_text_justification**Prototype:**

```
diguyViewLabelJustification get_text_justification();
```

Description:

Returns how the text on the label is being justified.

Returns:

One of the following values:

- ◇ DIGUY_GRAPHICS_LABEL_JUSTIFIED_LEFT
- ◇ DIGUY_GRAPHICS_LABEL_JUSTIFIED_RIGHT
- ◇ DIGUY_GRAPHICS_LABEL_JUSTIFIED_CENTER

Callable From:

- ◇ C++
- ◇ Script

5.89.12 function diguyViewLabel::set_text_offset_x**Prototype:**

```
void set_text_offset_x(int offset);
```

Description:

Adds an x offset to the text part of the label.

Callable From:

- ◇ C++
- ◇ Script

5.89.13 function diguyViewLabel::get_text_offset_x**Prototype:**

```
int get_text_offset_x();
```

Returns:

returns the x offset of the text part of the label.

Callable From:

- ◇ C++
- ◇ Script

5.89.14 function diguyViewLabel::set_text_offset_y**Prototype:**

```
void set_text_offset_y(int offset);
```

Description:

Adds a y offset to the text part of the label.

Callable From:

- ◇ C++
- ◇ Script

5.89.15 function diguyViewLabel::get_text_offset_y**Prototype:**

```
int get_text_offset_y();
```

Returns:

returns the y offset of the text part of the label.

Callable From:

- ◇ C++
- ◇ Script

Position Functions

5.89.16 function diguyViewLabel::set_x

Prototype:

```
void set_x(float screen_x_offset,
           float interp_time = 0.0f);
```

Description:

Sets the x location of the label. Positive numbers are relative to the left side of the screen; negative numbers are relative to the right side of the screen.

The optional `interp_time` argument will make the label animate to its new location.

Arguments:

<code>screen_x_offset</code>	x offset in pixels
<code>interp_time</code>	amount of time in seconds used to shift to new position

Callable From:

◇ C++
◇ Script

5.89.17 function diguyViewLabel::get_x

Prototype:

```
float get_x();
```

Description:

Gets the x location of the label.

Note: this will return the current x value, not the desired one.

Callable From:

◇ C++
◇ Script

5.89.18 function diguyViewLabel::set_y**Prototype:**

```
void set_y(float screen_y_offset,
           float interp_time = 0.0f);
```

Description:

Sets the y location of the label. Positive numbers are relative to the bottom of the screen; negative numbers are relative to the top of the screen.

The optional `interp_time` argument will make the label animate to its new location.

Arguments:

<code>screen_y_offset</code>	y offset in pixels
<code>interp_time</code>	amount of time in seconds used to shift to new position

Callable From:

- ◇ C++
- ◇ Script

5.89.19 function diguyViewLabel::get_y**Prototype:**

```
float get_y();
```

Description:

Gets the y location of the label.

Note: this will return the current y value, not the desired one.

Callable From:

- ◇ C++
- ◇ Script

5.89.20 function diguyViewLabel::set_negative_wraps_x**Prototype:**

```
void set_negative_wraps_x(int wrap);
```

Description:

Sets if having a negative value wraps to the other side of the viewport.
Defaults to 1.

Arguments:

wrap	1 if x should wrap; 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.89.21 function diguyViewLabel::get_negative_wraps_x**Prototype:**

```
int get_negative_wraps_x();
```

Returns:

1 if x wraps its offset when negative

Callable From:

- ◇ C++
- ◇ Script

5.89.22 function diguyViewLabel::set_negative_wraps_y**Prototype:**

```
void set_negative_wraps_y(int wrap);
```

Description:

Sets if having a negative value wraps to the other side of the viewport.
Defaults to 1.

Arguments:

wrap	1 if y should wrap; 0 if not

Callable From:

- ◇ C++
- ◇ Script

5.89.23 function diguyViewLabel::get_negative_wraps_y**Prototype:**

```
int get_negative_wraps_y();
```

Returns:

1 if y wraps its offset when negative

Callable From:

- ◇ C++
- ◇ Script

5.89.24 function diguyViewLabel::set_window_justification_x**Prototype:**

```
void set_window_justification_x(diguyViewLabelJustification just);
```

Description:

Sets how the label is positioned relative to the window, possible values are:

- ◇ DIGUY_GRAPHICS_LABEL_JUSTIFIED_LEFT
- ◇ DIGUY_GRAPHICS_LABEL_JUSTIFIED_RIGHT
- ◇ DIGUY_GRAPHICS_LABEL_JUSTIFIED_CENTER

Depending on how the flag is set the x offset will be measured from the right of the screen, the left or from the center of the window.

Callable From:

- ◇ C++
- ◇ Script

5.89.25 function diguyViewLabel::get_window_justification_x**Prototype:**

```
diguyViewLabelJustification get_window_justification_x();
```

Returns:

label's x positioning parameter described above

Callable From:

- ◇ C++
- ◇ Script

5.89.26 function diguyViewLabel::set_window_justification_y**Prototype:**

```
void set_window_justification_y(diguyViewLabelJustification just);
```

Description:

Sets how the label is positioned relative to the window. Possible values are:

- ◇ DIGUY_GRAPHICS_LABEL_JUSTIFIED_BOTTOM
- ◇ DIGUY_GRAPHICS_LABEL_JUSTIFIED_TOP
- ◇ DIGUY_GRAPHICS_LABEL_JUSTIFIED_CENTER

Depending on how the flag is set the y offset will be measured from the bottom of the screen, from the top or from the center of the window.

Callable From:

- ◇ C++
- ◇ Script

5.89.27 function diguyViewLabel::get_window_justification_y**Prototype:**

```
diguyViewLabelJustification get_window_justification_y();
```

Returns:

label's y positioning parameter described above

Callable From:

- ◇ C++
- ◇ Script

5.89.28 function diguyViewLabel::set_minimum_width**Prototype:**

```
void set_minimum_width(int min_x);
```

Description:

Sets the minimum width of the label. The default is 0, which will size the label width to fit the label text or bar graph.

Arguments:

<code>min_x</code>	minimum width, in pixels

Callable From:

- ◇ C++
- ◇ Script

5.89.29 function diguyViewLabel::get_minimum_width**Prototype:**

```
int get_minimum_width();
```

Returns:

minimum width of the label, as set by `set_minimum_width()`

Callable From:

- ◇ C++
- ◇ Script

5.89.30 function diguyViewLabel::set_minimum_height**Prototype:**

```
void set_minimum_height(int min_y);
```

Description:

Similar to `set_minimum_width()`, but sets minimum label height.

5.89.31 function diguyViewLabel::get_minimum_height**Prototype:**

```
int get_minimum_height();
```

Description:

Similar to `set_minimum_height()`, but returns minimum label height.

5.89.32 function diguyViewLabel::get_width**Prototype:**

```
int get_width();
```

Returns:

the current label width

Callable From:

- ◇ C++
- ◇ Script

5.89.33 function diguyViewLabel::get_height**Prototype:**

```
int get_height();
```

Returns:

the current label height

Callable From:

- ◇ C++
- ◇ Script

5.89.34 function diguyViewLabel::set_full_window**Prototype:**

```
void set_full_window(int full_window);
```

Description:

Turns the label into a full-window overlay. This can be useful for things like Heads-Up Displays (HUDs).

If there is a background image (see [set_background_image\(\)](#)), the image will be scaled to fit the entire screen.

Default is 0 (not full-window). Note: turning a label into a full window will automatically make it non-clickable.

Arguments:

`full_window` pass 1 for full-window, 0 for not full-window

Callable From:

◇ C++
◇ Script

5.89.35 function diguyViewLabel::get_full_window

Prototype:

```
int get_full_window();
```

Returns:

whether label is full-window, as set by [set_full_window\(\)](#)

Callable From:

◇ C++
◇ Script

5.89.36 function diguyViewLabel::reset

Prototype:

```
void reset();
```

Description:

Resets the parameters of the label to the values from the constructor.

Callable From:

- ◇ C++
- ◇ Script

5.89.37 function diguyViewLabel::set_visible

Prototype:

```
void set_visible(int visible);
```

Description:

Sets the visibility of a label. Default is 1 (visible).

Arguments:

`visible` pass 1 to make label visible, 0 for invisible

Callable From:

- ◇ C++
- ◇ Script

5.89.38 function diguyViewLabel::get_visible**Prototype:**

```
int get_visible();
```

Returns:

visibility of label, as set by `set_visible()`

Callable From:

- ◇ C++
- ◇ Script

Background Functions

5.89.39 function diguyViewLabel::set_background_color**Prototype:**

```
void set_background_color(float r, float g, float b, float a,  
    float interp_time = 0.0f);
```

Description:

Sets the color of the label background.

The optional `interp_time` argument will make the label gradually shift to its new color.

Arguments:

`r, g, b, a` RGBA values of new color, between 0.0 and 1.0

`interp_time` amount of time in seconds used to shift to new color

Default color is translucent blue (0, 0, 0.3, 0.7).

Callable From:

◇ C++

◇ Script

5.89.40 function diguyViewLabel::set_background_image**Prototype:**

```
void set_background_image(const char* filename);
```

Description:

Sets the background image of the label. The file will be looked for in \$DIGUY*customgeometry*/rgb first, then in \$DIGUY*geometry*rgb.

The default is no background image.

Callable From:

- ◇ C++
- ◇ Script

5.89.41 function diguyViewLabel::set_background_enabled**Prototype:**

```
void set_background_enabled(int background_enabled);
```

Description:

Enables or disables label background. Default is 1 (enabled).

Arguments:

`background_enabled` pass 1 to enable shadow, 0 to disable

Callable From:

- ◇ C++
- ◇ Script

Border Functions

5.89.42 function diguyViewLabel::set_border_color

Prototype:

```
void set_border_color(float r, float g, float b, float a,
    float interp_time = 0.0f);
```

Description:

Similar to [set_border_color\(\)](#), but sets the border color of the label background.

Default color is translucent white (1, 1, 1, 0.7).

r, g, b, a RGBA values of new color, between 0.0 and 1.0
interp_time amount of time in seconds used to shift to new color

Callable From:

- ◇ C++
- ◇ Script

5.89.43 function diguyViewLabel::set_margin_size**Prototype:**

```
void set_margin_size(int margin_size);
```

Description:

Sets the margin, of the label. Default is 6 pixels.

Arguments:

margin_size margin size in pixels

Callable From:

◇ C++

◇ Script

5.89.44 function diguyViewLabel::set_border_enabled**Prototype:**

```
void set_border_enabled(int border_enabled);
```

Description:

Sets if the border should be visible. Default is 1 (enabled).

Arguments:

border_enabled pass 1 to enable, 0 to disable

Callable From:

◇ C++

◇ Script

Bar Graph Functions

These functions allow view labels to be used as bar graph-like visuals. This can be useful for providing a graphical representation for various parameters in a scenario; for example, to show the level of different control parameters of a DI-Guy AI Agent.

Lua Example:

```
local label_bar_graph = this_scenario:find_or_create_label("bar_graph");

-- set position and minimum sizes
label_bar_graph:set_x(10.0);
label_bar_graph:set_y(20.0);
label_bar_graph:set_minimum_width(10.0);
label_bar_graph:set_minimum_height(100.0);

-- enable bar graph functionality:
-- 4 bars/lines
-- lines are vertical
-- lines are 8 pixels wide
-- 4 pixels between lines
label_bar_graph:enable_bar_graph(4, 1, 10, 4);

-- set bar graph line colors
label_bar_graph:set_bar_graph_line_color(0, 1.0, 0.0, 0.0, 1.0); -- red
label_bar_graph:set_bar_graph_line_color(1, 0.0, 1.0, 0.0, 1.0); -- green
label_bar_graph:set_bar_graph_line_color(2, 0.0, 0.0, 1.0, 1.0); -- blue
label_bar_graph:set_bar_graph_line_color(3, 1.0, 1.0, 0.0, 1.0); -- yellow

-- set bar graph line magnitudes (height in this case)
label_bar_graph:set_bar_graph_line_magnitude_ratio(0, 1.0);
label_bar_graph:set_bar_graph_line_magnitude_ratio(1, 0.25);
label_bar_graph:set_bar_graph_line_magnitude_ratio(2, 0.5);
label_bar_graph:set_bar_graph_line_magnitude_ratio(3, 1.0);
```

5.89.45 function diguyViewLabel::enable_bar_graph**Prototype:**

```
void enable_bar_graph(int num_lines,
    int lines_are_vertical = 1,
    int line_thickness = 10,
    int space_between_lines = 2,
    int derive_minimum_label_size = 1);
```

Description:

Enables bar graph functionality for this label. Note that this doesn't automatically turn off text or background image.

Label position and minimum sizes should be set before this is called.

Arguments:

<code>num_lines</code>	number of lines/bars that will be shown
<code>lines_are_vertical</code>	pass 1 for lines to go bottom to top, 0 for lines to go left to right
<code>line_thickness</code>	line thickness, in pixels
<code>space_between_lines</code>	space between lines, in pixels
<code>derive_minimum_label_size</code>	derive and set the minimum label size, as if <code>set_minimum_width()</code> and <code>set_minimum_height()</code> were called to fit this bar graph

Callable From:

- ◇ C++
- ◇ Script

5.89.46 function diguyViewLabel::disable_bar_graph**Prototype:**

```
void disable_bar_graph();
```

Description:

Disables bar graph functionality for this label. Bar graph lines will no longer be visible.

5.89.47 function diguyViewLabel::get_bar_graph_num_lines**Prototype:**

```
int get_bar_graph_num_lines();
```

Returns:

number of bars/lines in bar graph

5.89.48 function
diguyViewLabel::get_bar_graph_lines_are_vertical**Prototype:**

```
int get_bar_graph_lines_are_vertical();
```

Returns:

1 if lines to go bottom to top, 0 if lines to go left to right

5.89.49 function diguyViewLabel::set_bar_graph_line_color**Prototype:**

```
void set_bar_graph_line_color(int line_number,  
float r, float g, float b, float a);
```

Description:

Sets the color of the specified line.

5.89.50 function

diguyViewLabel::set_bar_graph_line_magnitude_ratio

Prototype:

```
void set_bar_graph_line_magnitude_ratio(int line_number, float ratio);
```

Description:

Sets how much of the usable space (bottom to top for vertical lines, left to right for horizontal lines) a line will take up at its maximum length.

Arguments:

`line_number` line to change

`ratio` how much of the line to display, from 0 (none) to 1 (all)

For example, if, based on the label height, the bar graph has at most 20 pixels to show each line, passing 0.5 for ratio will fill 10 pixels with the line color, while passing 1.0 for ratio will fill all 20 pixels.

5.89.51 function

diguyViewLabel::set_bar_graph_line_magnitude_pixels

Prototype:

```
void set_bar_graph_line_magnitude_pixels(int line_number, int pixels);
```

Description:

Similar to `set_bar_graph_line_magnitude_ratio()`, but sets the length of the line based on absolute pixels.

Note that care should be taken to not overrun the current height or width of the label when using this function.

Arguments:

`line_number` line to change

`pixels` line length in pixels

Misc Functions

5.89.52 function diguyViewLabel::set_is_temporary

Prototype:

```
void set_is_temporary(int temporary);
```

Description:

Sets if a label is temporary. Temporary labels are destroyed on scenario reset, and will not re-appear unless they are recreated.

Pass 1 to make label temporary, 0 to make it permanent.

5.89.53 function diguyViewLabel::get_is_temporary**Prototype:**

```
int get_is_temporary();
```

Returns:

1 if a label is temporary, 0 if not; see [set_is_temporary\(\)](#)

5.89.54 function diguyViewLabel::set_draw_limit**Prototype:**

```
void set_draw_limit(float draw_limit);
```

Description:

Sets how far away the label will be drawn.

Arguments:

`draw_limit` -1 to always draw, otherwise will draw till `draw_limit` is reached

Callable From:

- ◇ C++
- ◇ Script

5.89.55 function diguyViewLabel::get_draw_limit**Prototype:**

```
float get_draw_limit();
```

Description:

Gets how far away the label will be drawn. Defaults to -1 which always draws.

Callable From:

- ◇ C++
- ◇ Script

5.89.56 function diguyViewLabel::set_is_clickable**Prototype:**

```
void set_is_clickable(int is_clickable);
```

Description:

Sets if the label is clickable with the mouse.

For character labels this is the equivalent of clicking on the character itself, and can trigger the CALLBACK_ID_IGUY_INTERACT callback.

For screen labels it triggers CALLBACK_ID_USER_SELECTED on mouse down, and CALLBACK_ID_USER_UNSELECTED on mouse up.

By default labels are clickable.

Arguments:

`is_clickable` pass 1 to enable, 0 to disable

Callable From:

- ◇ C++
- ◇ Script

5.89.57 function diguyViewLabel::get_is_clickable

Prototype:

```
int get_is_clickable();
```

Returns:

1 if label responds to mouse clicks, 0 if not; see [set_is_clickable\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.89.58 function diguyViewLabel::set_is_noise**Prototype:**

```
void set_is_noise(int is_noise);
```

Description:

Sets if the label is "noise".

The textures of noise labels will be shifted every frame.

By default labels are not noise.

Arguments:

`is_noise` pass 1 to enable, 0 to disable

Callable From:

- ◇ C++
- ◇ Script

5.89.59 function diguyViewLabel::get_is_noise**Prototype:**

```
int get_is_noise();
```

Returns:

1 if label textures shift every frame, 0 if not; see [set_is_noise\(\)](#)

Callable From:

- ◇ C++
- ◇ Script

5.89.60 function diguyViewLabel::get_type_name**Prototype:**

```
const char* get_type_name();
```

Resturs:

NULL-terminated string "label"

5.89.61 enumeration

```
enum {
    CALLBACK_ID_USER_SELECTED = 1,
    CALLBACK_ID_USER_UNSELECTED,
    CALLBACK_ID_IGUY_INTERACT
};
```

Description:

This is an enumeration of the different callbacks that can be registered with [add_callback\(\)](#) and [add_callback_script\(\)](#).

Usable From:

- ◇ C++
- ◇ Script

5.89.62 function diguyViewLabel::add_callback_script**Prototype:**

```
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function adds a user callback script. Callback scripts can be removed with [remove_callback_script\(\)](#).

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback to be added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

lua specific:

When the script is called, the object for which it is being called will be in the `callback_object` global.

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

lua Example:

```
-----
local on_clicked_callback = [[
    local la_name = callback_object:get_name();
    bdi_log_print(BDI_LOG_WARN, "Calling on_click for label ".. la_name .. "'.\n");
]]
-----

label:add_callback_script(diguyViewLabel_CALLBACK_ID_USER_UNSELECTED,
    on_clicked_callback,
```

```
"lua");
```

5.89.63 function diguyViewLabel::remove_callback_script

Prototype:

```
int remove_callback_script(int callback_id,  
    const char* callback_script,  
    const char* callback_script_type = NULL);
```

Description:

This function removes a user callback script previously added with [add_callback_script\(\)](#).

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback previously added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script`, all callback scripts whose ids match `callback_id` and whose types match `callback_script_type` will be removed.

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

lua specific:

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Lua Example:

```
--
-- Remove all callback scripts with id
-- diguyViewLabel_CALLBACK_ID_USER_UNSELECTED
--
label:remove_callback_script(diguyViewLabel_CALLBACK_ID_USER_UNSELECTED,
nil,
"lua");
```

5.89.64 function diguyViewLabel::add_callback

Prototype:

```
int add_callback(int callback_id,
    diguyViewLabelCallback* callback,
    void* callback_params = NULL,
    void* callback_user_data = NULL);
```

Description:

This function adds a user callback. Callbacks can be removed with [remove_callback\(\)](#) or [remove_callback_with_user_data\(\)](#).

Arguments:

<code>callback</code>	pointer to function with prototype <code>diguyViewLabelCallback</code> (typedefed above)
<code>callback_id</code>	integer id of when this callback is to be called
<code>callback_params</code>	not currently used; pass NULL
<code>callback_user_data</code>	pointer for user's own use; DI-Guy will do nothing to the contents of this pointer beyond

passing it back when the callback is invoked

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.89.65 function diguyViewLabel::remove_callback

Prototype:

```
int remove_callback(int callback_id,
                    diguyViewLabelCallback* callback);
```

Description:

This function removes a user callback. All callbacks matching the specified callback_id and callback function will be removed.

Arguments:

callback_id	integer id of when this callback is to be called
callback	pointer to function with prototype diguyViewLabelCallback (typedefed above)

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

Description:

This function removes a user callback. All callbacks matching the specified `callback_id` and `callback_user_data` pointer will be removed.

Arguments:

`callback_id` integer id of when this callback is to be called
`callback_user_data` pointer for user's own use

Returns:

0 on success, -1 on failure

Callable From:

◆ C++

5.89.66 function

`diguyViewLabel::remove_callback_with_user_data`

Prototype:

```
int remove_callback_with_user_data(int callback_id,
void* callback_user_data);
```


5.90 diguyViewLight

diguyViewLight.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [diguy_typedefs.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ General Functions
- ◆ Load and Apply Functions
- ◆ Callback Functions
- ◆ Experimental Functions

5.90.1 class diguyViewLight

```
class BDI_DECLSPEC_diguy diguyViewLight
{
public:
    const char* get_name();
    const char* get_type_name();
    void set_direction(float x, float y, float z);
    void get_direction(float* x, float* y, float* z);
    void set_position(float x, float y, float z);
    void get_position(float* x, float* y, float* z);
    void set_ambient_color(float r, float g, float b, float a);
    void get_ambient_color(float* r, float* g, float* b, float* a);
    void set_diffuse_color(float r, float g, float b, float a);
    void get_diffuse_color(float* r, float* g, float* b, float* a);
    void set_specular_color(float r, float g, float b, float a);
    void get_specular_color(float* r, float* g, float* b, float* a);
    int load_settings(const char* settings_name);
#ifdef CPLUSPLUS_ONLY
    int load_settings(diguyViewLightSettings* settings);
#endif
    void apply(int i = 0);
    enum {
        CALLBACK_ID_SETTINGS_CHANGED = 1
    };
#ifdef CPLUSPLUS_ONLY
    int add_callback(int callback_id,
        diguyViewLightCallback* callback,
        void* callback_params = 0,
        void* callback_user_data = 0);
    int remove_callback(int callback_id,
        diguyViewLightCallback* callback);
    int remove_callback_with_user_data(int callback_id,
        void* callback_user_data);
#endif
    int add_callback_script(int callback_id,
        const char* callback_script,
        const char* callback_script_type = NULL);
    int remove_callback_script(int callback_id,
```

```

    const char* callback_script,
    const char* callback_script_type = NULL);
void set_near_plane(float near_plane);
float get_near_plane();
void set_far_plane(float far_plane);
float get_far_plane();
void set_casts_shadows(int casts_shadows);
int get_casts_shadows();

int set_scene_objects_cast_shadows(int value);
int get_scene_objects_cast_shadows();
int set_brightness_multiplier(float multiplier);
float get_brightness_multiplier();
void set_shadow_volume_size(float volume);
float get_shadow_volume_size();
void set_shadow_offset_multiplier(float multiplier);
float get_shadow_offset_multiplier();
void set_spotlight_angle(float angle);
float get_spotlight_angle();
void set_texture_name(const char* name);
void set_active(int active);
int get_active();
int set_shadow_map_update_paused(int pause);
int get_shadow_map_update_paused();
int set_shadow_map_update_frequency(int frame_frequency);
int get_shadow_map_update_frequency();

```

General Functions

5.90.2 function diguyViewLight::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the object

Callable From:

- ◇ C++
- ◇ Script

5.90.3 function diguyViewLight::get_type_name

Prototype:

```
const char* get_type_name();
```

Description:

Returns the type name of the object. This pointer will never be NULL.

Returns:

type name of the object

Callable From:

- ◇ C++
- ◇ Script

5.90.4 function diguyViewLight::set_direction

Prototype:

```
void set_direction(float x, float y, float z);
```

Description:

This function sets the direction of this light.

Arguments:

x	x direction, in meters
y	y direction, in meters
z	z direction, in meters

Callable From:

- ◇ C++
- ◇ Script

5.90.5 function diguyViewLight::get_direction

Prototype:

```
void get_direction(float* x, float* y, float* z);
```

Description:

This function gets the direction of this light. Values are returned via the passed pointers.

Arguments:

x	x direction, in meters
y	y direction, in meters
z	z direction, in meters

Callable From:

- ◇ C++
- ◇ Script

5.90.6 function diguyViewLight::set_position

Prototype:

```
void set_position(float x, float y, float z);
```

Description:

This function sets the position of this light.

Arguments:

x	x position, in meters
y	y position, in meters
z	z position, in meters

Callable From:

- ◇ C++
- ◇ Script

5.90.7 function diguyViewLight::get_position

Prototype:

```
void get_position(float* x, float* y, float* z);
```

Description:

This function gets the position of this light. Values are returned via the passed pointers.

Arguments:

x	x position, in meters
y	y position, in meters
z	z position, in meters

Callable From:

- ◇ C++
- ◇ Script

5.90.8 function diguyViewLight::set_ambient_color

Prototype:

```
void set_ambient_color(float r, float g, float b, float a);
```

Description:

This function sets the ambient color.

Arguments:

r	red component
g	blue component
b	green component
a	alpha component

Callable From:

- ◇ C++
- ◇ Script

5.90.9 function diguyViewLight::get_ambient_color

Prototype:

```
void get_ambient_color(float* r, float* g, float* b, float* a);
```

Description:

This function gets the ambient color of this light. Values are returned via the passed pointers.

Arguments:

r	red component
g	blue component
b	green component
a	alpha component

Callable From:

- ◇ C++

◇ Script

5.90.10 function diguyViewLight::set_diffuse_color

Prototype:

```
void set_diffuse_color(float r, float g, float b, float a);
```

Description:

This function sets the diffuse color.

Arguments:

r	red component
g	blue component
b	green component
a	alpha component

Callable From:

◇ C++
◇ Script

5.90.11 function diguyViewLight::get_diffuse_color**Prototype:**

```
void get_diffuse_color(float* r, float* g, float* b, float* a);
```

Description:

This function gets the diffuse color of this light. Values are returned via the passed pointers.

Arguments:

r	red component
g	blue component
b	green component
a	alpha component

Callable From:

- ◇ C++
- ◇ Script

5.90.12 function diguyViewLight::set_specular_color**Prototype:**

```
void set_specular_color(float r, float g, float b, float a);
```

Description:

This function sets the specular color.

Arguments:

r	red component
g	blue component
b	green component
a	alpha component

Callable From:

- ◇ C++
- ◇ Script

5.90.13 function diguyViewLight::get_specular_color**Prototype:**

```
void get_specular_color(float* r, float* g, float* b, float* a);
```

Description:

This function gets the specular color of this light. Values are returned via the passed pointers.

Arguments:

r	red component
g	blue component
b	green component
a	alpha component

Callable From:

- ◇ C++
- ◇ Script

Load and Apply Functions

5.90.14 function diguyViewLight::load_settings

Prototype:

```
int load_settings(const char* settings_name);
```

Description:

Loads the specified settings into this light.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.90.15 function diguyViewLight::load_settings

Prototype:

```
int load_settings(diguyViewLightSettings* settings);
```

Description:

Loads the specified settings into this light.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.90.16 function diguyViewLight::apply

Prototype:

```
void apply(int i = 0);
```

Description:

Applies the light to OpenGL state. Currently DI-Guy only supports one light in it's default OpenGL implementation.

At least one light must be applied for advanced DI-Guy reference shaders to work.

Callback Functions

5.90.17 enumeration

```
enum {
    CALLBACK_ID_SETTINGS_CHANGED = 1
};
```

Description:

This is an enumeration of the different callbacks that can be registered with [add_callback\(\)](#) and [add_callback_script\(\)](#).

Usable From:

- ◇ C++
- ◇ Script

5.90.18 function diguyViewLight::add_callback**Prototype:**

```
int add_callback(int callback_id,
                 diguyViewLightCallback* callback,
                 void* callback_params = 0,
                 void* callback_user_data = 0);
```

Description:

This function adds a user callback.

Arguments:

<code>callback</code>	pointer to function with prototype <code>diguyViewLightCallback</code> (typedefed above)
<code>callback_id</code>	integer id of when this callback is to be called
<code>callback_params</code>	struct containing additional parameters needed by some callbacks; actual type depends on <code>callback_id</code>
<code>callback_user_data</code>	pointer for user's own use; DI-Guy will do nothing to the contents of this pointer beyond passing it back when the callback is invoked

`callback_id` should be one of the following values:

CALLBACK_ID_SETTINGS_CHANGED

This callback will be called if the current settings of this light change

Callbacks return a value of type `diguyCallbackReturn`, which will be `DIGUY_CALLBACK_STOP` or `DIGUY_CALLBACK_CONTINUE`. If the callback returns `DIGUY_CALLBACK_STOP`, the default handler of the function will not be called; the callback is asserting that it has done everything necessary for the function call. If the callback returns `DIGUY_CALLBACK_CONTINUE`, the default handler for the function will be called after the callback.

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.90.19 function diguyViewLight::remove_callback**Prototype:**

```
int remove_callback(int callback_id,  
                    diguyViewLightCallback* callback);
```

Description:

This function removes a user callback. All callbacks matching the specified `callback_id` and callback function will be removed.

Arguments:

<code>callback_id</code>	integer id of when this callback is to be called
<code>callback</code>	pointer to function with prototype diguyViewLightCallback (typedefed above)

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.90.20 function**diguyViewLight::remove_callback_with_user_data****Prototype:**

```
int remove_callback_with_user_data(int callback_id,  
void* callback_user_data);
```

Description:

This function removes a user callback. All callbacks matching the specified callback_id and callback_user_data pointer will be removed.

Arguments:

callback_id	integer id of when this callback is to be called
callback_user_data	pointer for user's own use

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.90.21 function diguyViewLight::add_callback_script**Prototype:**

```
int add_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function adds a user callback script. Callback scripts can be removed with [remove_callback_script\(\)](#).

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback to be added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

lua specific:

When the script is called, the object for which it is being called will be in the `callback_object` global.

To pass NULL when calling from a lua script, use `nil`.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.90.22 function diguyViewLight::remove_callback_script**Prototype:**

```
int remove_callback_script(int callback_id,
    const char* callback_script,
    const char* callback_script_type = NULL);
```

Description:

This function removes a user callback script previously added with [add_callback_script\(\)](#).

Arguments:

<code>callback_id</code>	integer id of the callback
<code>callback_script</code>	script text of callback previously added
<code>callback_script_type</code>	the type of script contained in <code>callback_script</code>

If NULL is passed for `callback_script`, all callback scripts whose ids match `callback_id` and whose types match `callback_script_type` will be removed.

If NULL is passed for `callback_script_type`, a default script type will be derived based on the default script interpreter of the scenario.

lua specific:

To pass NULL when calling from a lua script, use nil.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

Experimental Functions

The following functions are experimental and may change or disappear with no warning.

5.90.23 function diguyViewLight::set_near_plane

Prototype:

```
void set_near_plane(float near_plane);
```

5.90.24 function diguyViewLight::get_near_plane**Prototype:**

```
float get_near_plane();
```

5.90.25 function diguyViewLight::set_far_plane**Prototype:**

```
void set_far_plane(float far_plane);
```

5.90.26 function diguyViewLight::get_far_plane**Prototype:**

```
float get_far_plane();
```

5.90.27 function diguyViewLight::set_casts_shadows**Prototype:**

```
void set_casts_shadows(int casts_shadows);
```

Description:

Sets if a light will cast shadow; this will only work in DI-Guy Scenario.

5.90.28 function diguyViewLight::get_casts_shadows**Prototype:**

```
int get_casts_shadows();
```

Returns:

Returns if a light will cast shadow; this is limited to DI-Guy Scenario

**5.90.29 function
diguyViewLight::set_scene_objects_cast_shadows****Prototype:**

```
int set_scene_objects_cast_shadows(int value);
```

5.90.30 function diguyViewLight::get_scene_objects_cast_shadows

Prototype:

```
int get_scene_objects_cast_shadows();
```

5.90.31 function diguyViewLight::set_brightness_multiplier

Prototype:

```
int set_brightness_multiplier(float multiplier);
```

5.90.32 function diguyViewLight::get_brightness_multiplier**Prototype:**

```
float get_brightness_multiplier();
```

5.90.33 function diguyViewLight::set_shadow_volume_size**Prototype:**

```
void set_shadow_volume_size(float volume);
```

5.90.34 function diguyViewLight::get_shadow_volume_size**Prototype:**

```
float get_shadow_volume_size();
```

5.90.35 function diguyViewLight::set_shadow_offset_multiplier**Prototype:**

```
void set_shadow_offset_multiplier(float multiplier);
```

5.90.36 function diguyViewLight::get_shadow_offset_multiplier**Prototype:**

```
float get_shadow_offset_multiplier();
```

5.90.37 function diguyViewLight::set_spotlight_angle**Prototype:**

```
void set_spotlight_angle(float angle);
```


5.90.38 function diguyViewLight::get_spotlight_angle**Prototype:**

```
float get_spotlight_angle();
```

5.90.39 function diguyViewLight::set_texture_name**Prototype:**

```
void set_texture_name(const char* name);
```

5.90.40 function diguyViewLight::set_active**Prototype:**

```
void set_active(int active);
```

5.90.41 function diguyViewLight::get_active**Prototype:**

```
int get_active();
```

5.90.42 function**diguyViewLight::set_shadow_map_update_paused****Prototype:**

```
int set_shadow_map_update_paused(int pause);
```

5.90.43 function**diguyViewLight::get_shadow_map_update_paused****Prototype:**

```
int get_shadow_map_update_paused();
```

5.90.44 function

diguyViewLight::set_shadow_map_update_frequency

Prototype:

```
int set_shadow_map_update_frequency(int frame_frequency);
```

5.90.45 function

diguyViewLight::get_shadow_map_update_frequency

Prototype:

```
int get_shadow_map_update_frequency();
```

5.91 diguyViewLightSettings

diguyViewLightSettings.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

Summary:

The [diguyViewLightSettings](#) class holds camera settings data that can be loaded into a [diguyViewLight](#) using the [diguyViewLight::load_settings\(\)](#) call.

The settings of a [diguyViewLightSettings](#) object cannot be directly applied to a graphics environment viewpoint; the settings must be loaded into a [diguyViewLight](#) to be applied.

The [diguyViewLightSettings](#) API closely mirrors the [diguyViewLight](#) API, with the exception of load and callback functions.

5.91.1 class diguyViewLightSettings

```
class BDI_DECLSPEC_diguy diguyViewLightSettings
{
public:
    const char* get_name();
    const char* get_type_name();
    void set_direction(float x, float y, float z);
    void get_direction(float* x, float* y, float* z);
    void set_ambient_color(float r, float g, float b, float a);
    void get_ambient_color(float* r, float* g, float* b, float* a);
    void set_diffuse_color(float r, float g, float b, float a);
    void get_diffuse_color(float* r, float* g, float* b, float* a);
    void set_specular_color(float r, float g, float b, float a);
    void get_specular_color(float* r, float* g, float* b, float* a);
```

5.91.2 function diguyViewLightSettings::get_name

Prototype:

```
const char* get_name();
```

Description:

The `diguyViewLightSettings` equivalent of `diguyViewLight::get_name()`.

5.91.3 function diguyViewLightSettings::get_type_name

Prototype:

```
const char* get_type_name();
```

Description:

The `diguyViewLightSettings` equivalent of `diguyViewLight::get_type_name()`.

5.91.4 function diguyViewLightSettings::set_direction

Prototype:

```
void set_direction(float x, float y, float z);
```

Description:

The `diguyViewLightSettings` equivalent of `diguyViewLight::set_direction()`.

5.91.5 function diguyViewLightSettings::get_direction

Prototype:

```
void get_direction(float* x, float* y, float* z);
```

Description:

The `diguyViewLightSettings` equivalent of `diguyViewLight::get_direction()`.

5.91.6 function diguyViewLightSettings::set_ambient_color

Prototype:

```
void set_ambient_color(float r, float g, float b, float a);
```

Description:

The `diguyViewLightSettings` equivalent of `diguyViewLight::set_ambient_color()`.

5.91.7 function diguyViewLightSettings::get_ambient_color

Prototype:

```
void get_ambient_color(float* r, float* g, float* b, float* a);
```

Description:

The `diguyViewLightSettings` equivalent of `diguyViewLight::get_ambient_color()`.

5.91.8 function diguyViewLightSettings::set_diffuse_color

Prototype:

```
void set_diffuse_color(float r, float g, float b, float a);
```

Description:

The `diguyViewLightSettings` equivalent of `diguyViewLight::set_diffuse_color()`.

5.91.9 function diguyViewLightSettings::get_diffuse_color

Prototype:

```
void get_diffuse_color(float* r, float* g, float* b, float* a);
```

Description:

The `diguyViewLightSettings` equivalent of `diguyViewLight::get_diffuse_color()`.

5.91.10 function diguyViewLightSettings::set_specular_color**Prototype:**

```
void set_specular_color(float r, float g, float b, float a);
```

Description:

The `diguyViewLightSettings` equivalent of `diguyViewLight::set_specular_color()`.

5.91.11 function diguyViewLightSettings::get_specular_color**Prototype:**

```
void get_specular_color(float* r, float* g, float* b, float* a);
```

Description:

The `diguyViewLightSettings` equivalent of `diguyViewLight::get_specular_color()`.

5.92 diguyViewPainter

diguyViewPainter.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

- ◆ [State-Setting Functions](#)
- ◆ [View Functions](#)
- ◆ [3D Draw Functions](#)
- ◆ [2D Draw Functions](#)

Link against: libdiguy

Summary:

The [diguyViewPainter](#) class acts as a helper for drawing various graphics primitives in DI-Guy Scenario 3D views.

Currently this class is only supported in DI-Guy Scenario, and should be utilized only inside of a `diguyScenario::CALLBACK_ID_POST_DRAW` callback, or inside of a mind draw function.

There is only one [diguyViewPainter](#). Retrieve a pointer to it by calling [diguyApp::get_view_painter\(\)](#).

5.92.1 class diguyViewPainter

```
class BDI_DECLSPEC_diguy diguyViewPainter
{
public:
    virtual void set_line_width(float width);
    virtual void set_pen_color(float red, float green, float blue, float alpha = 1.0f);
    virtual void set_pen2_color(float red, float green, float blue, float alpha = 1.0f);
    virtual void set_brush_color(float red, float green, float blue, float alpha = 1.0f);
    virtual void get_viewport_size(float* width, float* height);
    virtual void world_to_screen(float world_x, float world_y, float world_z,
        float* view_x, float* view_y, float* view_z);
    virtual void draw_line(float x1, float y1, float z1,
        float x2, float y2, float z2);
    virtual void draw_line_2c(float x1, float y1, float z1,
        float x2, float y2, float z2);
    virtual void draw_sphere(float x, float y, float z,
        float radius,
        int draw_edges = 1,
        int draw_fill = 1);
    virtual void draw_box(float x1, float y1, float z1,
        float x2, float y2, float z2,
```

```

        int draw_edges = 1,
        int draw_fill = 1);
virtual void draw_rotated_box(float x1, float y1, float z1,
    float x2, float y2, float z2,
    float yaw, float roll, float pitch,
    float xc, float yc, float zc,
    int edges = 1,
    int fill = 1);
virtual void draw_arrow(float x1, float y1, float z1,
    float x2, float y2, float z2,
    float shaft_width = 0.02,
    float arrow_width = 0.05,
    int draw_edges = 1,
    int draw_fill = 1);
virtual void draw_disc(float x, float y, float z,
    float radius,
    int draw_edges = 1,
    int draw_fill = 1,
    int num_sides = 12);
virtual void draw_circle(float x, float y, float z,
    float normal_x, float normal_y, float normal_z,
    float radius,
    int draw_edges = 1,
    int draw_fill = 1,
    int num_sides = 12);
virtual void draw_torus(float x, float y, float z,
    float inner_radius,
    float outer_radius,
    int draw_edges = 1,
    int draw_fill = 1,
    int num_sides = 12);
virtual void draw_cylinder(float x, float y, float z,
    float radius,
    float height,
    int draw_edges = 1,
    int draw_fill = 1);
virtual void draw_wedge(float x, float y, float z,
    float target_x, float target_y, float target_z,
    float inner_radius,
    float width,
    int draw_edges = 1,
    int draw_fill = 1);
virtual void draw_wedge_with_normal(float x, float y, float z,
    float target_x, float target_y, float target_z,
    float nx, float ny, float nz,
    float inner_radius,
    float width,
    int draw_edges = 1,
    int draw_fill = 1);
virtual void begin_2d_text_mode();
virtual void end_2d_text_mode();
virtual void draw_text(float screen_x, float screen_y, const char* text);
virtual void draw_2d_rect(float x0,
    float y0,
    float x1,
    float y1,
    int draw_edges = 1,
    int draw_fill = 1);
virtual void draw_private_path_shape(diguyCharacter* diguy_character,
    int include_waypoints = 0);

```

State-Setting Functions

5.92.2 function diguyViewPainter::set_line_width

Prototype:

```
virtual void set_line_width(float width);
```

Description:

Sets the current width of lines drawn by the painter.

Callable From:

- ◇ C++
- ◇ Script

5.92.3 function diguyViewPainter::set_pen_color

Prototype:

```
virtual void set_pen_color(float red, float green, float blue, float alpha = 1.0f);
```

Description:

Sets the color of the default pen. Pens typically draw lines and borders around shapes. Color values should be between 0.0 and 1.0.

Callable From:

- ◇ C++
- ◇ Script

5.92.4 function diguyViewPainter::set_pen2_color

Prototype:

```
virtual void set_pen2_color(float red, float green, float blue, float alpha = 1.0f);
```

Description:

Sets the color of the secondary pen. Color values should be between 0.0 and 1.0.

Callable From:

- ◇ C++
- ◇ Script

5.92.5 function diguyViewPainter::set_brush_color

Prototype:

```
virtual void set_brush_color(float red, float green, float blue, float alpha = 1.0f)
```

Description:

Sets the color of the brush. Brushes are typically used for filling shapes. Color values should be between 0.0 and 1.0.

Callable From:

- ◇ C++
- ◇ Script

View Functions

5.92.6 function diguyViewPainter::get_viewport_size

Prototype:

```
virtual void get_viewport_size(float* width, float* height);
```

Description:

Fills out the width and height arguments with the dimensions of the primary 3D view.

Callable From:

- ◇ C++
- ◇ Script

5.92.7 function diguyViewPainter::world_to_screen

Prototype:

```
virtual void world_to_screen(float world_x, float world_y, float world_z,  
float* view_x, float* view_y, float* view_z);
```

Description:

This function converts a world (x, y, z) coordinate into a view (x, y, z) coordinate.

Note: this function must be called **before** 2d text mode is entered with the [begin_2d_text_mode\(\)](#) function.

If the returned view z value is greater than 1, the returned coordinate is behind the camera.

3D Draw Functions

In all 3D draw functions, if there are edges to be drawn they are drawn with the current pen color as set by `set_pen_color( )`. If there is an area to be filled it will be filled with the current brush color, as set by `set_brush_color( )`.

5.92.8 function diguyViewPainter::draw_line

Prototype:

```
virtual void draw_line(float x1, float y1, float z1,  
                      float x2, float y2, float z2);
```

Description:

Draws a line from (x1, y1, z1) to (x2, y2, z2) using the pen color.

Callable From:

- ◇ C++
- ◇ Script

5.92.9 function diguyViewPainter::draw_line_2c

Prototype:

```
virtual void draw_line_2c(float x1, float y1, float z1,  
    float x2, float y2, float z2);
```

Description:

Draws a line from (x1, y1, z1) to (x2, y2, z2) using both of the pen colors.
The color of the line is interpolated from point 1 to point 2.

Callable From:

- ◇ C++
- ◇ Script

5.92.10 function diguyViewPainter::draw_sphere

Prototype:

```
virtual void draw_sphere(float x, float y, float z,  
    float radius,  
    int draw_edges = 1,  
    int draw_fill = 1);
```

Description:

Draws a sphere with its center at (x, y, z), with radius radius.

Callable From:

- ◇ C++
- ◇ Script

5.92.11 function diguyViewPainter::draw_box

Prototype:

```
virtual void draw_box(float x1, float y1, float z1,
    float x2, float y2, float z2,
    int draw_edges = 1,
    int draw_fill = 1);
```

Description:

Draws an axis aligned bounding box with one corner at (x1, y1, z1), the other at (x2, y2, z2).

Callable From:

- ◇ C++
- ◇ Script

5.92.12 function diguyViewPainter::draw_rotated_box

Prototype:

```
virtual void draw_rotated_box(float x1, float y1, float z1,
    float x2, float y2, float z2,
    float yaw, float roll, float pitch,
    float xc, float yc, float zc,
    int edges = 1,
    int fill = 1);
```

Description:

Draws a rotated bounding box with one corner at unrotated (x1, y1, z1), and another at unrotated (x2, y2, z2), applying the rotation Euler angles (given in BDI_EULER_ANGLE_ORDER_ZXY order in degrees). The box is drawn at center point (xc, yc, zc) in global space, and then rotated.

Callable From:

- ◇ C++
- ◇ Script

5.92.13 function diguyViewPainter::draw_arrow**Prototype:**

```
virtual void draw_arrow(float x1, float y1, float z1,
    float x2, float y2, float z2,
    float shaft_width = 0.02,
    float arrow_width = 0.05,
    int draw_edges = 1,
    int draw_fill = 1);
```

Description:

Draws an arrow from (x1, y1, z1) to (x2, y2, z2).

Callable From:

- ◇ C++
- ◇ Script

5.92.14 function diguyViewPainter::draw_disc**Prototype:**

```
virtual void draw_disc(float x, float y, float z,
    float radius,
    int draw_edges = 1,
    int draw_fill = 1,
    int num_sides = 12);
```

Description:

Draws a disc with its center at (x, y, z), and the given radius.

Callable From:

- ◇ C++
- ◇ Script

5.92.15 function diguyViewPainter::draw_circle**Prototype:**

```
virtual void draw_circle(float x, float y, float z,
    float normal_x, float normal_y, float normal_z,
    float radius,
    int draw_edges = 1,
    int draw_fill = 1,
    int num_sides = 12);
```

Description:

Draws a circle at (x, y, z), with normal (normal_x, normal_y, normal_z), and the given radius.

Callable From:

- ◇ C++
- ◇ Script

5.92.16 function diguyViewPainter::draw_torus**Prototype:**

```
virtual void draw_torus(float x, float y, float z,
    float inner_radius,
    float outer_radius,
    int draw_edges = 1,
    int draw_fill = 1,
    int num_sides = 12);
```

Description:

Draws a torus at (x, y, z), with normal (normal_x, normal_y, normal_z), and the given radius.

Callable From:

- ◇ C++
- ◇ Script

5.92.17 function diguyViewPainter::draw_cylinder**Prototype:**

```
virtual void draw_cylinder(float x, float y, float z,
    float radius,
    float height,
    int draw_edges = 1,
    int draw_fill = 1);
```

Description:

Draws a Z-axis-up cylinder, with its bottom cap centered at (x1, y1, z). The cylinder will have the given radius and height.

Callable From:

- ◇ C++
- ◇ Script

5.92.18 function diguyViewPainter::draw_wedge**Prototype:**

```
virtual void draw_wedge(float x, float y, float z,
    float target_x, float target_y, float target_z,
    float inner_radius,
    float width,
    int draw_edges = 1,
    int draw_fill = 1);
```

Description:

Draws wedge.

Callable From:

- ◇ C++
- ◇ Script

5.92.19 function diguyViewPainter::draw_wedge_with_normal**Prototype:**

```
virtual void draw_wedge_with_normal(float x, float y, float z,  
    float target_x, float target_y, float target_z,  
    float nx, float ny, float nz,  
    float inner_radius,  
    float width,  
    int draw_edges = 1,  
    int draw_fill = 1);
```

2D Draw Functions

Before 2D drawing functions are called the view must be put into a 2D drawing state by calling `begin_2d_text_mode()`.

5.92.20 function diguyViewPainter::begin_2d_text_mode**Prototype:**

```
virtual void begin_2d_text_mode();
```

Description:

Puts the OpenGL view matrix into a 2D orthogonal state so that 2D primitives will be drawn in 2D view coordinates rather than in 3D world coordinates.

This call **must** be balanced by a call to [end_2d_text_mode\(\)](#), or overall drawing will be broken.

Callable From:

- ◇ C++
- ◇ Script

5.92.21 function diguyViewPainter::end_2d_text_mode**Prototype:**

```
virtual void end_2d_text_mode();
```

Description:

Returns the OpenGL view matrix back into a 3D perspective state.

This call **must** be balanced by a call to [begin_2d_text_mode\(\)](#), or overall drawing will be broken.

Callable From:

- ◇ C++
- ◇ Script

5.92.22 function diguyViewPainter::draw_text**Prototype:**

```
virtual void draw_text(float screen_x, float screen_y, const char* text);
```

Description:

Draws the passed text string at view coordinates (x, y).

Should only be called inside of a `begin_2d_text_mode()` and `end_2d_text_mode()` block.

The text is drawn with the current pen color.

Callable From:

- ◇ C++
- ◇ Script

5.92.23 function diguyViewPainter::draw_2d_rect**Prototype:**

```
virtual void draw_2d_rect(float x0,
    float y0,
    float x1,
    float y1,
    int draw_edges = 1,
    int draw_fill = 1);
```

Description:

Draws a 2D rectangle with the given points as the corners.

Should only be called inside of a `begin_2d_text_mode()` and `end_2d_text_mode()` block.

The edges are drawn with the current pen color, the fill is drawn with the current brush color.

Callable From:

- ◇ C++

◇ Script

5.92.24 function diguyViewPainter::draw_private_path_shape

Prototype:

```
virtual void draw_private_path_shape(diguyCharacter* diguy_character,  
    int include_waypoints = 0);
```

5.93 diguyWaypoint

diguyWaypoint.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

- ◆ [General Functions](#)
- ◆ [Position and Orientation Functions](#)
- ◆ [Control Point Functions](#)
- ◆ [Other Functions](#)
- ◆ [Deprecated Functions](#)

Link against: **libdiguy**

5.93.1 class diguyWaypoint

```
class BDI_DECLSPEC_diguy diguyWaypoint
{
public:
    const char* get_name();
    long get_uid();
    int get_selected();
    void set_position(float x, float y, float z);
    void set_position_double(double x, double y, double z);
    void get_position(float* x, float* y, float* z);
    void get_position_double(double* tx, double* ty, double* tz);
    void set_orientation_in(float rz, float rx, float ry);
    void get_orientation_in(float* rz, float* rx, float* ry);
    void set_orientation_out(float rz, float rx, float ry);
    void get_orientation_out(float* rz, float* rx, float* ry);
    void set_yaw_coupled(int yaw_coupled);
    int get_yaw_coupled();
    void set_yaw_coupling_offset(float yaw_coupling_offset);
    float get_yaw_coupling_offset();
    void set_roll_coupled(int roll_coupled);
    int get_roll_coupled();
    void set_roll_coupling_offset(float roll_coupling_offset);
    float get_roll_coupling_offset();
    void set_pitch_coupled(int pitch_coupled);
    int get_pitch_coupled();
    void set_pitch_coupling_offset(float pitch_coupling_offset);
    float get_pitch_coupling_offset();
    void set_weight_in(float weight_in);
    float get_weight_in();
    void set_weight_out(float weight_out);
    float get_weight_out();
    void set_weight_coupled(int weight_coupled);
    int get_weight_coupled();
    void get_control_point_in_position(float* x, float* y, float* z);
    void get_control_point_out_position(float* x, float* y, float* z);
```

```

void get_forward_normal_in(float* x, float* y, float* z);
void get_left_normal_in(float* x, float* y, float* z);
void get_up_normal_in(float* x, float* y, float* z);
void get_forward_normal_out(float* x, float* y, float* z);
void get_left_normal_out(float* x, float* y, float* z);
void get_up_normal_out(float* x, float* y, float* z);
void set_num_segments(int num_segments);
int get_num_segments();
void set_x(float x);
float get_x();
void set_y(float y);
float get_y();
void set_z(float z);
float get_z();
void set_yaw_in(float yaw);
float get_yaw_in();
void set_yaw_out(float yaw);
float get_yaw_out();
int get_yaw_smooth();
void set_yaw_smooth(int yaw_smooth);
float get_yaw_offset();
void set_yaw_offset(float yaw_offset);
void set_roll_in(float roll);
float get_roll_in();
void set_roll_out(float roll);
float get_roll_out();
int get_roll_smooth();
void set_roll_smooth(int roll_smooth);
float get_roll_offset();
void set_roll_offset(float roll_offset);
void set_pitch_in(float pitch);
float get_pitch_in();
void set_pitch_out(float pitch);
float get_pitch_out();
int get_pitch_smooth();
void set_pitch_smooth(int pitch_smooth);
float get_pitch_offset();
void set_pitch_offset(float pitch_offset);
int get_distance_smooth();
void set_distance_smooth(int distance_smooth);

```

General Functions

Unless otherwise noted all functions in this class section are callable from:

- ◆ C++
- ◆ Script

5.93.2 function diguyWaypoint::get_name

Prototype:

```
const char* get_name();
```

Description:

Returns the name of the object. This pointer will never be NULL.

Returns:

name of the object

5.93.3 function diguyWaypoint::get_uid

Prototype:

```
long get_uid();
```

Description:

All waypoints are assigned a unique identifier, or uid. This function returns this waypoints's uid.

Note: unique identifiers will change between DI-Guy runs!

Returns:

unique identifier of object

5.93.4 function diguyWaypoint::get_selected

Prototype:

```
int get_selected();
```

Returns:

1 if waypoint is selected in DI-Guy Scenario or DI-Guy Author, 0 if not.

Position and Orientation Functions

The *Boston Dynamics* global coordinate system is right-handed, with X forward, Z up, and Y to the left. Rotation directions follow standard right-handed coordinate system conventions:

- ◆ positive rotations about Z cause (rz) a yaw to the left
- ◆ positive rotations about X cause (rx) a counter-clockwise roll
- ◆ position rotations about Y cause (ry) a forward pitch

NOTE:

If modifications are made to the waypoint using any of the `set_*( )` commands, if the waypoint is part of a `diguyCharacterPath` or `diguyPathShape`, the `diguyCharacterPath::update( )` or `diguyPathShape::update( )` function must be called for changes to take effect. Multiple changes can be made before the update calls are made.

5.93.5 function `diguyWaypoint::set_position`

Prototype:

```
void set_position(float x, float y, float z);
```

Description:

This function sets the x, y, z coordinates of the waypoint in meters.

Requires subsequent call to `diguyCharacterPath::update( )` or `diguyPathShape::update( )`. (See above.)

Returns:

x, y, z coordinate of waypoint in meters

5.93.6 function diguyWaypoint::set_position_double

Prototype:

```
void set_position_double(double x, double y, double z);
```

Description:

Double precision version of [set_position\(\)](#).

5.93.7 function diguyWaypoint::get_position

Prototype:

```
void get_position(float* x, float* y, float* z);
```

Returns:

x, y, z coordinate of waypoint in meters

Note: in Lua this will return 3 values.

5.93.8 function diguyWaypoint::get_position_double

Prototype:

```
void get_position_double(double* tx, double* ty, double* tz);
```

Description:

Double precision version of [get_position\(\)](#).

Note: in Lua this will return 3 values.

5.93.9 function diguyWaypoint::set_orientation_in

Prototype:

```
void set_orientation_in(float rz, float rx, float ry);
```

Description:

Sets the orientation of the path entering the waypoint.

If the angles in and angles out are coupled (e.g., [get_yaw_coupled\(\)](#) returns 1) this will set rz out, as well. The new angle out will be angle in plus the angle coupling offset.

Requires subsequent call to [diguyCharacterPath::update\(\)](#) or [diguyPathShape::update\(\)](#). (See above.)

Arguments:

rz, rx,	orientations in degrees
ry	

5.93.10 function diguyWaypoint::get_orientation_in**Prototype:**

```
void get_orientation_in(float* rz, float* rx, float* ry);
```

Description:

Retrieves the orientation of the path entering the waypoint.

Arguments:

<code>rz, rx,</code>	orientations in degrees counter-clockwise from the positive
<code>ry</code>	X axis

Note: in Lua this will return 3 values.

5.93.11 function diguyWaypoint::set_orientation_out**Prototype:**

```
void set_orientation_out(float rz, float rx, float ry);
```

Description:

Equivalent to [set_orientation_in\(\)](#), but for the path exiting the waypoint.

5.93.12 function diguyWaypoint::get_orientation_out

Prototype:

```
void get_orientation_out(float* rz, float* rx, float* ry);
```

Description:

Equivalent to [get_orientation_in\(\)](#), but for the path exiting the waypoint.

Note: in Lua this will return 3 values.

5.93.13 function diguyWaypoint::set_yaw_coupled

Prototype:

```
void set_yaw_coupled(int yaw_coupled);
```

Description:

This function sets whether the yaw in and yaw out angles are coupled together. In general, for paths to look smooth, the in and out angles should be coupled together with 0 coupling offset.

Requires subsequent call to [diguyCharacterPath::update\(\)](#) or [diguyPathShape::update\(\)](#). (See above.)

Arguments:

`yaw_coupled` pass 1 to couple yaw in to yaw out

5.93.14 function diguyWaypoint::get_yaw_coupled**Prototype:**

```
int get_yaw_coupled();
```

Returns:

whether the yaw in and yaw out angles are coupled together; see [set_yaw_coupled\(\)](#)

5.93.15 function diguyWaypoint::set_yaw_coupling_offset**Prototype:**

```
void set_yaw_coupling_offset(float yaw_coupling_offset);
```

Description:

This function sets by how much the yaw in and yaw out angles should be offset when they are coupled together.

Requires subsequent call to [diguyCharacterPath::update\(\)](#) or [diguyPathShape::update\(\)](#). (See above.)

Arguments:

`yaw_coupling_offset` new yaw coupling offset, in degrees

5.93.16 function diguyWaypoint::get_yaw_coupling_offset**Prototype:**

```
float get_yaw_coupling_offset();
```

Returns:

the offset between yaw in and yaw out when the yaw angles are coupled together; see [set_yaw_coupling_offset\(\)](#)

5.93.17 function diguyWaypoint::set_roll_coupled**Prototype:**

```
void set_roll_coupled(int roll_coupled);
```

Description:

This function is the same as its yaw counterpart, but rotation is about the x axis.

5.93.18 function diguyWaypoint::get_roll_coupled**Prototype:**

```
int get_roll_coupled();
```

Description:

This function is the same as its yaw counterpart, but rotation is about the x axis.

5.93.19 function diguyWaypoint::set_roll_coupling_offset**Prototype:**

```
void set_roll_coupling_offset(float roll_coupling_offset);
```

Description:

This function is the same as its yaw counterpart, but rotation is about the x axis.

5.93.20 function diguyWaypoint::get_roll_coupling_offset**Prototype:**

```
float get_roll_coupling_offset();
```

Description:

This function is the same as its yaw counterpart, but rotation is about the x axis.

5.93.21 function diguyWaypoint::set_pitch_coupled**Prototype:**

```
void set_pitch_coupled(int pitch_coupled);
```

Description:

This function is the same as its yaw counterpart, but rotation is about the y axis.

5.93.22 function diguyWaypoint::get_pitch_coupled**Prototype:**

```
int get_pitch_coupled();
```

Description:

This function is the same as its yaw counterpart, but rotation is about the y axis.

5.93.23 function diguyWaypoint::set_pitch_coupling_offset**Prototype:**

```
void set_pitch_coupling_offset(float pitch_coupling_offset);
```

Description:

This function is the same as its yaw counterpart, but rotation is about the y axis.

5.93.24 function diguyWaypoint::get_pitch_coupling_offset

Prototype:

```
float get_pitch_coupling_offset();
```

Description:

This function is the same as its yaw counterpart, but rotation is about the y axis.

Control Point Functions

Unless otherwise noted all functions in this class section are callable from:

- ◆ C++
- ◆ Script

5.93.25 function diguyWaypoint::set_weight_in**Prototype:**

```
void set_weight_in(float weight_in);
```

Description:

This function sets the weight of the control handle heading into the waypoint. The weights of the waypoint set how long the "control handles" of the waypoint are, in meters. The longer the control handle, the more the path will be stretched in the direction of the waypoint's orientation.

If the weight in and weight out are coupled ([get_weight_coupled\(\)](#) returns 1) this will set weight out, as well.

Requires subsequent call to [diguyCharacterPath::update\(\)](#) or [diguyPathShape::update\(\)](#). (See above.)

5.93.26 function diguyWaypoint::get_weight_in**Prototype:**

```
float get_weight_in();
```

Returns:

weight heading into the waypoint, in meters; see [set_weight_in\(\)](#)

5.93.27 function diguyWaypoint::set_weight_out**Prototype:**

```
void set_weight_out(float weight_out);
```

Description:

Similar to [set_weight_in\(\)](#), but for the outgoing weight.

5.93.28 function diguyWaypoint::get_weight_out**Prototype:**

```
float get_weight_out();
```

Returns:

weight heading out of the waypoint, in meters; see [set_weight_out\(\)](#).

5.93.29 function diguyWaypoint::set_weight_coupled**Prototype:**

```
void set_weight_coupled(int weight_coupled);
```

Description:

This function sets whether the weight in and weight out are coupled together.

If this waypoint is part of a [diguyCharacterPath](#) or [diguyPathShape](#), the [diguyCharacterPath::update\(\)](#) or [diguyPathShape::update\(\)](#) function must be called for changes to take effect.

5.93.30 function diguyWaypoint::get_weight_coupled**Prototype:**

```
int get_weight_coupled();
```

Returns:

whether the weight in and weight out are coupled together; see [set_weight_coupled\(\)](#)

5.93.31 function diguyWaypoint::get_control_point_in_position**Prototype:**

```
void get_control_point_in_position(float* x, float* y, float* z);
```

Description:

This function returns the world position of the control point heading into the waypoint. This is derived from the waypoint's position, orientation in, and the weights of the in control bar.

5.93.32 function diguyWaypoint::get_control_point_out_position**Prototype:**

```
void get_control_point_out_position(float* x, float* y, float* z);
```

Description:

Similar to `get_control_point_in_position()`, but for out control point.

5.93.33 function diguyWaypoint::get_forward_normal_in**Prototype:**

```
void get_forward_normal_in(float* x, float* y, float* z);
```

Description:

This function returns the normal vector of the control bar heading into the waypoint.

5.93.34 function diguyWaypoint::get_left_normal_in**Prototype:**

```
void get_left_normal_in(float* x, float* y, float* z);
```

Description:

Similar to [get_forward_normal_in\(\)](#), but for left normal.

5.93.35 function diguyWaypoint::get_up_normal_in**Prototype:**

```
void get_up_normal_in(float* x, float* y, float* z);
```

Description:

Similar to `get_out_normal_in( )`, but for up normal.

5.93.36 function diguyWaypoint::get_forward_normal_out**Prototype:**

```
void get_forward_normal_out(float* x, float* y, float* z);
```

Description:

Similar to `get_forward_normal_in( )`, but for out control bar.

5.93.37 function diguyWaypoint::get_left_normal_out**Prototype:**

```
void get_left_normal_out(float* x, float* y, float* z);
```

Description:

Similar to [get_left_normal_in\(\)](#), but for out control bar.

5.93.38 function diguyWaypoint::get_up_normal_out**Prototype:**

```
void get_up_normal_out(float* x, float* y, float* z);
```

Description:

Similar to [get_up_normal_in\(\)](#), but for out control bar.

Other Functions

5.93.39 function diguyWaypoint::set_num_segments

Prototype:

```
void set_num_segments(int num_segments);
```

Description:

This function sets how many segments the path portion this waypoint begins will be divided into. This is strictly for visualization purposes only and will have no functional effect.

If this waypoint is part of a [diguyCharacterPath](#) or [diguyPathShape](#), the [diguyCharacterPath::update\(\)](#) or [diguyPathShape::update\(\)](#) function must be called for changes to take effect.

Arguments:

`num_segments` number of segments to divide path portion into;

default is 64

5.93.40 function diguyWaypoint::get_num_segments**Prototype:**

```
int get_num_segments();
```

Returns:

how many segments the path portion this waypoint begins will be divided into; see [set_num_segments\(\)](#)

Deprecated Functions

The following functions are deprecated as of DI-Guy 10

As these functions may disappear in a future version of DI-Guy, it is recommended that you replace calls to these functions from your application.

5.93.41 function diguyWaypoint::set_x**Prototype:**

```
void set_x(float x);
```

This function is deprecated. Use [set_position\(\)](#) instead.

5.93.42 function diguyWaypoint::get_x**Prototype:**

```
float get_x();
```

This function is deprecated. Use [get_position\(\)](#) instead.

5.93.43 function diguyWaypoint::set_y

Prototype:

```
void set_y(float y);
```

This function is deprecated. Use [set_position\(\)](#) instead.

5.93.44 function diguyWaypoint::get_y

Prototype:

```
float get_y();
```

This function is deprecated. Use [get_position\(\)](#) instead.

5.93.45 function diguyWaypoint::set_z**Prototype:**

```
void set_z(float z);
```

This function is deprecated. Use [set_position\(\)](#) instead.

5.93.46 function diguyWaypoint::get_z**Prototype:**

```
float get_z();
```

This function is deprecated. Use [get_position\(\)](#) instead.

5.93.47 function diguyWaypoint::set_yaw_in**Prototype:**

```
void set_yaw_in(float yaw);
```

This function is deprecated. Use [set_orientation_in\(\)](#) instead.

5.93.48 function diguyWaypoint::get_yaw_in**Prototype:**

```
float get_yaw_in();
```

This function is deprecated. Use [get_orientation_in\(\)](#) instead.

5.93.49 function diguyWaypoint::set_yaw_out**Prototype:**

```
void set_yaw_out(float yaw);
```

This function is deprecated. Use [set_orientation_out\(\)](#) instead.

5.93.50 function diguyWaypoint::get_yaw_out**Prototype:**

```
float get_yaw_out();
```

This function is deprecated. Use [get_orientation_out\(\)](#) instead.

5.93.51 function diguyWaypoint::get_yaw_smooth**Prototype:**

```
int get_yaw_smooth();
```

This function is deprecated. Use [get_yaw_coupled\(\)](#) instead.

5.93.52 function diguyWaypoint::set_yaw_smooth**Prototype:**

```
void set_yaw_smooth(int yaw_smooth);
```

This function is deprecated. Use [set_yaw_coupled\(\)](#) instead.

5.93.53 function diguyWaypoint::get_yaw_offset**Prototype:**

```
float get_yaw_offset();
```

This function is deprecated. Use [get_yaw_coupling_offset\(\)](#) instead.

5.93.54 function diguyWaypoint::set_yaw_offset**Prototype:**

```
void set_yaw_offset(float yaw_offset);
```

This function is deprecated. Use [set_yaw_coupling_offset\(\)](#) instead.

5.93.55 function diguyWaypoint::set_roll_in**Prototype:**

```
void set_roll_in(float roll);
```

This function is deprecated. Use [set_orientation_in\(\)](#) instead.

5.93.56 function diguyWaypoint::get_roll_in**Prototype:**

```
float get_roll_in();
```

This function is deprecated. Use [get_orientation_in\(\)](#) instead.

5.93.57 function diguyWaypoint::set_roll_out**Prototype:**

```
void set_roll_out(float roll);
```

This function is deprecated. Use [set_orientation_out\(\)](#) instead.

5.93.58 function diguyWaypoint::get_roll_out**Prototype:**

```
float get_roll_out();
```

This function is deprecated. Use [get_orientation_out\(\)](#) instead.

5.93.59 function diguyWaypoint::get_roll_smooth**Prototype:**

```
int get_roll_smooth();
```

This function is deprecated. Use [get_roll_coupled\(\)](#) instead.

5.93.60 function diguyWaypoint::set_roll_smooth**Prototype:**

```
void set_roll_smooth(int roll_smooth);
```

This function is deprecated. Use [set_roll_coupled\(\)](#) instead.

5.93.61 function diguyWaypoint::get_roll_offset**Prototype:**

```
float get_roll_offset();
```

This function is deprecated. Use [get_roll_coupling_offset\(\)](#) instead.

5.93.62 function diguyWaypoint::set_roll_offset**Prototype:**

```
void set_roll_offset(float roll_offset);
```

This function is deprecated. Use [set_roll_coupling_offset\(\)](#) instead.

5.93.63 function diguyWaypoint::set_pitch_in**Prototype:**

```
void set_pitch_in(float pitch);
```

This function is deprecated. Use [set_orientation_in\(\)](#) instead.

5.93.64 function diguyWaypoint::get_pitch_in**Prototype:**

```
float get_pitch_in();
```

This function is deprecated. Use [get_orientation_in\(\)](#) instead.

5.93.65 function diguyWaypoint::set_pitch_out

Prototype:

```
void set_pitch_out(float pitch);
```

This function is deprecated. Use [set_orientation_out\(\)](#) instead.

5.93.66 function diguyWaypoint::get_pitch_out

Prototype:

```
float get_pitch_out();
```

This function is deprecated. Use [get_orientation_out\(\)](#) instead.

5.93.67 function diguyWaypoint::get_pitch_smooth**Prototype:**

```
int get_pitch_smooth();
```

This function is deprecated. Use [get_pitch_coupled\(\)](#) instead.

5.93.68 function diguyWaypoint::set_pitch_smooth**Prototype:**

```
void set_pitch_smooth(int pitch_smooth);
```

This function is deprecated. Use [set_pitch_coupled\(\)](#) instead.

5.93.69 function diguyWaypoint::get_pitch_offset**Prototype:**

```
float get_pitch_offset();
```

This function is deprecated. Use [get_pitch_coupling_offset\(\)](#) instead.

5.93.70 function diguyWaypoint::set_pitch_offset**Prototype:**

```
void set_pitch_offset(float pitch_offset);
```

This function is deprecated. Use [get_pitch_coupling_offset\(\)](#) instead.

5.93.71 function diguyWaypoint::get_distance_smooth**Prototype:**

```
int get_distance_smooth();
```

This function is deprecated. Use [get_weight_coupled\(\)](#) instead.

5.93.72 function diguyWaypoint::set_distance_smooth**Prototype:**

```
void set_distance_smooth(int distance_smooth);
```

This function is deprecated. Use [set_weight_coupled\(\)](#) instead.

5.94 DI-Guy AI, Main Header File

diguy_ai.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

◆ [Initialization Functions](#)

Link against: libdiguy

Initialization Functions

5.94.1 function diguy_ai_initialize

Prototype:

```
int diguy_ai_initialize();
```

Description:

This function turns on DI-Guy AI. It should be called after the main DI-Guy initialize call.

Callable From:

◇ C++

Returns:

0 on success, -1 on failure

C++ Example:

```
#include <diguy_graphics_ogl.h>

diguy_ogl_initialize(NULL);
diguy_ai_initialize();
```

5.94.2 function diguy_ai_deinitialize

Prototype:

```
int diguy_ai_deinitialize();
```

Description:

This function turns off DI-Guy AI. It should be called before the main DI-Guy deinitialize call.

Callable From:

◇ C++

Returns:

0 on success, -1 on failure

C++ Example:

```
diguy_ai_deinitialize();
diguy_deinitialize();
```

5.95 DI-Guy API, Main Header File

diguy_api.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [declspec_diguy.h](#)

Contents:

- ◆ [Initialization Functions](#)
- ◆ [Scenario Creation Functions](#)
- ◆ [Utility Functions](#)

The API is subject to change in future releases.

Link against: libdiguy

Initialization Functions

5.95.1 function `diguy_set_cfg_file`

Prototype:

```
int diguy_set_cfg_file(const char* cfg_file);
```

Description:

This function sets the filename of the top-level configuration file that will be read by the DI-Guy software. The default is "diguy.cfg".

This function must be called before `diguy_initialize_[platform]()`.

It should rarely be necessary to use this function.

Callable From:

◇ C++

Arguments:

`cfg_file` filename of desired configuration file

Returns:

0 on success, -1 on failure

C++ Example:

```
#include <diguy_graphics_ogl.h>

diguy_set_cfg_file("custom.cfg");
diguy_ogl_initialize(NULL);
```

5.95.2 function diguy_set_diguy_base_dir_path

Prototype:

```
int diguy_set_diguy_base_dir_path(const char* directory_name);
```

Description:

This function sets the base directory in which to search for DI-Guy files.

This function must be called before `diguy_initialize_[platform]()`.

It should rarely be necessary to use this function. In typical use, this directory will be read from the DIGUY environment variable.

Calling this function will override the directory specified in the DIGUY environment variable.

The base directory can be retrieved by using the function `diguyApp::get_diguy_base_dir_path()`.

Callable From:

◇ C++

Arguments:

`directory_name` override directory in which to search for DI-Guy files

Returns:

0 on success, -1 on failure

C++ Example:

```
#include <diguy_graphics_ogl.h>

diguy_set_diguy_base_dir_path("c:/DI-Guy_alternate_directory");
diguy_ogl_initialize(NULL);
```

5.95.3 function diguy_deinitialize

Prototype:

```
int diguy_deinitialize();
```

Description:

This function shuts down the DI-Guy software, removing all characters and cleaning up memory allocated by the library. No objects created by the DI-Guy software should be referenced after this call is made.

This function should only be called once per program execution.

Do not try to re-initialize the DI-Guy software after calling this function.

Callable From:

◇ C++

Returns:

0 on success, -1 on failure

C++ Example:

```
diguy_deinitialize();
```


Scenario Creation Functions

5.95.4 function `diguy_get_app`

Prototype:

```
diguyApp* diguy_get_app();
```

Description:

This function returns a pointer to the DI-Guy application object. Only one application object exists per DI-Guy run. The application object is created during DI-Guy initialize calls; e.g. `diguy_ogl_initialize()`.

Returns:

a pointer of type `diguyApp`

Callable From:

◇ C++

C++ Example:

```
diguyApp* app = diguy_get_app();
```

5.95.5 function diguy_create_scenario

Prototype:

```
diguyScenario* diguy_create_scenario();
```

Description:

This function allocates and returns an empty object of type `diguyScenario`. This is the top-level "container" of most DI-Guy types, and includes objects of type `diguyCharacter`, `diguyViewCamera`, `diguySceneObject`, and `diguySignal`.

Callable From:

◇ C++

Returns:

Pointer to object of type `diguyScenario` if successful; NULL if not.

C++ Example:

```
diguyScenario* sc = diguy_create_scenario();  
if (sc)  
{  
    sc->load("my_scenario.dss");  
}
```

5.95.6 function `diguy_destroy_scenario`

Prototype:

```
int diguy_destroy_scenario(diguyScenario* scenario);
```

Description:

This function destroys an object of type `diguyScenario` allocated by `diguy_create_scenario()`.

Callable From:

◇ C++

Returns:

0 on success, -1 on failure

C++ Example:

```
diguyScenario* sc = diguy_create_scenario();  
// ...
```

```
diguy_destroy_scenario(sc);
```

Utility Functions

5.95.7 function diguy_convert_euler_angle_order**Prototype:**

```
int diguy_convert_euler_angle_order(float src_rx,
    float src_ry,
    float src_rz,
    diguyEulerAngleOrder src_order,
    float* dest_rx,
    float* dest_ry,
    float* dest_rz,
    diguyEulerAngleOrder dest_order);
```

Description:

This utility function changes the order of Euler angle triples. DI-Guy uses the order ZXY for all 3D rotations.

Callable From:

◇ C++

Returns:

0 on success, -1 on failure

5.95.8 function diguy_convert_euler_angles_to_quaternion**Prototype:**

```
int diguy_convert_euler_angles_to_quaternion(const float* euler_angle_ptr,
    float* quat_ptr);
```

Description:

This utility function converts a DI-Guy ZXY Euler angle triple into a quaternion.

Callable From:

◇ C++

Returns:

0 on success, -1 on failure

5.95.9 function**diguy_extract_translation_and_euler_angles_from_matrix****Prototype:**

```
int diguy_extract_translation_and_euler_angles_from_matrix(float mat[4][4],
float* translation,
float* euler_angles,
diguyEulerAngleOrder euler_angle_order);
```

Description:

To be written.

Callable From:

◇ C++

Returns:

0 on success, -1 on failure

5.95.10 function**diguy_create_matrix_from_translation_and_euler_angles****Prototype:**

```
int diguy_create_matrix_from_translation_and_euler_angles(float mat[4][4],
float translation[3],
float euler_angles[3],
diguyEulerAngleOrder euler_angle_order);
```

Description:

To be written.

Callable From:

◇ C++

Returns:

0 on success, -1 on failure

5.96 DI-Guy AI, Main Header File

diguy_author.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_constants.h](#) | [diguyAuthorInterface.h](#) | [declspec_diguy.h](#)

Contents:

◆ [Initialization Functions](#)

Link against: libdiguy

Initialization Functions

5.96.1 function `diguy_author_initialize`

Prototype:

```
int diguy_author_initialize(diguyAuthorInterfaceCreateFunc* create_func);
```

Description:

This function enables DI-Guy Author. It should be called after the main DI-Guy initialize call.

Note that DI-Guy Author will deinitialize automatically when DI-Guy shuts down.

Arguments:

`create_func` pointer to function that creates a `diguyAuthorInterface` subclass object

DI-Guy Author relies on receiving input from the Host IG that is integrating DI-Guy Author functionality. To do this, the `diguyAuthorInterface` class needs to be subclassed, with the proper virtual functions overridden.

DI-Guy uses this indirect approach to creating objects instead of just setting the object directly to avoid potential memory model problems on some platforms, that can result in difficult to debug problems.

Callable From:

◇ C++

Returns:

0 on success, -1 on failure

5.96.2 function `diguy_author_get_author_enabled`

Prototype:

```
int diguy_author_get_author_enabled();
```

Description:

This function returns whether or not DI-Guy Author functionality is enabled.

Returns:

1 if DI-Guy Author enabled and initialized, 0 if not

5.96.3 function `diguy_author_get_author_interface`

Prototype:

```
diguyAuthorInterface* diguy_author_get_author_interface();
```

Description:

This function returns a pointer to a `diguyAuthorInterface` object. Member functions in this object can be used to communicate input, e.g. mouse clicks or key presses, to DI-Guy so it can create and edit DI-Guy characters and objects in the Host IG.

Returns:

pointer of type `diguyAuthorInterface`

5.97 diguy_author_ui.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

5.97.1 function diguy_author_ui_initialize

Prototype:

```
int diguy_author_ui_initialize(void* qapp = NULL);
```

5.97.2 function diguy_author_ui_deinitialize

Prototype:

```
int diguy_author_ui_deinitialize();
```

5.97.3 function diguy_author_ui_process_ui_events

Prototype:

```
int diguy_author_ui_process_ui_events();
```

5.98 DI-Guy Changes and Version History

diguy_changes.h

DI-Guy SDK Version 12.5.2

Contents:

- ◆ DI-Guy Version 12.5
 - ◇ DI-Guy API
 - Version 12.5.1
 - Version 12.5.0
 - ◇ DI-Guy Scenario
 - Version 12.5.0
 - ◇ DI-Guy AI
 - Version 12.5.0
 - ◇ Graphics API Changes
 - Version 12.5.1
 - Version 12.5.0
 - ◇ DI-Guy Networking Changes
 - Version 12.5
- ◆ DI-Guy Version 12.0
 - ◇ DI-Guy Scenario
 - Version 12.0.1
 - ◇ DI-Guy AI
 - ◇ DI-Guy API
 - Version 12.0.3
 - Version 12.0.2
 - Version 12.0.1
 - Version 12.0.0
 - Graphics API Changes
- ◆ DI-Guy Version 11
 - ◇ DI-Guy API
 - Version 11.6.0
 - Graphics API Changes
 - Version 11.0.x
 - Version 11.0.4
 - Version 11.0.2
 - Version 11.0.1
 - Version 11.0.0
 - Version 10.5.1
 - Version 10.1.7
 - Version 10.1.6
 - Version 10.1.4
 - Version 10.1.3
 - Version 10.1.1
 - Version 10.1.0
 - ◇ DI-Guy Scenario
 - Version 11.5.1
 - Version 11.0.1

DI-Guy SDK Version 12.5.2 Index

- ◇ DI-Guy Motion Editor
 - Version 11.0.1
 - Version 11.0.0
- ◇ DI-Guy Character Viewer
 - Version 11
- ◇ DI-Guy for Vega Prime
 - Version 11.5.1
 - Version 11.0.1
- ◇ DI-Guy Networking
 - Version 11.0.1
- ◆ DI-Guy Applications Version 10
 - ◇ DI-Guy API 10
 - Version 10.0.10
 - Version 10.0.8
 - Version 10.0.6
 - Version 10.0.5
 - Version 10.0.4
 - Version 10.0.3
 - Version 10.0.1
 - Version 10.0.0
 - Version 9.1.5
 - Version 9.1.4
 - Version 9.1.2
 - Version 9.1.1
 - Version 9.1.0
 - ◇ DI-Guy Scenario 10
 - Version 10.0.2
 - Version 10.0.0
 - Version 9.1.3
 - Version 9.1.2
 - Version 9.1.1
 - Version 9.1.0
 - ◇ DI-Guy AI 10
 - Version 10.0.4
 - Version 10.0.3
 - Version 10.0.2
 - Version 9.1.4
 - Version 9.1.3
 - Version 9.1.2
 - Version 9.1.1
 - ◇ DI-Guy Motion Editor
 - ◇ DI-Guy Character Viewer 10
 - Version 10.0.0
 - ◇ DI-Guy for Vega Prime 10
 - Version 10.0.4
 - Version 9.1.2
 - ◇ DI-Guy Networking
 - Version 9.1.6
- ◆ DI-Guy Applications, 9.0 and Earlier
 - ◇ DI-Guy API
 - Version 9.0.4

DI-Guy SDK Version 12.5.2 Index

- [Version 9.0.1](#)
- [Version 9.0.0](#)
- [Version 8.6.3](#)
- [Version 8.6.2](#)
- [Version 8.6.1](#)
- [Version 8.5.0](#)
- [Version 8.1.0](#)
- [Version 8.0.6](#)
- [Version 8.0.5](#)
- [Version 8.0.4](#)
- [Version 8.0.3](#)
- [Version 8.0.2](#)
- [Version 8.0.1](#)
- [Version 8.0.0](#)
- [Version 7.1.20](#)
- [Version 7.1.19](#)
- [Version 7.1.18](#)
- [Version 7.1.16](#)
- [Version 7.1.14](#)
- [Version 7.1.13](#)
- [Version 7.1.12](#)
- [Version 7.1.11](#)
- [Version 7.1.10](#)
- [Version 7.1.9](#)
- [Version 7.1.8](#)
- [Version 7.1.7](#)
- [Version 7.1.6](#)
- [Version 7.1.5](#)
- [Version 7.1.4](#)
- [Version 7.1.3](#)
- [Version 7.1.2](#)
- [Version 7.1.1](#)
- [Version 7.1.0](#)
- [Version 7.0.9](#)
- [Version 7.0.8](#)
- [Version 7.0.6](#)
- [Version 7.0.5](#)
- [Version 7.0.3](#)
- [Version 7.0.2](#)
- [Version 7.0.1](#)
- [Version 7.0.0](#)
- [Version 6.1.10](#)
- [Version 6.1.9](#)
- [Version 6.1.8](#)
- [Version 6.1.7](#)
- [Version 6.1.6](#)
- [Version 6.1.5](#)
- [Version 6.1.4](#)
- [Version 6.1.3](#)
- [Version 6.1.2](#)
- [Version 6.1.1](#)

DI-Guy SDK Version 12.5.2 Index

- [Version 6.1.0](#)
- [Version 6.0.8](#)
- [Version 6.0.6](#)
- [Version 6.0.5](#)
- [Version 6.0.4](#)
- [Version 6.0.3](#)
- [Version 6.0.2](#)
- [Version 6.0.1](#)
- [Version 6.0.0](#)
- [Version 5.3.5](#)
- [Version 5.3.4](#)
- [Version 5.3.3](#)
- [Version 5.3.2](#)
- [Version 5.3.1](#)
- [Version 5.2.2](#)
- [Version 5.2.1](#)
- [Version 5.2.0](#)
- [Version 5.1.25](#)
- [Versions 5.1.0 – 5.1.24](#)

◇ [DI-Guy Scenario](#)

- [Version 9.1.4](#)
- [Version 9.1.0](#)
- [Version 9.0.2](#)
- [Version 9.0.0](#)
- [Version 8.6.3](#)
- [Version 8.6.2](#)
- [Version 8.6.1](#)
- [Version 8.5.0](#)
- [Version 8.0.3](#)
- [Version 8.0.2](#)
- [Version 8.0.0](#)
- [Version 7.1.19](#)
- [Version 7.1.16](#)
- [Version 7.1.15](#)
- [Version 7.1.14](#)
- [Version 7.1.12](#)
- [Version 7.1.11](#)
- [Version 7.1.8](#)
- [Version 7.1.7](#)
- [Version 7.1.6](#)
- [Version 7.1.4](#)
- [Version 7.1.3](#)
- [Version 7.1.2](#)
- [Version 7.1.1](#)
- [Version 7.1.0](#)
- [Version 7.0.8](#)
- [Version 7.0.5](#)
- [Version 7.0.3](#)
- [Version 7.0.1](#)
- [Version 6.1.10](#)
- [Version 6.1.9](#)

DI-Guy SDK Version 12.5.2 Index

- [Version 6.1.8](#)
- [Version 6.1.7](#)
- [Version 6.1.6](#)
- [Version 6.1.4](#)
- [Version 6.1.1](#)
- [Version 6.1.0](#)
- [Version 6.0.8](#)
- [Version 6.0.2](#)
- [Version 6.0.1](#)
- [Version 6.0.0](#)
- [Version 5.3.5](#)
- [Version 5.3.2](#)
- [Version 5.3.1](#)
- [Version 5.3.0](#)
- [Version 5.2.3](#)
- [Version 5.2.1](#)
- [Version 5.2.0](#)
- ◇ [DI-Guy AI](#)
 - [Version 9.0.1](#)
 - [Version 9.0.0](#)
 - [Version 8.6.3](#)
 - [Version 8.6.1](#)
 - [Version 8.6.0](#)
 - [Version 8.5.0](#)
- ◇ [DI-Guy Motion Editor](#)
 - [Version 9.0.1](#)
 - [Version 9.0.0](#)
 - [Version 8.6.3](#)
 - [Version 7.1.11](#)
 - [Version 7.1.0](#)
 - [Version 6.1.9](#)
 - [Version 6.1.0](#)
 - [Version 6.0.6](#)
 - [Version 6.0.0](#)
- ◇ [DI-Guy Character Viewer](#)
 - [Version 8.0.2](#)
 - [Version 6.1.9](#)
 - [Version 6.0.2](#)
 - [Version 6.0.1](#)
 - [Version 6.0.0](#)
- ◇ [DI-Guy for Vega Prime](#)
 - [Version 9.0.3](#)
 - [Version 9.0.1](#)
 - [Version 9.0.0](#)
 - [Version 8.6.3](#)
 - [Version 8.0.3](#)
 - [Version 8.0.2](#)
 - [Version 8.0.0](#)
 - [Version 7.0.8](#)
 - [Version 7.0.7](#)
 - [Version 7.0.6](#)

DI-Guy SDK Version 12.5.2 Index

- Version 7.0.3
- Version 7.0.0
- Version 6.1.11
- Version 6.1.10
- Version 6.1.9
- Version 6.1.8
- Version 6.1.1
- Version 6.0.8
- Version 6.0.7
- Version 6.0.6
- Version 6.0.5
- Version 6.0.2
- Version 6.0.1
- Version 5.3.4
- Version 5.3.3
- Version 5.2.1
- ◆ DI-Guy Data, 9.0 and Earlier
 - Version 9.0.1
 - Version 9.0.0
 - Version 8.0.0
 - Version 7.0.0
 - Version 6.1.2
 - Version 6.1.1
 - Version 6.1.0
 - Version 6.0.2
 - Version 6.0.1
 - Version 6.0.0
 - Version 5.3.5
 - Version 5.3.2
 - Version 5.3.1
 - Version 5.2.2

DI-Guy Version 12.5

DI-Guy API

Note that most changes to the DI-Guy API also apply to DI-Guy Scenario.

Version 12.5.1

- ◆ Fixed bug where customer sourced dds files failed to load because we assumed they were lz4'ed.
- ◆ Fixed bug where files with capitalized extensions eg "FLT" failed to load.
- ◆ Fixed bug where invalid xrefs caused parent bounding boxes to fail to register as possibly valid.
- ◆ Fixed hard crash when drawing empty octtree.
- ◆ Fixed matrix state glitch when drawing octtree and scene object bounds
- ◆ Fixed path planning bug in funnel code when far above the z origin.
- ◆ Fixed bug where browsing for a new scene object file name didn't force octtree recreation.
- ◆ Fixed logic bug in check visibility code.
- ◆ Fixed bug where changing appearances could cause single frame muzzle flashes and loss of character skinning matrices.
- ◆ Fixed bug where flt files were unable to have their material edited by the character viewer
- ◆ Fixed bug where changing character type caused all action beads to collapse on a path.
- ◆ Added additional guide example showing differences between similar and dissimilar character types.
- ◆ Fixed bug in built-in OpenGL renderer where fixed function mode didn't properly bind textures. Fixes textureless segmented characters rendering in Motion Editor.
- ◆ Fixed joystick not properly loading in I-Guy mode.
- ◆ Fixed ray testing against scaled scene objects.
- ◆ Fixed off by one error in link combo in character viewer.
- ◆ Fixed bug in Lua debugger. "Step over" and "Step out" buttons were not working correctly
- ◆ General improvements to code that tests `get_enabled()` and `get_dead()` on a character, the code will now also test the character's `t_out` compared to current time.
- ◆ Added `diguyAgentParams:set_influence_box_enabled()` (and `get`)
- ◆ Fixed bug with ground clamping on large scenarios.
- ◆ Fixed performance issue with receiving props over the network.
- ◆ Improved shadow pass rendering performance.
- ◆ Shader cleanup.
- ◆ Changed Graphics API shader samples to default point light `#define` to off.
- ◆ Fixed crash when executing gesture that wasn't made for a character.
- ◆ Fixed crash when loading degenerate flt file xrefs.
- ◆ Improved memory use when decompressing many dds files.
- ◆ `diguyCharacter::set_orientation()` now checks if the character has a vehicle controller and if so, it now allows the caller to hard-set the vehicle's orientation. Previously, only the vehicle controller determined this.
- ◆ `diguyCharacter::find_path()` now checks if the `character_type` contains the substring 'vehicle' and if so, the path produced will be wide enough to allow the

character to pass, based upon character's bounding-box's width.

- ◆ Added a number of try/catch blocks to deal with running out of memory while loading extremely large scenarios.
- ◆ Added code/UI to allow DI-Guy Scenario to unload textures when not used for a long time. Off by default.
- ◆ Added `diguyCharacter::trigger_shader_refresh()`, a function that triggers a refresh of the shader for a character.
- ◆ Added a number of experimental `diguyApp` memory management functions:

```

◇ diguyApp::set_threshold_to_unload_textures()
◇ diguyApp::get_threshold_to_unload_textures()
◇ diguyApp::set_texture_unload_frequency()
◇ diguyApp::get_texture_unload_frequency()
◇ diguyApp::get_unload_unused_after_scenario_load()
◇ diguyApp::set_unload_unused_after_scenario_load()

```

Version 12.5.0

CRITICAL api changes

◇ C++ callback:

```
float AltitudeFunction( diguyScenario* scenario, float x, float y, float old_z);
```

```
float AltitudeFunction( diguyScenario* scenario, float x, float y, float old_z, int * valid);
```

◇ Graphics API

- `bdiGraphicsInitGraphicsAPI.use_4x3_shader_matrices` must be set to 0 if you aren't planning on upgrading shaders to use 4x3 matrices.
- As a performance optimization we are no longer transposing our internal matrix representation in C++ before sending to OpenGL. 4x4 matrix bind should get updated to

```
glUniformMatrix4fv(m_bind_location_ufrm_link_matrices,  // uniform var loca
num_mats,  // number of matrices
!get_transposed_matrix_data(),  // whether to transpose matrix
mat_data);  // pointer to matrix data
```

New Features

- ◇ New adaptive guide allowing characters to fix dead reckoning errors with appropriate step and strafe motions. Now the default in the networking system as well as featured in the guide example.
- ◇ New texture variation system allowing for infinite variations on next-gen characters. See `diguyCharacter::set_use_texture_variations()`.
- ◇ Next-gen characters now have a number of multiple available head appearances.

- ◇ Motion engine should now be fully multi-threaded, thread control api is part of [diguyApp](#).
- ◇ New C++ documentation system, utilizing doxygen.
- ◇ Fixed matrix update issues with motion lod system and skinned characters, motion lod system can be used again with them, same issues had been happening with segmented characters as well. We recommend using multi-threaded pipeline instead of motion lods at this point.
- ◇ New ground clamp option 'g' – approximation based on ground clamping vehicle bounds
- ◇ Scene objects can now be collada DAEs.
- ◇ Added support for bump maps on scene objects.
- ◇ New multi-threaded asynchronous texture loader, controllable via scenario UI and
[diguyCharacter::set_texture_load_behavior\(\)](#) and
[diguySceneObject::set_texture_load_behavior\(\)](#)
- ◇ Collision Infrastructure Improvements:
 - Improved performance of ray tests with scenes with many props.
 - Scene Object spatial partitioning system has been overhauled allowing scene objects to be reposition without recreating the octtree.
 - Octtree memory usage has been reduced by 10%.
 - Octtree is now available for end users who's objects are not flt files, see scene_object_mesh programming example.
- ◇ New Scene Object collision control API functions:
 - [diguyScenario::create_user_defined_scene_object\(\)](#)
 - [diguySceneObject::add_vertex_to_user_defined\(\)](#)
 - [diguySceneObject::add_triangle_to_user_defined\(\)](#)
 - [diguySceneObject::add_vertex_array_to_user_defined\(\);](#)
 - [diguySceneObject::add_triangle_array_to_user_defined\(\);](#)
 - [diguySceneObject::add_triangle_to_user_defined\(\);](#)
 - [diguySceneObject::build_user_defined\(\);](#)
- ◇ New API functions:
 - [diguyCharacter::get_facefx_node_value\(\)](#)
 - [diguyCharacter::get_max_shader_quality_level\(\)](#)
 - [diguyCharacter::get_texture_load_behavior\(\)](#)
 - [diguyCharacter::get_use_texture_variations\(\)](#)
 - [diguyCharacter::set_max_shader_quality_level\(\)](#)
 - [diguyCharacter::set_texture_load_behavior\(\)](#)
 - [diguyCharacter::set_use_texture_variations\(\)](#)
 - [diguyImpact::get_impact_velocity\(float* x, float* y, float* z\)](#)
 - [diguyImpact::get_is_from_net_detonation\(\)](#)

- `diguyImpact::get_DIS_munition_septet_value(int septet_index)`
- `diguyImpact::get_DIS_site()`
- `diguyImpact::get_DIS_host()`
- `diguyImpact::get_DIS_event_number()`
- `diguyImpact::get_DIS_munition_number()`
- `diguyImpact::get_DIS_detonation_result()`
- `diguyImpact::get_DIS_burst_warhead_type()`
- `diguyImpact::get_DIS_burst_fuze()`
- `diguyImpact::get_DIS_burst_quantity()`
- `diguyImpact::get_DIS_burst_rate()`
- `diguyScenario::get_character_type_from_appearance();`
- `diguyScenario::get_num_character_types_with_appearance()`
- `diguyScenario::get_weapon_fire_effects_enabled()`
- `diguyScenario::set_default_intersection_function()`
- `diguyScenario::set_intersection_function()`
- `diguyScenario::set_particle_module_disabled()`
- `diguyScenario::set_weapon_fire_effects_enabled()`
- `diguySceneObject::get_max_shader_quality_level()`
- `diguySceneObject::get_texture_load_behavior()`
- `diguySceneObject::set_max_shader_quality_level()`
- `diguySceneObject::set_texture_load_behavior()`
- `diguy_ogl_apply_active_lights();`

- ◇ Upgraded lua/C++ binding library. Should greatly improve the readability of error warnings when lua improperly calls C++.
- ◇ `diguyCharacter::get_ai_state_string()` signature has changed to have flags for optionally including callstack info.
- ◇ `diguyCharacter::is_within_distance_n_of_member_of_group()` factors in if a character is enabled.
- ◇ Fixes to the bead constraints system (and user interface) renamed beads should keep the script and decision constraints attached to them.
- ◇ Fixed bugs where script and decision beads attached to the end of paths and stop signs didn't always trigger, or triggered later than might be expected. This might introduce slight timing changes to existing scenarios.
- ◇ Characters are now sorted before drawing, and depending on the rendering pass drawn front to back or back to front.
- ◇ Fixed bug in `diguyCharacter::force_partial_path()` when a target action bead is specified.
- ◇ Fixed crash in `diguyScenario::preload_appearance()` when a non-existent appearance is specified.
- ◇ `diguyScenario::preload_appearance()` should now preload textures for appearances that are based on material swaps.
- ◇ References to aim algorithm 7 have been removed.
- ◇ Deprecated Functions Removed:

- `diguyApp::create_load_manager()` replaced with `diguyScenario::create_load_manager()`

DI-Guy SDK Version 12.5.2 Index

- diguyApp::get_load_manager() replaced with [diguyScenario::get_character_load_manager\(\)](#)
- diguyApp::get_load_manager() replaced with [diguyScenario::get_character_load_manager\(\)](#) -> set_enabled(0)
- diguyCharacter::set_max_impact_hits() replaced with [diguyCharacter::set_weapon_max_impacts\(\)](#)
- diguyCharacter::get_paused() replaced with [diguyCharacter::get_is_network_paused\(\)](#)
- diguyCharacter::get_visible() replaced with [diguyCharacter::get_is_drawn\(\)](#)
- diguyCharacter::set_aim_param_desired_locate_time()
- diguyCharacter::get_aim_param_desired_locate_time()
- diguyCharacter::set_aim_param_seek_tolerance()
- diguyCharacter::get_aim_param_seek_tolerance()
- diguyCharacter::set_aim_interpolation_flag()
- diguyCharacter::get_aim_interpolation_flag()
- diguyCharacter::set_aim_smooth_flag()
- diguyCharacter::get_aim_smooth_flag()
- diguyCharacter::set_aim_divisions()
- diguyCharacter::set_aim_param_time_constant() replaced with set_aim_param_acquire_factor()
- diguyCharacter::get_aim_param_time_constant() replaced with get_aim_param_acquire_factor()
- diguyCharacter::set_aim_param_time_constant2() replaced with set_aim_param_new_aim_duration()
- diguyCharacter::get_aim_param_time_constant2() replaced with get_aim_param_new_aim_duration()
- diguyCharacter::set_aim_param_reset_time_constant() replaced with set_aim_param_new_aim_factor()
- diguyCharacter::get_aim_param_reset_time_constant() replaced with get_aim_param_new_aim_factor()

◊ Deprecated Callbacks Removed:

- diguyCrowd::CALLBACK_ID_NEW_FOCUS_CHARACTER
use
diguyCharacter::CALLBACK_ID_AGENT_NEW_FOCUS_CHARACTER
- diguyCrowd::CALLBACK_ID_ATTACKER_OUT_OF_TARGETS
use
diguyCharacter::CALLBACK_ID_AGENT_ATTACK_OUT_OF_TARGETS
- diguyCrowd::CALLBACK_ID_ATTACKER_NO_VISIBLE_TARGETS
use
diguyCharacter::CALLBACK_ID_AGENT_ATTACK_NO_VISIBLE_TARGETS
- diguyCrowd::CALLBACK_ID_PURSUED_CHARACTER_REACHED
use
diguyCharacter::CALLBACK_ID_AGENT_PURSUE_TARGET_REACHED
- diguyCrowd::CALLBACK_ID_WANDER_TARGET_REACHED
use
diguyCharacter::CALLBACK_ID_AGENT_WANDER_TARGET_REACHED

DI-Guy SDK Version 12.5.2 Index

- ◇ Fixed bug with T-in/T-out and review data on scene object characters.
- ◇ Updated perl header file examples with lua equivalents.
- ◇ Character viewer debug shader can now be used on scene objects
- ◇ Fixed bug on blinking characters when t-in was not 0.
- ◇ Fixed Expressive Faces pop when interacting with gaze code.
- ◇ Added the ability to pause the shadow map update and lower the update frequency.
- ◇ Added controls for adjusting the shadow map frustum and shadow acne fudge factor.
- ◇ Character culling is now based on the base link, not the position link.
- ◇ New DI-Guy Author APIs:

- [diguyAuthorInterface](#)
- [diguyAuthorVisualMaterial](#)
- [diguyAuthorVisualMultiline](#)
- [diguyAuthorVisualObjectHandle](#)
- [diguyAuthorVisualObjectHandle](#)
- [diguyAuthorVisualRegionMesh](#)
- [diguyAuthorVisualRegionPaintbrush](#)
- [diguyAuthorVisualWaypoint](#)

DI-Guy Scenario

Version 12.5.0

- ◇ New lighting system triggered by particle effects and weapons fire.
- ◇ Improvements to 3d visuals for path editing.
- ◇ Merged some subwidgets in character page.
- ◇ Improved interactive Face Expression Editing.
- ◇ Improved determinism of fast forward, and setting current time via the VCR UI.
- ◇ Streamlined performance settings. Crowd Load Manager, Load Manager tabs have been phased out, settings are now located in the performance page.
- ◇ Changing camera settings will now force a motion update on all characters.
- ◇ Property tree UI's now have slider widgets, easier to change values interactively.
- ◇ Labels above character's heads now autoscale and draw z-sorted.
- ◇ Right click menu now contains a shortcut to turning on the debug label.
- ◇ Right click on log window selection can now go to package selected or Character selected.
- ◇ Shader UI includes the ability to edit character variation texture colors.
- ◇ Physx library has been rewritten to use Physx 3.2, this is upgraded from version 2.8, Physx installer should no longer be needed.
- ◇ Fixed a number of sound looping issues.

DI-Guy AI

Version 12.5.0

- ◇ Improved turning motion system, now on by default.
- ◇ Variant system now treats the normal variant as an indicator to automatically pick an appropriate variant for the character.
Controlled by
`diguyCharacter::agent_set_auto_variant_selection()`
- ◇ System will now cycle through appropriate motions randomly.
- ◇ `LuaCharacter – sleep_ignore_messages` avoids resuming the coroutine.
- ◇ AI movement callbacks are now queued up and triggered at the end of the multi-threaded update.
- ◇ New AI State Inspector – mini lua watch window, activate via right click menu.
- ◇ Added `diguyCrowdProfile::create_character()` short cut for using a profile to create a character.
- ◇ Significantly improved speed of collision influence calculation for crowds.
- ◇ Crowd profiles have been updated to use new variation system.
- ◇ Performance improvement for adding/deleting/transforming scene objects.
- ◇ Performance improvement for intersection testing against scene objects and props.
- ◇ New API functions:
 - `diguyAgentParams::set_turning_motions_enabled()`
 - `diguyAgentParams::get_turning_motions_enabled()`
 - `diguyCharacter::agent_set_auto_variant_selection()`
 - `diguyCharacter::agent_get_auto_variant_selection()`
 - `diguyCrowdProfile::create_character()`

Graphics API Changes

Version 12.5.1

- ◇ Added functionality to support color picking intersection testing for scene graph implementations.
- ◇ Added sample code demonstrating intersection testing with `OpenSceneGraph`.
- ◇ New API Functions:
 - `diguyGraphicsShape::get_bind_color_array()`

Version 12.5.0

- ◇ Modified `diguyOsgGraphicsMesh::build()` to load appearance textures on demand instead of pre-loading all appearance textures.
- ◇ Added support for 4x3 skinned matrices, saves 25% of the bandwidth and uniform constants sent per character draw call. Now the default for OpenGL. Configurable via
`bdiGraphicsInitOgl.use_4x3_shader_matrices` and

`bdiGraphicsInitGraphicsAPI.use_4x3_shader_matrices`,
`diguy_graphics_get_osg_defaults()` and
`diguy_graphics_get_dx9_defaults()` have been added to
make it a bit easier to get the defaults set up.

◇ Added sample code demonstrating use of 4x3 matrices for transform data.

◇ Greatly expanded shader techniques API,

`diguyGraphicsShaderTechnique` can now be subclassed and
end users can create new shaders on the fly with

`diguyGraphicsShaderTechnique::find_or_create_shader_variation()`

End users can override existing functionality and return alternatively
compiled shaders depending on their own criteria, sample code is
provided demonstrating conditionally compiling in point light
support but only using shader when necessary.

◇ Fixed matrix transpose mistake in `diguyOglGraphicsLink.cpp`

◇ Cleaned up computation of local space camera.

◇ Removed use of display lists from `diguyOglGraphicsMaterial.cpp`

◇ Cleaned up `diguyOglGraphicsShaderInstance.cpp` to reflect the use
of shader techniques.

◇ Added sample code demonstrating updating lighting state for simple
point lights on characters.

◇ Added sample code demonstrating using `#defines` to control
compilation of various shader features.

◇ Added support for variation system to sample code, this is shown in
`diguyOglGraphicsShape` and `diguyOsgGraphicsShape`.

◇ Added `diguyOglGraphicsShaderTechnique` demonstrating how to
modify built-in shaders and overriding our existing shader LOD
system.

◇ New API Functions:

```
· diguyGraphicsShaderProgram::get_num_additional_feature_name()
· diguyGraphicsShaderProgram::get_additional_feature_name()
· diguyGraphicsShaderProgram::has_additional_feature()
· diguyGraphicsShaderProgram::get_vertex_shader_filename()
· diguyGraphicsShaderProgram::get_pixel_shader_filename()
· diguyGraphicsShaderProgram::get_transposed_matrix_data()
· diguyGraphicsShaderProgram::load()
· diguyGraphicsShaderProgram::get_use_4x3_shader_matrix_data()
· diguyGraphicsShaderTechnique::get_shader_by_distance()
· diguyGraphicsShaderTechnique::get_program_index_by_distance()
· diguyGraphicsShaderTechnique::get_shadow_pass_shader()
· diguyGraphicsShaderTechnique::find_or_create_shader_variation()
· diguyGraphicsShaderTechnique::pick_shader_program()
· diguyGraphicsShaderTechnique::build();
· diguyGraphicsShaderTechnique::add_additional_shader()
· diguyGraphicsShaderTechnique::get_num_additional_shaders()
· diguyGraphicsShaderTechnique::get_additional_shader_at_index()
· diguyGraphicsShape::get_variation_texture()
· diguyGraphicsShape::get_variation_texture_size()
· diguyGraphicsShape::update_variation_texture()
```

- Init params:
`bdiGraphicsInitGraphicsAPI.transpose_shader_matrices`
- Init params:
`bdiGraphicsInitGraphicsAPI.use_4x3_shader_matrices`

DI-Guy Networking Changes

Version 12.5

◇ New API Functions:

- `diguyNetInterface::set_use_adaptive_guide()`
- `diguyNetInterface::get_use_adaptive_guide()`

◇ New Custom PDU's (Note: Functionality currently limited to DIS.)

- `DIGUY_CUSTOM_PDU_ID_SET_SHAPE_VARIATION`
replicates variation texture over the network
- `DIGUY_CUSTOM_PDU_ID_EXECUTE_GESTURE`
replicates the
`diguyCharacter::execute_gesture()`
call over the network
- `DIGUY_CUSTOM_PDU_ID_CHARACTER_SCALE`
replicates the
`diguyCharacter::set_scale()` value over
the network.

- ◇ `DIGUY_CUSTOM_PDU_ID_SET_CHAR_APPEARANCE` Now
contains a second optional string for head appearances

DI-Guy Version 12.0

DI-Guy Scenario

Version 12.0.1

- ◇ Added visualization option for gaze vectors.
- ◇ Restored missing default face expressions and chain settings libraries to new.dss.
- ◇ Added display of more callback information to the Event Mapper User Interface.
- ◇ Fixed bugs with water shader.

DI-Guy AI

DI-Guy API

Note that most changes to the DI-Guy API also apply to DI-Guy Scenario.

Version 12.0.3

- ◇ Updated network model map for vehicles and also resolved issue with vehicles stuttering on paths.
- ◇ Made fire and detonation PDUs use matching event ids. Added velocities to Fire PDUs. M16 velocity is used by default for all weapons but can be overridden in munition config file.
- ◇ Fixed DI-Guy Scenario bugs in copy/paste paths, action spreadsheet.
- ◇ Fixed minor content bugs involving missing torso on insurgent.

Version 12.0.2

- ◇ Fixed `diguyApp::reload_shader_programs()` to support the graphics API. See `diguyOglGraphicsShaderProgram.cpp` for minor change: shaders that are already loaded must be detached on reload.
- ◇ Added `diguyCharacter::set_shader_lod()` and `diguyCharacter::get_shader_lod()`.
- ◇ Modified `diguyCharacter::set_face_expression()` to support a strength multiplier.
- ◇ Modified `diguyOsgGraphicsMesh::build()` to load appearance textures on demand instead of pre-loading all appearance textures.
- ◇ Call callbacks with id `CALLBACK_ID_SCENE_OBJECT_IMPACT` if `diguyScenario::trigger_detonation()` or `diguyCharacter::trigger_detonation_on_character()` are called.
- ◇ Values in `diguy_dis_constants.h` now available in scripting.
- ◇ Added a number of DIS detonation data values to `diguyImpact`; e.g., `diguyImpact::get_DIS_detonation_result()`.

Version 12.0.1

DI-Guy SDK Version 12.5.2 Index

- ◇ AI flee code now properly uses feeler callback to pick the longest available flee route when near buildings.
- ◇ Fleshed out FaceFX api, fixed a number of after action review issues, added the following functions:

- `diguyCharacter::stop_facefx_animations()`
- `diguyCharacter::get_num_facefx_nodes()`;
- `diguyCharacter::get_facefx_node_name()`;
- `diguyCharacter::find_facefx_node_index()`;
- `diguyCharacter::set_facefx_node_value()`;

- ◇ Updates to FaceFX sample content.
- ◇ Added the following functions to help enable advanced shader materials in the graphics api:

- `diguyOglGraphicsShaderProgram::bind_uniform_with_location()`
- `diguyOglGraphicsShaderProgram::bind_uniform_with_location()`
- `diguyOglGraphicsShaderProgram::get_attrib_bind_location()`
- `diguyOglGraphicsShaderProgram::get_uniform_bind_location()`

- ◇ Streamlined shader bind and unbind calls.
- ◇ Fixed issue where even with only 1 update thread was used some of the multi-threaded infrastructure was active.
- ◇ Added warnings to various `add_callback_script` functions to warn if the same callback is repeatedly added.
- ◇ Update Middle Eastern AI's to avoid perpetual memory growth.
- ◇ Changed multi-threading to only use the Qt threading framework when more than one thread is allocated to DI-Guy.

Version 12.0.0

- ◇ In API programs shadow disks are off by default. Now the "set_draw_character_shadow_disks" setting in .dss files will be ignored. To turn shadow disks on or off an explicit call to `diguyScenario::set_draw_character_shadow_disks()` needs to be made.
- ◇ Fixed `diguyCharacter::agent_set_current_params_from_profile()` so arguments are in proper order. **Note:** If you have existing calls to `agent_set_initial_params_from_profile()`, verify that you are getting proper behavior.
- ◇ Added the following functions:
 - `diguyAgentParams::set_wander_max_wander_dist()`
 - `diguyAgentParams::set_wander_sets_desired_orientation()`
- ◇ Removed the following experimental functions; should use minds instead:
 - `diguyCharacter::add_state_machine_instance()`
 - `diguyCharacter::remove_state_machine_instance()`
 - `diguyCharacter::remove_all_state_machine_instances()`

DI-Guy SDK Version 12.5.2 Index

- diguyCharacter::find_state_machine_instance()
- diguyScenario::get_num_state_machines()
- diguyScenario::create_state_machine()
- diguyScenario::destroy_state_machine()
- diguyScenario::get_state_machine_at_index()
- diguyScenario::find_state_machine()
- diguyScenario::get_num_state_machine_instances()
- diguyScenario::add_state_machine_instance()
- diguyScenario::remove_state_machine_instance()
- diguyScenario::get_this_state_machine_instance()
- diguyCrowd::all_members_add_state_machine_instance()
- diguyCrowd::all_members_remove_state_machine_instance()

◇ Removed the following deprecated functions:

- diguyApp::set_student_log_callback()
- diguyApp::student_log_print()
- diguyApp::set_user_query_yes_no_callback()
- diguyApp::user_query_yes_no()
- diguyApp::set_user_query_text_callback()
- diguyApp::set_user_query_item_callback()
- diguyApp::user_query_item()
- diguyApp::set_user_inform_callback()
- diguyApp::user_inform()

Graphics API Changes

DI-Guy Version 11

DI-Guy API

Note that most changes to the DI-Guy API also apply to DI-Guy Scenario.

Version 11.6.0

- ◊ Added `diguyRegionPaintbrushMode` enum to `diguy_constants.h`.
- ◊ Fixed problem with `agent_set_current_params_from_profile()` use in decision clauses.
- ◊ Made the following functions callable from decisions:
 - `diguyCharacter::agent_move_on()`
 - `diguyCharacter::agent_move_to_region()`
 - `diguyCharacter::agent_path_follow()`
 - `diguyCharacter::agent_resume_path_follow()`
 - `diguyCharacter::agent_set_current_focus_character()`
 - `diguyCharacter::agent_set_current_focus_group()`
- ◊ Added the following functions:
 - `diguyApp::set_region_paintbrush_mode()`
 - `diguyApp::get_region_paintbrush_mode()`
 - `diguyApp::set_munition_type_DIS_impact_septet()`
 - `diguyApp::set_munition_type_DIS_tracer_septet()`
 - `diguyCharacter::trigger_detonation_on_character()`
 - `diguyCharacter::get_vehicle_body_orientation()`
 - `diguyScenario::trigger_decision_delayed()`
 - `diguyScenario::trigger_script_delayed()`
 - `diguyScenario::get_altitude_from_point()`
 - `diguyScenario::destroy_region()`
 - `diguyScenario::save()`
 - `diguyScenario::save_as()`
 - `diguyScenario::get_filename()`
 - `diguyScenario::get_filename_without_directory()`
 - `diguyScenario::get_unsaved()`
 - `diguyScenario::set_unsaved(int unsaved)`
 - `diguyRegion::get_random_factor()`
- ◊ Remove unused playback mode enum `DIGUY_SCENARIO_PLAYBACK_MODE_RECORD`.
- ◊ Made changes to some scenario call operations. Previously the `diguyScenario::post_draw()` call had no effect; it now allows various post draw type operations to be optionally invoked. Previously these operations were automatically done at the end of `diguyScenario::draw_pass2()`.
- ◊ Added the following functions:

- `diguyScenario::set_overall_history_type()`
- `diguyScenario::get_overall_history_type()`
- `diguyScenario::manually_invoke_callbacks_now()`
- `diguyScenario::set_current_region()`
- `diguyScenario::get_current_region()`

Graphics API Changes

- ◇ Moved the following functions from `diguyGraphicsShaderInstance` to `diguyGraphicsShaderProgram`:

- `diguyGraphicsShaderInstance::bind_link_matrices()`
- `diguyGraphicsShaderInstance::bind_color_array()`

Your Graphics API subclasses will need to be updated. `bind_link_matrices()` will now potentially be called multiple times per character depending on its complexity. For example, animated faces from facefx will cause multiple matrix binds.

The signature for `diguyGraphicsTexture::bind_now()` has changed to `diguyGraphicsTexture::bind_now(diguyTextureMapType texture_type)` to support multi-texturing. Your Graphics API subclasses need to be updated.

Version 11.0.x

- ◇ Added ability to hear sounds when history is enabled and time is rewound to a point midway in the sound. Note that the function `diguyScenario::set_history_max_sound_instances()` lets users set how many sound instances to be remembered; default is 100.
- ◇ Updated where DI-Guy looks for its license file. It is looked for in this order:

- file specified by the environment variable `DIGUY_LICENSE_FILE`
- file specified by the environment variable `BDILM_LICENSE_FILE`
- file specified by the environment variable `LM_LICENSE_FILE`
- `license.dat` in DI-Guy installation `flexlm` subdir

- ◇ Turned off character shadow disks by default when using SDK; they're still on by default in DI-Guy Scenario.

Version 11.0.4

- ◇ Resolved issues with wheeled vehicles on paths, `force_action`, and `resume_path`

DI-Guy SDK Version 12.5.2 Index

- Problem with vehicles not resuming their path and remaining motionless after `force_action("still")`
- Problem with vehicles resuming path and wheels not spinning
- Problems with position discontinuities when `force_action("still")` is applied to a vehicle on a path

Version 11.0.2

- ◇ Updated where DI-Guy looks for its license file. It is looked for in this order:
 - file specified by the environment variable `DIGUY_LICENSE_FILE`
 - file specified by the environment variable `BDILM_LICENSE_FILE`
 - file specified by the environment variable `LM_LICENSE_FILE`
 - `license.dat` in DI-Guy installation `flexlm` subdir
- ◇ Turned off character shadow disks by default when using SDK; they're still on by default in DI-Guy Scenario.
- ◇ Removed use of `glCallList` in materials for the Graphics API sample code and the internal OpenGL renderer. Call lists are deprecated and have more overhead than just calling material functions directly.
- ◇ Added early out sample code for materials to graphics API; end users should now have the tools to skip unnecessary material binds.
- ◇ Added a new geode/mesh merge as part of the `bdg` creation; should cut number of meshes and draw calls in half for most of the non-skinned characters.
- ◇ Added new draw stats profiler to improve feedback on pipeline bottlenecks. Profiler now available in Character Viewer. (Press `Ctrl-P` a few times.)
- ◇ Added linear draw pipeline init argument.
- ◇ Updated Benchmark sample program with new options and statistics.
- ◇ Fixed bug where get link positions would not take character scale into effect.
- ◇ Added the following functions to DI-Guy Graphics API:
 - `diguyGraphicsLink::get_link_transformation_matrix()`
 - `diguyGraphicsLink::begin_character_draw()`
 - `diguyGraphicsLink::end_character_draw()`
 - `diguyGraphicsLink::get_use_linear_traversal_draw()`
 - `diguyGraphicsLink::set_use_linear_traversal_draw()`
 - `diguyGraphicsTexture::get_last_bound_texture()`
 - `diguyGraphicsTexture::clear_last_bound_texture()`
- ◇ Added sample code to the OpenGL graphics API reference implementation demonstrating the use of the link local transformation matrix to simplify the number of OpenGL function made.

DI-Guy SDK Version 12.5.2 Index

- ◇ Added sample code to the OpenGL graphics API reference implementation to demonstrate using `diguyGraphicsTexture::get_last_bound_texture()` to minimize state changes.
- ◇ Added the following functions to DI-Guy OpenGL rendering API:

- `diguy_ogl_get_num_frame_stats()`
- `diguy_ogl_frame_stat_name()`
- `diguy_ogl_get_frame_stat()`;
- `diguy_ogl_clear_all_frame_stats()`

Version 11.0.1

- ◇ Added the following functions to DI-Guy Graphics API:

- `diguyGraphicsMaterial::get_last_bound_material()`
- `diguyGraphicsMaterial::clear_last_bound_material()`

- ◇ Added sample code to the OpenGL Graphics API reference implementation to demonstrate using `diguyGraphicsMaterial::get_last_bound_material()` to minimize state changes.
- ◇ Fixed binary geometry serialization bug on x64 machines.
- ◇ Added `diguyScenario::is_valid_character_group_pointer()`

Version 11.0.0

- ◇ Removed deprecated functions:

- `diguyChainSimulation::set_connection_character()`

- ◇ New APIs:

- `diguyFormation`
- `diguyVehicleController`
- `diguyViewButtonPanel` (DI-Guy Scenario only)

Version 10.5.1

- ◇ Removed deprecated functions:

- `diguyScenario::set_tin()`
- `diguyScenario::get_tin()`
- `diguyScenario::trigger_script_event()`
- `diguyScenario::add_custom_appearance()`
- `diguyScenario::add_custom_appearance_based_on_existing_appearance()`
- `diguyScenario::add_named_callback()`
- `diguyScenario::remove_named_callback()`
- `diguyScenario::call_named_callback()`
- `diguyScenario::set_iguy_character()`

Version 10.1.7

◇ Added the following functions:

- `diguyCharacter::set_max_unprocessed_impacts()`
- `diguyCharacter::get_max_unprocessed_impacts()`
- `diguyScenario::set_max_unprocessed_impacts()`
- `diguyScenario::get_max_unprocessed_impacts()`
- `diguyCharacter::reset_shader_program_to_default()`
- `diguySceneObject::reset_shader_program_to_default()`

Version 10.1.6

◇ Added functions to `diguyViewCamera` and `diguyViewCameraSettings`:

- `diguyViewCamera::set_speed_fov_degrees_per_wheel_click()`
- `diguyViewCamera::get_speed_fov_degrees_per_wheel_click()`
- `diguyViewCameraSettings::set_speed_fov_degrees_per_wheel_click()`
- `diguyViewCameraSettings::get_speed_fov_degrees_per_wheel_click()`

◇ Added flag to `diguyScenario` that explicitly enables or disables tout ("T Out") for the scenario, instead of using the arbitrary value of 60000 seconds to mean tout should have no effect.

◇ Added flag similar to above that enables or disables character tout.

- `diguyScenario::set_tout_enabled()`
- `diguyScenario::get_tout_enabled()`
- `diguyCharacter::set_tout_enabled()`
- `diguyCharacter::get_tout_enabled()`

◇ Added the following functions:

- `diguyCharacter::get_muzzle_link()`
- `diguyCharacter::get_muzzle_offset()`
- `diguyCharacter::stop_appearance_effect()`
- `diguyCharacter::set_weapon_tracers_enabled()`
- `diguyCharacter::get_weapon_tracers_enabled()`
- `diguyCharacter::set_weapon_shell_eject_enabled()`
- `diguyCharacter::get_weapon_shell_eject_enabled()`
- `diguyCharacter::set_weapon_smoke_enabled()`
- `diguyCharacter::get_weapon_smoke_enabled()`

Version 10.1.4

◇ Added

`diguyCharacter::CALLBACK_ID_AGENT_MIND_CURRENT_STATE_CHANGED`.

◇ Added the following functions for getting and setting mind data.

- `diguyCharacter::get_mind_current_state_name()`
- `diguyCharacter::set_mind_field_as_string()`
- `diguyCharacter::set_mind_field_as_float()`
- `diguyCharacter::get_mind_field_as_float()`
- `diguyCharacter::set_mind_field_as_int()`
- `diguyCharacter::get_mind_field_as_int()`

- ◊ Added the following functions for setting agent params, and other agent behavior.

- `diguyCharacter::agent_set_initial_params_from_profile()`
- `diguyCrowd::set_initial_params_from_profile()`
- `diguyAgentParams::set_move_on_t_min()`
- `diguyAgentParams::get_move_on_t_min()`
- `diguyAgentParams::set_move_on_t_max()`
- `diguyAgentParams::get_move_on_t_max()`
- `diguyCharacter::agent_move_on()`

Version 10.1.3

- ◊ Added the following blink functions to `diguyCharacter`.

- `diguyCharacter::set_automatic_blink_duration_in()`
- `diguyCharacter::get_automatic_blink_duration_in()`
- `diguyCharacter::set_automatic_blink_duration_out()`
- `diguyCharacter::get_automatic_blink_duration_out()`
- `diguyCharacter::set_automatic_blink_target_in()`
- `diguyCharacter::get_automatic_blink_target_in()`
- `diguyCharacter::set_automatic_blink_target_out()`
- `diguyCharacter::get_automatic_blink_target_out()`
- `diguyCharacter::set_automatic_blinks_enabled()`
- `diguyCharacter::get_automatic_blinks_enabled()`

- ◊ Added the following gesture functions to `diguyCharacter`.

- `diguyCharacter::execute_table_gesture()`
- `diguyCharacter::set_executing_table_gesture_desired_action()`
- `diguyCharacter::end_executing_gesture()`
- `diguyCharacter::abort_executing_gesture()`

- ◊ Added the following misc functions to `diguyCharacter`.

- `diguyCharacter::get_gaze_is_active()`
- `diguyCharacter::get_lpoint_is_active()`

Version 10.1.1

- ◊ Added the following functions to `diguyCharacter`.

- `diguyCharacter::get_is_aiming()`

Version 10.1.0

- ◊ Added the following functions for improved LOD handling.

- `diguyScenario::set_scale_graphics_lod_ranges_from_view_settings()`
- `diguyScenario::get_scale_graphics_lod_ranges_from_view_settings()`
- `diguyLoadManager::set_scale_zone_extents_from_view_settings()`
- `diguyLoadManager::get_scale_zone_extents_from_view_settings()`

- ◊ Added the following functions to `diguyCharacter`.

- `diguyCharacter::set_graphics_ptr()`
- `diguyCharacter::get_num_pose_overrides()`
- `diguyCharacter::get_pose_override_at_index()`

DI-Guy Scenario

Version 11.5.1

- ◇ Fixed intersection test issue with scene objects
- ◇ Fixed bogus fog changed notification
- ◇ Improved sound history recording and playback

Version 11.0.1

- ◇ Fixed crash with using `qt_invoke` without `lqt` loaded.

DI-Guy Motion Editor

Version 11.0.1

- ◇ Fixed Motion Arc extended info handling.
- ◇ Fixed issue with the poser not being available on some skinned character links.

Version 11.0.0

- ◇ Initial support for handling skinned characters.

DI-Guy Character Viewer

Version 11

- ◇ Added Shadow and normal previewer.
- ◇ Added LOD picker.

DI-Guy for Vega Prime

Version 11.5.1

- ◇ Initial support for `vegaPrime 5.x`.
- ◇ fixed crash on exit using some modules.
- ◇ Changed `setScriptPerlEnable` default from `true` to `false`.

Version 11.0.1

- ◇ Initial support for `vegaPrime 4.x`.

DI-Guy Networking

Version 11.0.1

- ◇ Fixed a number of bugs with transmitting gazing and aiming behavior over network.
- ◇ Improvements to log messages.

DI-Guy Applications Version 10

DI-Guy API 10

Note that most changes to the DI-Guy API also apply to DI-Guy Scenario.

Version 10.0.10

- ◇ Impact processing on characters now considers mind wakeup callbacks when deciding how the impact affects the character, in addition to the straight callbacks that are considered now.
- ◇ Added functions:

- `diguyAgentParams::set_vehicle_base_influence_box()`
- `diguyAgentParams::get_vehicle_base_influence_box()`

- ◇ Added the following `diguyCharacter` callbacks that report when an agent is being push around by other characters or vehicles.

- `diguyCharacter::CALLBACK_ID_AGENT_NON_ZERO_REPULSION`
- `diguyCharacter::CALLBACK_ID_AGENT_ZERO_REPULSION`

Version 10.0.8

- ◇ Added merging of regions during scenario merge.
- ◇ Fixed merging of crowds to include merging of crowd behavior regions.
- ◇ Added `diguyScenario::CALLBACK_ID_PLAYBACK_MODE_CHANGED`.
- ◇ Added `diguy_set_diguy_base_dir_path()` function.
- ◇ Added `remove_on_scenario_load` argument to most `add_default_*_callback()` functions; allows for "permanent" callbacks to be added that persist through scenario loads. The default is

1. meaning remove the callback when a scenario is loaded.

Version 10.0.6

- ◇ Fixed visual issues of skinned appearances when the character is scaled or is using a motion LOD greater than 1.
- ◇ Fixed issue where default scenario callbacks were not being called.
- ◇ Added functions:

- `diguyCharacter::set_graphics_api_node_ptr()`
- `diguyCharacter::get_graphics_api_node_ptr()`
- `diguyScenario::set_graphics_api_node_ptr()`
- `diguyScenario::get_graphics_api_node_ptr()`

Version 10.0.5

- ◇ Fixed hard crashes with on-screen labels outside of the ogl pipeline.
- ◇ Fixed issues with octtree creation and visibility checks.

DI-Guy SDK Version 12.5.2 Index

- ◇ Fix for vc8 mdd examples using lua.
- ◇ Added missing header file for DI-Guy Bench Marking example.
- ◇ Added infrastructure for hooking up callbacks so visibility tests can work in the SDK. See the following functions:

- `diguyScenario::set_default_point_los_function()`
- `diguyScenario::set_default_character_los_function()`
- `diguyCharacter::set_point_los_function()`
- `diguyCharacter::get_point_los_function()`
- `diguyCharacter::set_character_los_function()`
- `diguyCharacter::get_character_los_function()`

Version 10.0.4

- ◇ Fixed bug where green boxes were appearing at the origin in some DI-Guy Graphics API examples.
- ◇ Characters with skinned appearances specified will switch to their default unskinned appearance on platforms without skinning support.
- ◇ Added `diguyScenario::get_altitude_function()` and `diguyScenario::set_altitude_function()` mainly needed for force local path via the API.
- ◇ Added `bdiGraphicsInitGraphicsAPI::characters_have_world_space_triangles` and `bdiGraphicsInitGraphicsAPI::scene_objects_have_world_space_triangles` function allow octtree to be built from internally loaded scene objects. **Note:** this functionality requires that `diguyGraphicsMesh::free_loader_memory()` not be called.

Version 10.0.3

- ◇ Changed `diguyScenario::set_default_lod_ranges()` to accept "all" for the character type. Passing "all" will set the default LOD ranges for all character types, not just one specific one.
- ◇ Added the following functions to DI-Guy Graphics API for more flexibility in handling graphics LODs.

- `diguyGraphicsShape::set_graphics_lod_ranges()`
- `diguyGraphicsShape::set_graphics_lod()`
- `diguyGraphicsShape::get_graphics_lod()`

Version 10.0.1

- ◇ Fixed bugs with intersecting characters added to the octtree.
- ◇ Fixed bugs where custom directory env variable was being ignored when looking up lua scripts.
- ◇ Added functions `diguyCharacter::create_complete_pose_override()` and `diguyCharacter::destroy_complete_pose_override()`.
- ◇ Fixed bug where OGL skinned characters were missing some polygons; please see graphics API samples if you are seeing this bug. Your VBO class may need slight changes.

Version 10.0.0

- ◇ Added support for character "type maps". Type maps add extra data to characters that make it easier to decide which character type and appearance best match the desired role for characters to be added to the scenario. See the class `diguyCharacterTypeMap`, and information about the "Appearance Wizard" for DI-Guy Scenario.
- ◇ Added the following functions to `diguyCharacter` API:

- `diguyCharacter::get_nearest_character_type_map( )`

- ◇ The DI-Guy Graphics API example directories `osg_12_examples` and `osg_22_examples` have been removed. They have been replaced with `osg_2_examples`.
- ◇ The `osg_2_examples` reference implementation now has support for skinned appearances. It uses the DI-Guy built-in geometry loader instead of OSG's, as DI-Guy has some specific requirements for loaded skinned geometry.
- ◇ `diguyScenario::add_callback( )` and `diguyScenario::add_callback_script( )` have been modified with an addition argument that specifies whether the callback should remain after a new scenario has been loaded. The previous behavior was for the callback to not be removed, which could cause problems. Users will now need to explicitly request that the callback remain.

Version 9.1.5

- ◇ The ability to show bar graph information has been added to the `diguyViewLabel` class. See `diguyViewLabel::enable_bar_graph( )`.
- ◇ Added the following functions to `diguyCharacter` API:

- `diguyCharacter::set_weapon_munition_type_override( )`
- `diguyCharacter::get_weapon_munition_type_override( )`
- `diguyCharacter::get_weapon_muzzle_munition_type( )`
- `diguyCharacter::get_default_muzzle_munition_type( )`

- ◇ Added the following functions to DI-Guy Graphics API:

- `diguyGraphicsLink::post_build( )`
- `diguyGraphicsLink::set_shader_instance( )`
- `diguyGraphicsMaterial::get_is_unlit( )`

- ◇ Added new DI-Guy Graphics API classes: `diguyGraphicsShaderProgram` and `diguyGraphicsShaderInstance`.
- ◇ Added `time_to_live` to `network.cfg` file.
- ◇ Removed the following experimental functions:

- `diguyScenario::load_shader( )`
- `diguyScenario::reload_shader_programs( )`

DI-Guy SDK Version 12.5.2 Index

- ◇ Added new DI-Guy functions to allow SDK users to intercept fire requests and plug in their own intersection test; see `diguyScenario::set_fire_weapon_intersection_function()` for more info.

Version 9.1.4

- ◇ Added the following functions to `diguyCharacter` API:

- `diguyCharacter::set_initial_path()`
- `diguyCharacter::get_initial_path()`

- ◇ The way that weapon sounds are handled has been changed.

- The following functions have been deprecated:

- `diguyCharacter::set_default_weapon_sound()`
- `diguyCharacter::get_default_weapon_sound()`
- `diguyCharacter::set_default_weapon_sound_gain()`
- `diguyCharacter::get_default_weapon_sound_gain()`

- The above deprecated functions are replaced by:

- `diguyCharacter::set_weapon_sound_override()`
- `diguyCharacter::get_weapon_sound_override()`
- `diguyCharacter::set_weapon_sound_gain_override()`
- `diguyCharacter::get_weapon_sound_gain_override()`

- ◇ Added the following `diguyCharacter` callbacks to make it easier to tie events to editing characters in the UI.

- `diguyCharacter::CALLBACK_ID_UI_MOUSE_DOWN`
- `diguyCharacter::CALLBACK_ID_UI_MOUSE_UP`

- ◇ Added `diguyViewPainter` class that allows the drawing of lines and

other primitives in DI-Guy Scenario 3D view windows. For access to this object call `diguyApp::get_painter()`.

- ◇ Added `diguyScenario::CALLBACK_ID_POST_DRAW`, allowing the scripting engine a chance to draw in the 3D view in DI-Guy Scenario. Characters with minds are also given the opportunity to draw; see the default `luaCharacter` script.

Version 9.1.2

- ◇ Added `diguyImpact::get_impact_type()` and `diguyImpact::get_impact_type_string()` which provide information about how the impact was generated. New accessors for querying normal information are also available.
- ◇ Added new Lua Scripting module:

DI-Guy SDK Version 12.5.2 Index

- The Lua module has all the functionality of the current Perl module.
- Lua is popular in the game development industry for it's speed and efficiency. Lua script execution is much less performance intensive than a similar Perl script.
- Lua is the basis for the new DI-Guy AI Mind framework.

◇ Scripts are no longer limited in length (was 32K).

◇ Deprecated functions:

- `diguyCharacter::get_base_link()`
replaced with
`diguyCharacter::get_position_link()`

◇ Added the following `diguyCharacter` callbacks to make it easier to tie events to editing characters in the UI.

- `diguyCharacter::CALLBACK_ID_UI_PRE_TRANSLATION`
- `diguyCharacter::CALLBACK_ID_UI_POST_TRANSLATION`

Version 9.1.1

◇ Added the following function to `diguyCharacter` API:

- `diguyCharacter::get_random_factor()`

◇ Added the following functions to `diguyCharacter` API for getting and setting character variables with a compact function name:

- `diguyCharacter::set_ivar()`
- `diguyCharacter::get_ivar()`
- `diguyCharacter::set_fvar()`
- `diguyCharacter::get_fvar()`
- `diguyCharacter::set_svar()`
- `diguyCharacter::get_svar()`

◇ Sound Module Improvements:

- Adding looping flag to `diguyCharacter::play_3d_sound()` and `diguyCharacter::play_sound()` allows sounds to repeat until explicitly stopped with `diguyCharacter::stop_sound()` or `diguyCharacter::stop_all_sounds()`
- 3D Sounds should now Doppler shift depending on the velocity of the emitter and listener.
- More a number of `diguySound` functions have been added to allow querying a sounds properties.

DI-Guy SDK Version 12.5.2 Index

- ◇ Fixed problem with the OpenFlight geometry loader. Light point records were causing the loader to fail.
- ◇ Added two new `diguyCharacter` callback ids:
 - `diguyCharacter::CALLBACK_ID_POST_CREATE_GEOMETRY`
 - `diguyCharacter::CALLBACK_ID_PRE_DESTROY_GEOMETRY`
- ◇ Added new plugin function, `post_reset()`. Allows plugins to do reset operations at the end of a scenario reset, instead of only at the beginning.

Version 9.1.0

- ◇ Added the following functions to `diguyCharacter` API:
 - `diguyCharacter::get_current_traveled_path_is()`
 - `diguyCharacter::get_current_appearance_is()`
 - `diguyCharacter::get_current_head_appearance_is()`
 - `diguyCharacter::execute_table_gesture()`

DI-Guy Scenario 10

Note that most changes to the DI-Guy API also apply to DI-Guy Scenario.

Version 10.0.2

- ◇ Fixed the appearance of state functions in the code browser.

Version 10.0.0

- ◇ Added the "Appearance Wizard" to places where character type and appearance can be selected. The Appearance Wizard provides an interactive UI for selecting desired attributes of the character to be created, such as category, age, sex, race, military branch, carried object, etc.
- ◇ Character types and appearances are now marked as "Recommended" and "Not Recommended". Many older character types and appearances have been nominally replaced with newer versions which should be preferred over the older ones. By default DI-Guy Scenario will show only the preferred ones, but can be changed to show all of them. Scenarios that currently use now not-recommended character types and appearances will still work as before.

Version 9.1.3

- ◇ Improvements to autocomplete, including `DIGUY_*` constants as well as dynamic analysis of lua objects.

Version 9.1.2

- ◇ Added new documentation viewer, accessible under the help menu.

Version 9.1.1

DI-Guy SDK Version 12.5.2 Index

- ◇ When selecting which agents' parameters are going to be edited, multi-select can now be used.
- ◇ Replaced Scenario Elements, Character Elements, View Settings, Event Handlers, and DI-Guy AI Elements windows with a single revamped Scenario Elements window that has collapsible tab groups.
- ◇ Removed experimental, incomplete "Merge Blitz" capability.

Version 9.1.0

- ◇ Script editing now uses an editing component based on Scintilla, this provides syntax highlighting/coloring, line numbering, basic auto-complete, and code folding. Currently supported control keys:
 - Ctrl '{' or Ctrl '}' – jump to matching brace
 - Ctrl f – find dialog
 - F3 – find again
- ◇ Added a number of ui improvements:
 - Beads now display the time they occur instead of just an index.
 - The Action Spreadsheet is available in the character elements page.
 - A number of element types can now be sorted in their selector view at the top of the page. Most widgets can be sorted by index and name. Some add the ability to be sorted by time, appearance, character type, etc.
 - The path page has a sorted list of all beads on a path.

DI-Guy AI 10

Version 10.0.4

- ◇ Added new AI_Populate SDK example; demonstrates populating the environment with lua minds based Agents via the SDK.

Version 10.0.3

- ◇ Improved behavior of paintbrush far from origin and at different heights.
- ◇ Added bright red log/package indicator on lua syntax errors.
- ◇ Improved behavior of code browser recompile button, pops up log window on syntax errors.

Version 10.0.2

- Fixed hard crash in path planner when pursuing with no behavior region set.
- Fixed a number of bugs with agents colliding with props.
- Fixed bug with crowd awareness code far from the origin.

Version 9.1.4

- ◇ Significant improvements to A* path planning performance, 10x speed up in most cases.

Version 9.1.3

- ◊ Improvements to path planning API, added the following functions to `diguyCharacter` API for navigating the world

- `diguyCharacter::agent_move_to_point()`
- `diguyCharacter::agent_move_to_point_via_subregions()`
- `diguyCharacter::agent_move_to_region()`
- `diguyCharacter::agent_move_to_region_via_subregions()`

Version 9.1.2

- ◊ New Lua based mind framework:

- New Lua Mind Editor in UI
- The crowd/people blitzers can now create characters with minds.

- ◊ Added the following functions to `diguyCharacter` API:

- `diguyCharacter::agent_get_crowd()`
- `diguyCharacter::agent_flee_crowd()`

- ◊ Added

`diguyCharacter::CALLBACK_ID_AGENT_VEHICLE_COLLISION`;
this callback is triggered on a vehicle when a vehicle hits an agent.

- ◊ Added `diguyRegion`, an API wrapper for painted meshes.

Retrievable with the function

`diguyScenario::find_region()`.

- ◊ Vehicle bounds are now smoothed when the vehicle changes speed.

- ◊ Added simple gazing model into agent focus behavior.

- ◊ Added Idle Behavior, agent will stop moving and be responsive to focus and simple requests but won't spontaneously move.

- ◊ Added `diguyCharacter::agent_begin_behavior()` allows starting a behavior by name. Typically some of the more elaborate functions are preferred, but this can be useful for idling and such.

- ◊ Changes to None behavior. Agent will attempt to do less and clear out some transient state to make the behavior of the behavior more consistent.

Version 9.1.1

- ◊ The following `diguyCrowd` callbacks are deprecated, and have been replaced with `diguyCharacter` equivalents.

- `diguyCrowd::CALLBACK_ID_NEW_FOCUS_CHARACTER`
- `diguyCrowd::CALLBACK_ID_ATTACKER_NO_VISIBLE_TARGETS`
- `diguyCrowd::CALLBACK_ID_ATTACKER_OUT_OF_TARGETS`
- `diguyCrowd::CALLBACK_ID_PURSUED_CHARACTER_REACHED`
- `diguyCrowd::CALLBACK_ID_WANDER_TARGET_REACHED`
- `diguyCrowd::agents_flee_crowd()`

DI-Guy SDK Version 12.5.2 Index

- ◇ Added the following `diguyCharacter` callbacks to make it easier to detect important events in agent behavior.

- `diguyCharacter::CALLBACK_ID_AGENT_NEW_BEHAVIOR`
- `diguyCharacter::CALLBACK_ID_AGENT_NEW_FOCUS_CHARACTER`
- `diguyCharacter::CALLBACK_ID_AGENT_ATTACK_OUT_OF_TARGETS`
- `diguyCharacter::CALLBACK_ID_AGENT_ATTACK_NO_VISIBLE_TARGETS`
- `diguyCharacter::CALLBACK_ID_AGENT_PURSUE_TARGET_REACHED`
- `diguyCharacter::CALLBACK_ID_AGENT_PURSUE_TARGET_LOST`
- `diguyCharacter::CALLBACK_ID_AGENT_MINGLE_TARGET_REACHED`
- `diguyCharacter::CALLBACK_ID_AGENT_MINGLE_TARGET_LOST`
- `diguyCharacter::CALLBACK_ID_AGENT_WANDER_TARGET_REACHED`
- `diguyCharacter::CALLBACK_ID_AGENT_WANDER_TARGET_LOST`
- `diguyCharacter::CALLBACK_ID_AGENT_FLEE_AREA_LEFT`
- `diguyCharacter::CALLBACK_ID_AGENT_FLEE_AREA_ENTERED`
- `diguyCharacter::CALLBACK_ID_AGENT_TRAVEL_WAYPOINT_REACHED`
- `diguyCharacter::CALLBACK_ID_AGENT_TRAVEL_FORWARD_DEST_REACHED`
- `diguyCharacter::CALLBACK_ID_AGENT_TRAVEL_BACKWARD_DEST_REACHED`

- ◇ DI-Guy AI agents can now flee from multiple characters, groups, and locations instead of just one.

- ◇ Functions

`diguyCharacter::agent_set_current_params_from_profile()`

and

`diguyCrowd::set_current_params_from_profile()`

have a new argument, `retain_focus_objects`. Setting this to 1 allows the specified to be loaded without changing the agents' current focus character or group.

DI-Guy Motion Editor

- ◇ Added support for filtering the action table editor, eases editing of large action tables.
- ◇ Added code to reverse motions.
- ◇ Grabbing a pose on a Linear Offset should interpolate properly.
- ◇ Motion arc list and action table motion list are now in sync.

DI-Guy Character Viewer 10

Version 10.0.0

- ◇ Added "Appearance Wizard" button to make selecting an appropriate character type and appearance easier.

DI-Guy for Vega Prime 10

Version 10.0.4

- ◇ Initial release compiling with vp 3.0.

Version 9.1.2

- ◇ Added support for turning on DI-Guy AI from vpDiguySettings UI. DI-Guy AI functionality is limited inside of Vega Prime
- ◇ Added support for turning on Lua scripting module from the the UI.
- ◇ Fixed long standing issue with character's having link offsets applied multiple times. (Also in 9.0.4)

DI-Guy Networking

Version 9.1.6

- ◇ Fire PDUs position should now be the muzzle of the gun, the entity type should match the munition type fired, and the velocity should be the normalized direction of the muzzle.
- ◇ Fixed issue where agents on non-flat terrains were sending erroneous z values.

DI-Guy Applications, 9.0 and Earlier

DI-Guy API

Note that most changes to the DI-Guy API also apply to DI-Guy Scenario.

Version 9.0.4

- ◇ Added new argument to `diguyCharacter::force_action()` that allows the forced action to begin partway into the action.
- ◇ Added the following functions to `diguyCharacter` API:
 - `diguyCharacter::get_most_recent_forced_action()`
 - `diguyCharacter::get_most_recent_forced_action_t()`
 - `diguyCharacter::get_most_recent_forced_action_t_offset()`

Version 9.0.1

- ◇ Added new guide example, see `$(DIGUY)\programming_examples\diguy_ogl\guide_example`
- ◇ Added the following functions to `diguyCharacter` API:
 - `diguyCharacter::get_distance_to_impact()`
 - `diguyCharacter::set_is_scene_object()`

DI-Guy SDK Version 12.5.2 Index

- `diguyCharacter::get_is_scene_object()`
- `diguyCharacter::set_apply_actor_scale_to_action_bead_xy_tr`
deprecated; use
`diguyCharacter::set_apply_actor_scale_to_action_bead_trave`
instead
- `diguyCharacter::get_apply_actor_scale_to_action_bead_xy_tr`
deprecated; use
`diguyCharacter::get_apply_actor_scale_to_action_bead_trave`
instead

- ◇ Manually invoking a `diguyCharacter` perl script callback via
`diguyCharacter::manually_invoke_event_handler()`
will now have the `$callback_object` variable properly set.
- ◇ Minor munition system improvements:

- Munitions sound effects now include sound gain.
- Added
`diguyScenario::trigger_detonation()`,
a function for creating a detonation at an arbitrary
xyz location.
- Fixed bug with location of munition particle effects.

- ◇ Added new Munition system demonstration; see
`AI_Munitions_Demo.dss` in scenarios directory.
- ◇ Improved default 3d sound behavior; prior default 3d sound falloff
was much too aggressive. Future DI-Guy releases will make more of
these values tuneable.

Version 9.0.0

- ◇ Promoted the following functions from experimental to official
status:

- `diguyScenario::find_character_at_screen_coords()`
- `diguyScenario::get_impact_at_screen_coords()`
- `diguyScenario::get_intersection_at_screen_coords()`
- `diguyScenario::force_octtree_generation()`
- `diguyScenario::intersect_static_geometry()`
- `diguyScenario::add_character_to_octtree()`
- `diguyScenario::remove_character_from_octtree()`
- `diguyCharacter::set_apply_actor_scale_to_action_bead_xy_tr`
- `diguyCharacter::get_apply_actor_scale_to_action_bead_xy_tr`

- ◇ Added functions to allow user to specify custom feeler functions:

- `diguyScenario::set_default_agent_feeler_function()`
- `diguyCharacter::agent_set_feeler_function()`
- `diguyCharacter::agent_get_feeler_function()`

- ◇ Fixed more issues with recent changes to reduce path memory
overhead.

DI-Guy SDK Version 12.5.2 Index

- ◇ Changing current appearance will now complain less if the resulting change doesn't require recompilation of path data.
- ◇ `diguy_dis_constants.h` now includes custom PDU enumeration.
- ◇ Changes made to aim algorithm 8 parameter functions in an effort to make them more descriptive. Note that only the set functions are listed; the corresponding get functions have changed, too.

- `diguyCharacter::set_aim_param_time_constant( )`
deprecated; use
`diguyCharacter::set_aim_param_acquire_factor( )`
instead
- `diguyCharacter::set_aim_param_reset_time_constant( )`
deprecated; use
`diguyCharacter::set_aim_param_new_aim_factor( )`
instead
- `diguyCharacter::set_aim_param_time_constant2( )`
deprecated; use
`diguyCharacter::set_aim_param_new_aim_duration( )`
instead

Version 8.6.3

- ◇ Fixed some issues with recent changes to reduce path memory overhead.

Version 8.6.2

- ◇ Changes made to support multiple active formations. New arguments added to `diguyCharacter::call_formation( )` and `diguyCharacter::break_formation( )` to support multiple formations. Default behavior remains the same as before.
- ◇ Significant memory savings when dealing with characters with long paths.
- ◇ Added new argument to `diguyCharacter::set_current_appearance( )` function: `allow_actor_change`. Previously changes to a character's appearance that would result in an actor (size of character) change were not allowed. Passing 1 (the default) for this argument now allows such a change.

Version 8.6.1

- ◇ Added function
`diguyCharacter::get_current_traveled_path( )`.
- ◇ Improved logic for picking dead action.
- ◇ Removed deprecated functions:
 - `diguyLoadManager::set_max_update_period( )`
 - `diguyLoadManager::set_nonvisible_update_rate( )`
 - `diguyLoadManager::set_zone_bounds( )`
 - `diguyLoadManager::set_zone_parameters( )`
- ◇ Improved DIS/HLA Networking:

DI-Guy SDK Version 12.5.2 Index

- Added support for Flat Earth and Geocentric coordinates.
- Coordinate system changes can now be applied while the system is connected to the network.
- DI-Guy characters can now have different Munition types which are sent over the network as part of the detonation pdu.

◇ Improved Munition Support:

Heavy rewrite/solidification of our munition and detonation code.

- Characters now have a munition type, defaults to "m16".
- Munition definitions are stored in the `munition_type.cfg` file. They have a number of required and optional fields including septet, radius, and special effects to create when triggered.
- The munition name and radius is now part of `diguyImpact` structure. This allows the end user to build damage models in scripts that are based on the munition type of an impact, along with fall-off effects based on distance from impact location.
- Characters hit by broadcast detonations will now call the `CALLBACK_ID_IMPACT` callback if they have one, this emulates what happens when characters are shot by other DI-Guys. If they are members of a crowd and the crowd has an impact callback that will be called, if no impact callback exists they will die.
- Crowds should now react to remote detonations.
- Impacting a scene object will now transmit a detonation.

Version 8.5.0

- ◇ Many new function calls available from decisions.
- ◇ Changed default aim algorithm from 7 to 8. To change it back to 7, call `diguyCharacter::set_aim_algorithm()`, passing 7 as the `aim_algorithm` argument.
- ◇ Improved `diguyCharacter::check_character_visibility()`.

DI-Guy SDK Version 12.5.2 Index

Character will now use the cervical joint as the source point, and test multiple positions on the target.

- ◇ Promoted `diguyCharacter` version 8 aim algorithm functions from experimental to official.
- ◇ Characters that are flagged as scene object are no longer baked into the world octtree. Editing them no longer requires an expensive octtree rebuild.
- ◇ Fixed access bug in `diguyScenario::find_path_shape()`.
- ◇ Added functions:

- `diguyCharacter::get_last_impact_is_from_vehicle()`
- `diguyCharacter::get_last_impact_speed_is_greater_than()`
- `diguyCharacterGuide::set_position_acquired()`
- `diguyCharacterGuide::set_orientation_acquired()`
- `diguyImpact::get_impact_speed()`

Version 8.1.0

- ◇ Added function `diguyScenario::preload_sound()`
- ◇ Added callback id
`diguyScenario::CALLBACK_ID_SAVE_SCENARIO_FILE`
- ◇ Added experimental functions:

- `diguyCharacter::get_nearest_waypoint_index_in_path()`
- `diguyCharacter::set_desired_orientation_to_current_orientation()`
- `diguyCharacter::check_character_visibility()`

Version 8.0.6

- ◇ Fixed issue where turreted vehicles broadcast angle did not match the DIS standard.
- ◇ Added experimental aim functions. These functions are for the new DI-Guy version 8 aim algorithm.

- `diguyCharacter::set_aim_algorithm()`
- `diguyCharacter::get_aim_algorithm()`
- `diguyCharacter::set_aim_param_seek_tolerance()`
- `diguyCharacter::get_aim_param_seek_tolerance()`
- `diguyCharacter::set_aim_param_time_constant()`
- `diguyCharacter::get_aim_param_time_constant()`
- `diguyCharacter::set_aim_param_time_constant2()`
- `diguyCharacter::get_aim_param_time_constant2()`
- `diguyCharacter::set_aim_param_reset_time_constant()`
- `diguyCharacter::get_aim_param_reset_time_constant()`
- `diguyCharacter::set_aim_param_azimuth_acquired_range()`
- `diguyCharacter::get_aim_param_azimuth_acquired_range()`
- `diguyCharacter::set_aim_param_elevation_acquired_range()`
- `diguyCharacter::get_aim_param_elevation_acquired_range()`
- `diguyCharacter::set_aim_param_max_per_frame_adj_factor()`
- `diguyCharacter::get_aim_param_max_per_frame_adj_factor()`
- `diguyCharacter::set_aim_converge_every_frame()`
- `diguyCharacter::get_aim_converge_every_frame()`
- `diguyCharacter::aim_converge()`

Version 8.0.5

- ◇ Fixed issue with TXP geometry loader. If external textures are marked as "Unknown" format, the actual format is now derived from the filename extension in the case of .rgb, .rgba, .int, and .inta texture files.

Version 8.0.4

- ◇ Improved documentation for function `diguyScenario::update()`.
- ◇ Fixed some issues with I-Guy and its use of joystick input. I-Guy input for DI-Guy 8 now more closely matches that of previous versions.
- ◇ Fixed some issues with muzzle flashes. In particular, muzzle flashes were not consistently showing up in DI-Guy Scenario, both during initial simulation and during review data playback.
- ◇ DI-Guy now includes a DIGuyLicenseTool application that makes it easier to gather the appropriate information needed to request a license from *Boston Dynamics*. This utility program can be run during Applications installation, or, after installation is complete, from the DI-Guy folder in the Start Menu.
- ◇ Fixed a problem with programming examples that used OpenAL with the VC7 static libs (Makefile.vc7_md, for example) would crash when an .ogg file was loaded.

Version 8.0.3

- ◇ Fixed some issues with I-Guy:
 - Firing the I-Guy's weapon now correctly uses the trajectory to the crosshair, instead of along the weapon barrel.
 - Keyboard control of looking up and down now possible; use 'Page Up' and 'Page Down' keys.
- ◇ Fixed problem with networked characters climbing hills.
- ◇ Fixed problem with networked characters transitioning too quickly.
- ◇ Added Visual Studio 7.1 solution for diguy_ogl/simple_playback example.
- ◇ Fixed issue with `diguyCharacter::stop_all_sounds()` and `diguyScenario::stop_all_sounds()` not saving history events.

Version 8.0.2

- ◇ Fixed problem with some textures not loading properly.
- ◇ Fixed problem with DirectX examples not shutting down cleanly if there is a license problem.

Version 8.0.1

- ◇ Fixed issues with temporary characters not always being destroyed.
- ◇ Fixed issues with temporary characters showing up in review playback.
- ◇ Fixed issues with appearance effects being recycled properly.

DI-Guy SDK Version 12.5.2 Index

- ◇ Fixed problem with eyes not participating in gazing.
- ◇ Rework of many examples in the `programming_examples` directory. Some new examples added, some older ones removed.
- ◇ Added README.txt files to `diguy_plugin` examples that better describe what the examples show, and how to use them in DI-Guy Scenario.

Version 8.0.0

- ◇ Initial 8.0 release candidate.
- ◇ Setting of tin (T In) and tout (T Out) calls for characters have been tightened up. For example, a character's tin can no longer be set to before the current scenario t. It could be before, but would cause potential problems. Warning messages are printed to the log when improper tin or tout times are attempted.

Version 7.1.20

- ◇ Added new `diguySignal` and `diguyScenario` functions:

```
· diguySignal::set_info_text()  
· diguySignal::get_info_text()  
· diguyScenario::signal_palette_show_all()  
· diguyScenario::signal_palette_hide_all()
```

- ◇ The signal `info_text` field allows signals to be tagged in the DI-Guy Scenario Signal Palette with text different than the signal name. The new show all and hide all functions take an optional filter that can be used to make the operation work on a subset of the signals.

Version 7.1.19

- ◇ Fixed problem with `vehicle_wheels` character in which first frame of playback had wheels in incorrect position.
- ◇ Improved documentation for `diguyCharacter` Performance Functions.
- ◇ Improved documentation for `diguyCharacter` functions that deal with enabling and disabling characters in various ways.
- ◇ Added functions:

```
· diguyLoadManager::set_zone_all_blends_disabled()  
· diguyLoadManager::get_zone_all_blends_disabled()  
· diguyLoadManager::set_zone_shape_switches_disabled()  
· diguyLoadManager::get_zone_shape_switches_disabled()  
· diguyLoadManager::set_zone_shape_callbacks_disabled()  
· diguyLoadManager::get_zone_shape_callbacks_disabled()  
· diguyLoadManager::set_zone_graphics_api_shape_update_disab  
· diguyLoadManager::get_zone_graphics_api_shape_update_disab  
· diguyCharacter::get_is_active()  
· diguyCharacter::set_invisible_flag()  
· diguyCharacter::get_invisible_flag()
```

- ◇ Added function `diguyCharacterGroup::get_is_network_paused()`, and deprecated function `diguyCharacterGroup::get_paused()`

DI-Guy SDK Version 12.5.2 Index

- ◊ Added function `diguyCharacterGroup::get_visible()`, and deprecated function `diguyCharacterGroup::get_is_drawn()`
- ◊ Made the following functions callable from decisions:

- `diguyCharacter::get_is_network_paused()`
- `diguyCharacter::get_is_network_published()`
- `diguyCharacter::get_is_network_reflected()`

Version 7.1.18

- ◊ Added functions:

- `diguy_extract_translation_and_euler_angles_from_matrix()`
- `diguy_create_matrix_from_translation_and_euler_angles()`
- `diguyCharacter::set_appearance()`
- `diguyCharacter::get_aim_is_possible()`

- ◊ Changed the name of several experimental functions:

- `diguyCharacter::start_appearance_effect()` ->
 `diguyCharacter::begin_appearance_effect()`
- `diguyCharacter::stop_appearance_effect()`
 ->
 `diguyCharacter::end_appearance_effect()`

- ◊ The following experimental functions were promoted to official API status:

- `diguyScenario::create_temporary_character()`
- `diguyScenario::send_character_to_recycle_bin()`
- `diguyScenario::retrieve_character_from_recycle_bin()`
- `diguyScenario::set_playback_mode()`
- `diguyScenario::set_playback_mode_stop()`
- `diguyScenario::set_playback_mode_play()`
- `diguyCharacter::get_is_temporary()`
- `diguyCharacter::translate_all_paths()`
- `diguyCharacter::rotate_all_paths_about_point()`
- `diguyCharacter::rotate_all_paths_about_current_waypoint()`
- `diguyCharacter::manually_invoke_event_handler()`

- ◊ The ability to change the tin ("T In") time of a scenario has been removed. THIS CHANGE WILL AFFECT SCENARIOS THAT USED THIS FUNCTIONALITY. Setting the tin of a scenario to anything other than 0 has always had unintended side-effects for DI-Guy, which is why this capability has been removed. The functions `diguyScenario::set_tin()` and `diguyScenario::get_tin()` are now deprecated as a result. (Note that `diguyCharacter::set_tin()` has not been changed.)

Version 7.1.16

- ◊ Added the following functions:

- `diguyCharacter::set_initial_up_vector()`
- `diguyCharacter::get_initial_up_vector()`
- `diguyCharacter::get_up_vector()`

Version 7.1.14

◊ Fixed issues with scenario merging. Chain settings and particle systems in particular did not merge well.

Version 7.1.13

◊ The following experimental functions were promoted to official API status:

- `diguyCharacter::set_minimum_cpl()`
- `diguyCharacter::get_minimum_cpl()`
- `diguyCharacter::get_maximum_possible_current_cpl()`
- `diguyCharacter::maximize_current_cpl()`
- `diguyCharacter::set_automatic_cpl_switching_enabled()`
- `diguyCharacter::get_automatic_cpl_switching_enabled()`
- `diguyCharacter::set_current_cpl()`
- `diguyCharacter::get_current_cpl()`
- `diguyCharacter::set_position_accumulation_disabled()`
- `diguyCharacter::get_position_accumulation_disabled()`
- `diguyCharacter::set_all_blends_disabled()`
- `diguyCharacter::get_all_blends_disabled()`
- `diguyCharacter::set_shape_switches_disabled()`
- `diguyCharacter::get_shape_switches_disabled()`
- `diguyCharacter::set_shape_callbacks_disabled()`
- `diguyCharacter::get_shape_callbacks_disabled()`
- `diguyCharacter::set_graphics_api_shape_update_disabled()`
- `diguyCharacter::get_graphics_api_shape_update_disabled()`
- `diguyCharacter::optimized_update()`
- `diguyCharacter::get_is_recycled()`
- `diguyCharacter::get_last_impact_record()`
- `diguyCharacter::get_last_impact_number()`
- `diguyCharacter::get_last_impact_been_processed()`
- `diguyCharacter::set_last_impact_been_processed()`
- `diguyCharacter::get_last_impact_link()`
- `diguyCharacter::get_last_impact_link_is()`
- `diguyCharacter::set_weapon_projectile_is_visible()`
- `diguyCharacter::get_weapon_projectile_is_visible()`
- `diguyCharacter::start_appearance_effect()`
- `diguyCharacter::stop_appearance_effect()`
- `diguyApp::set_precompute_quaternions_enabled()`
- `diguyApp::get_precompute_quaternions_enabled()`
- `diguyApp::set_num_rendering_passes()`
- `diguyApp::get_num_rendering_passes()`

◊ The following deprecated functions were removed:

- `diguyApp::set_use_varsets_enabled()`
- `diguyApp::get_use_varsets_enabled()`

· diguyCharacterGroup::set_all_members_t()

Version 7.1.12

◇ Added new functions to diguyCharacter:

- diguyCharacter::set_position_to_other_character_position()
- diguyCharacter::set_desired_position_to_current_position()
- diguyCharacter::set_desired_position_to_waypoint()
- diguyCharacter::set_desired_position_to_path_waypoint()
- diguyCharacter::set_desired_position_to_path_shape_waypoint()
- diguyCharacter::find_guide()

Version 7.1.11

◇ The following functions were added to [diguySensorRegion](#) that make it easier to write callback handlers for sensor regions:

- get_autosense_callback_candidate_entered()
- get_autosense_callback_candidate_left()

◇ Characters now fire live rounds by default when [diguyCharacter::fire_weapon\(\)](#) is called. THIS IS A CHANGE FROM PREVIOUS DEFAULT BEHAVIOR. The old behavior can be restored in two ways:

- In DI-Guy Scenario, the Weapon Params tab on the Character Elements palette has a checkbox called "Weapon Fires Live Rounds". The checkbox can be unchecked, followed by pressing the "Copy to All Characters" button.
- In a program using the DI-Guy API, the function [diguyCharacter::set_weapon_fires_live_rounds\(\)](#) can be called.

◇ Character Labels Visibility is now set to 'None' in the default .dss file.

◇ Added function to diguyCharacterGuide:

- diguyCharacterGuide::remove_all_disabled_guides()

◇ Made the following functions available to Decision Beads and Decision Events:

- [diguyCharacter::remove_all_guides\(\)](#)
- [diguyCharacter::remove_all_disabled_guides\(\)](#)

◇ Removed DI-Guy Graphics API programming example [ogl_examples_high_precision/simple_playback_high_precision](#). Replaced it with streamlined example [ogl_examples/simple_playback_far_positions](#).

◇ Fixed problem with props/scene objects showing up at origin in after-action review.

Version 7.1.10

- ◇ Fixed a problem with the DI-Guy Graphics API that prevented textures on some characters from being correctly bound under certain circumstances.

Version 7.1.9

- ◇ The following optional functions were added to plugins:
 - void key_press()
 - void key_release()
- ◇ Added `diguyIGuyController` class for C++ and script control over the I-Guy character.
- ◇ Setting the dead reckoning time threshold is now done by setting the variable `time_threshold` in the `network.cfg` file. The variable `tick_period` has been deprecated and will no longer have any effect. The default value is 5 seconds.

Version 7.1.8

- ◇ Added callback and event handling support for scenario variables:
 - `diguyScenario::add_default_variable_callback()`
 - `diguyScenario::remove_default_variable_callback()`
 - `diguyScenario::remove_default_variable_callback_with_user_data()`
 - `diguyScenario::add_default_variable_callback_script()`
 - `diguyScenario::remove_default_variable_callback_script()`
 - `diguyScenario::register_variable_event_handler()`
 - `diguyScenario::register_variable_event_handler_from_library()`
 - `diguyScenario::register_variable_event_handler_script()`
 - `diguyScenario::has_registered_variable_event_handler()`
 - `diguyScenario::unregister_variable_event_handler()`
 - `diguyVariable::CALLBACK_ID_VALUE_CHANGED`
 - `diguyVariable::add_callback()`
 - `diguyVariable::remove_callback()`
 - `diguyVariable::remove_callback_with_user_data()`
 - `diguyVariable::add_callback_script()`
 - `diguyVariable::remove_callback_script()`
 - `diguyVariable::map_event_handler_to_callback_id()`
 - `diguyVariable::unmap_event_handler_from_callback_id()`

Version 7.1.7

- ◇ Added parameters `polygon_offset_factor` and `polygon_offset_units` to `bdiGraphicsInitOgl` structure to allow for better control over subface rendering in OpenGL.
- ◇ Addressed some issues with `diguyGraphicsLink` and far positions.

Version 7.1.6

- ◇ Added functions to `diguySensorRegion`:
 - `diguySensorRegion::contains_character_from_group()`

◇ Added experimental functions

`diguyScenario::create_temporary_character()` and `diguyCharacter::get_is_temporary()`. These functions allow for the creation of 'temporary' characters that are not saved in a .dss file and are recycled on scenario reset. Temporary characters are useful as fired munitions, effects, etc.

◇ Added functions to `diguyCharacterGuide`:

- `diguyCharacterGuide::get_position_acquired()`
- `diguyCharacterGuide::get_orientation_acquired()`

◇ Added new callbacks to `diguyCharacter`:

- `diguyCharacter::CALLBACK_ID_GUIDE_POSITION_ACQUIRED`
- `diguyCharacter::CALLBACK_ID_GUIDE_POSITION_UNACQUIRED`
- `diguyCharacter::CALLBACK_ID_GUIDE_ORIENTATION_ACQUIRED`
- `diguyCharacter::CALLBACK_ID_GUIDE_ORIENTATION_UNACQUIRED`
- `diguyCharacter::CALLBACK_ID_CURRENT_TOUT_REACHED`

◇ Added function

`diguyCharacter::set_custom_orientation_rx_and_ry()`.

This function allows the roll and pitch components of a character to be set without taking the character off of its current path, if there is one. The character's up vector (as set by `diguyCharacter::set_up_vector()`) must be 'c' (custom) for these values to have an effect.

◇ Added new distance calculation functions:

- `diguyCharacter::get_distance_to_character()`
- `diguyCharacter::get_distance_xy_to_character()`
- `diguyCharacter::get_distance_to_path()`

◇ Added new experimental functions for locating a nearby character.

These functions have a `check_visibility` argument; if 1, the function will do a simple ray trace against scene object geometry.

- `diguyCharacter::get_nearest_active_character()`
- `diguyCharacter::get_nearest_active_character_in_group()`

◇ Added

`diguySensorRegion::contains_character_by_type()`

◇ Added `diguyCharacter::force_local_path()` new

function that can grab a preexisting path. copy and transform it into the local space of the character and switch to it.

◇ Removed function `diguy_dis_get_action_from_entity_state()` from `diguy_module_dis.h`.

`diguyCharacter::get_action_from_entity_state()` should be called instead to map DIS values to an action.

Version 7.1.5

◇ Added experimental transformation functions for paths:

- `diguyCharacterPath::translate()`
- `diguyCharacterPath::rotate_about_point()`
- `diguyPathShape::translate()`
- `diguyPathShape::rotate_about_point()`

◇ Added experimental performance tuning functions:

- `diguyScenario::send_character_to_recycle_bin()`
- `diguyScenario::retrieve_character_from_recycle_bin()`

◇ Deprecated experimental performance tuning functions:

- `diguyApp::set_use_varsets_enabled()`
- `diguyApp::get_use_varsets_enabled()`

Version 7.1.4

◇ Added function

`diguyCharacterPath::get_path_shape()`, that provides access to a path's underlying path shape.

◇ Added functions

`diguyCharacter::get_nearest_active_character()`
and

`diguyCharacter::get_nearest_active_character_in_group()`.

These functions will walk through either all characters or all characters in a group, and return the nearest character that is both alive and enabled. An optional argument specifies whether to skip characters that are invisible to the caller because a scene object is in the way.

◇ Added convenience functions `diguyCharacter::get_distance()` and `diguyCharacter::get_distance_xy()` for getting the distance between two characters.

Version 7.1.3

◇ Deprecated `diguyChainSimulation::set_connection_character()` function. Replaced with

`diguyChainSimulation::set_chain_mass_connection_character()`.

◇ Added ability to set all chain mass positions in the call

`diguyChainSimulation::set_chain_mass_position()`
by passing `-1` for the `chain_mass_index`.

◇ Added ability to fix all chain mass positions in the call

`diguyChainSimulation::set_chain_mass_position_is_fixed()`
by passing `-1` for the `chain_mass_index`.

◇ Added aiming ability for characters with turrets.

◇ Fixed problems with debug versions of programming examples that caused heap corruption on some Windows systems.

Version 7.1.2

◇ Added new callback to `diguyCharacter`:

- `diguyCharacter::CALLBACK_ID_DESIRED_ACTION_REACHED`

Version 7.1.1

- ◇ Added a new experimental character type called "chain_simulation" that can do realtime simulation of chains, ropes, and hoses. The appearance of the chain_simulation character determines its simulation settings. Chain simulation settings are created and on the DI-Guy Scenario Chain Sim Settings page, or via the `diguyChainSettings` class. Runtime parameters of the chain simulation are set by obtaining a `diguyChainSimulation` object via the `diguyCharacter::get_chain_simulation()` call.
- ◇ Fixed issues with `diguyCharacter::set_current_appearance()`. 0 was returned even if appearance change failed; not correctly returns -1 on failure.
- ◇ Added new experimental functions to event bead classes. These functions make it possible to specify that event beads should begin and/or when a specific action bead or waypoint is reached. Members are:

```

· diguyCharacterPathAimBead::get_begin_at_object_name()
· diguyCharacterPathAimBead::set_begin_at_object_name()
· diguyCharacterPathAimBead::get_end_at_object_name()
· diguyCharacterPathAimBead::set_end_at_object_name()
· diguyCharacterPathDecisionBead::get_begin_at_object_name()
· diguyCharacterPathDecisionBead::set_begin_at_object_name()
· diguyCharacterPathDecisionBead::get_end_at_object_name()
· diguyCharacterPathDecisionBead::set_end_at_object_name()
· diguyCharacterPathGazeBead::get_begin_at_object_name()
· diguyCharacterPathGazeBead::set_begin_at_object_name()
· diguyCharacterPathGazeBead::get_end_at_object_name()
· diguyCharacterPathGazeBead::set_end_at_object_name()
· diguyCharacterPathScriptBead::get_begin_at_object_name()
· diguyCharacterPathScriptBead::set_begin_at_object_name()
· diguyCharacterPathScriptBead::get_end_at_object_name()
· diguyCharacterPathScriptBead::set_end_at_object_name()

```

- ◇ Function `diguyApp::load_plugin()` has three new optional arguments: `required_by_scenario`, `unload_on_scenario_close`, and `accepts_mouse_input`.
- ◇ Added functions to `diguyImpact`:

```

· diguyImpact::get_location_in_link_coordinates()

```

- ◇ Added functions to `diguyScenario`:

```

· diguyScenario::get_impact_at_screen_coords()

```

- ◇ Added functions to `diguyCharacter`:

```

· diguyCharacter::set_label_text_shadow_color()
· diguyCharacter::get_label_text_shadow_color()

```

- `diguyCharacter::set_label_text_shadow_is_visible()`
- `diguyCharacter::get_label_text_shadow_is_visible()`

◇ Character label information is now saved in the scenario file and restored when the scenario is loaded.

Version 7.1.0

◇ Added function `diguy_get_app()`, a cleaner function for retrieving the `diguyApp` object than `diguyApp::get_app()`.

◇ Added experimental transformation functions for paths:

- `diguyCharacter::translate_all_paths()`
- `diguyCharacter::rotate_all_paths_about_point()`
- `diguyCharacter::rotate_all_paths_about_current_waypoint()`
- `diguyCharacterGroup::translate_all_paths()`
- `diguyCharacterGroup::rotate_all_paths_about_point()`
- `diguyCharacterGroup::rotate_all_paths_about_current_waypoint()`

◇ Added functions `diguyApp::get_diguy_base_dir()` and `diguyApp::get_diguy_subdir()`.

◇ Added `diguyCharacter::get_parent()` function.

◇ Added `diguyViewLight::set_shadowmap_size()` and `diguyViewLight::get_shadowmap_size()`; allows setting shadow map resolution. Must be a power of 2.

◇ Made various internal performance enhancements for scenario updates.

◇ Fixed a problem with motion LODs updating all links every now and then, even if the current action hasn't changed.

◇ Made the scenario be able to go past 60000 seconds. Now a scenario tout greater than 60000 seconds is interpreted as unlimited.

◇ Fixed a problem with motion LODs updating too many links.

◇ Made various internal performance enhancements for scenario updates.

◇ Added performance tuning functions:

- `diguyCharacter::set_automatic_cpl_switching_enabled()`
- `diguyCharacter::get_automatic_cpl_switching_enabled()`
- `diguyCharacter::set_minimum_cpl()`
- `diguyCharacter::get_minimum_cpl()`
- `diguyCharacter::get_maximum_possible_current_cpl()`
- `diguyCharacter::maximize_current_cpl()`
- `diguyCharacter::get_current_cpl()`
- `diguyCharacter::set_position_accumulation_disabled()`
- `diguyCharacter::get_position_accumulation_disabled()`
- `diguyCharacter::set_all_blends_disabled()`
- `diguyCharacter::get_all_blends_disabled()`
- `diguyCharacter::set_shape_switches_disabled()`
- `diguyCharacter::get_shape_switches_disabled()`
- `diguyCharacter::set_shape_callbacks_disabled()`
- `diguyCharacter::get_shape_callbacks_disabled()`
- `diguyCharacter::set_graphics_api_shape_update_disabled()`

- `diguyCharacter::get_graphics_api_shape_update_disabled()`
- `diguyCharacter::optimized_update()`
- `diguyScenario::optimized_update()`
- `diguyScenario::send_character_to_recycle_bin()`
- `diguyScenario::retrieve_character_from_recycle_bin()`
- `diguyGraphicsLink::get_data_ptr()`
- `diguyGraphicsLink::get_translation_data_index()`
- `diguyGraphicsLink::get_rotation_data_index()`
- `diguyGraphicsLink::get_scale_data_index()`
- `diguyGraphicsLink::get_quat_data_ptr()`
- `diguyGraphicsLink::get_quat_data_index()`
- `diguyApp::set_use_varsets_enabled()`
- `diguyApp::get_use_varsets_enabled()`
- `diguyApp::set_precompute_quaternions_enabled()`
- `diguyApp::get_precompute_quaternions_enabled()`

Version 7.0.9

- ◇ Rewrote portions of network mapping code, custom pdus/specialized motions will now work with more characters. This requires a new version of data, but any character that has been properly tagged with variants and postures should transmit better.
- ◇ Rewrote speed code, networked characters should scale speed to try and match packets that are coming in.

Version 7.0.8

- ◇ Trying to get the position of a invalid link will behave a bit better.

Version 7.0.6

- ◇ Creating a path when a character has none will properly set the initial path name.

Version 7.0.5

- ◇ Fixed a crash that sometimes happened while reading comments out of texture .attr files.

Version 7.0.3

- ◇ Fixed a problem with the `diguyGraphicsShape::update()` function not always being called.
- ◇ Fixed various bugs with VR-Link and DI-Guy time sync code.
- ◇ Added warning if `diguyCharacter::set_desired_action()` is called with 0.0f as the speed.

Version 7.0.2

- ◇ Fixed a problem with gazing and aiming not zeroing in on the correct positions when using DI-Guy for Vega.

Version 7.0.1

- ◇ Deprecated "named_callback" functions in `diguyApp`. Event handler functions should be used instead.
- ◇ Renamed some `diguyViewCamera` functions:

DI-Guy SDK Version 12.5.2 Index

- diguyViewCamera::get_world_offset_position()
has become
`diguyViewCamera::get_segment_origin()`
- diguyViewCamera::get_local_position() has
become
`diguyViewCamera::get_position_segment_offset()`
- diguyViewCamera::get_local_fix() has become
`diguyViewCamera::get_fix_segment_offset()`

◇ Added function

`diguyCharacterGroup::update_all_members()`, and
deprecated function `diguyCharacterGroup::set_all_members_t()`

◇ Fixed problem with deleting one scenario and creating another
causing a crash.

◇ Fixed crash problem with function `diguy_set_cfg_file()`.

◇ Changed the default settings of automatic switching of graphics
LODs and motion LODs, as set by

`diguyScenario::set_automatic_graphics_lod_switching()`

and

`diguyScenario::set_automatic_motion_lod_switching()`.

Previously automatic LOD switching was always enabled unless
explicitly disabled by calling the above functions. Now automatic
LOD switching will be disabled by default in DI-Guy API
applications, but remain enabled by default in DI-Guy Scenario.

◇ Added the following functions for adding and removing appearances:

- `diguyScenario::add_appearance()`
- `diguyScenario::add_appearance_based_on_existing_appearance()`
- `diguyScenario::remove_appearance()`

◇ `diguyScenario::add_custom_appearance()` deprecated; use
`diguyScenario::add_appearance()`.

◇ `diguyScenario::add_custom_appearance_based_on_existing_appearance()`
deprecated; use

`diguyScenario::add_appearance_based_on_existing_appearance()`.

◇ Promoted the following functions from experimental to official
status:

- `diguyCharacter::get_label_text()`
- `diguyCharacter::set_label_text()`
- `diguyCharacter::get_label_is_visible()`
- `diguyCharacter::set_label_is_visible()`
- `diguyCharacter::set_label_text_color()`
- `diguyCharacter::get_label_text_color()`
- `diguyCharacter::set_label_background_color()`
- `diguyCharacter::get_label_background_color()`

◇ Removed `diguyLoadManager` camera functions (load manager
camera now always tracks the scenario's primary view camera):

- `diguyLoadManager::add_camera()`

DI-Guy SDK Version 12.5.2 Index

- `diguyLoadManager::add_view_camera( )`
- `diguyLoadManager::move_camera( )`
- `diguyLoadManager::change_view_camera_settings( )`
- `diguyLoadManager::remove_camera( )`
- `diguyLoadManager::remove_all_cameras( )`

Version 7.0.0

- ◇ Fixed problem with `diguyCharacter::set_speed( )`. The set speed was being reset to the action's default speed after a short time, which is not correct behavior.
- ◇ Changed values returned by `diguyCharacter::get_aim_azimuth( )` and `diguyCharacter::get_aim_elevation( )` to be character-relative rather than world-relative.

Version 6.1.10

- ◇ Fixed event propagation bug when event tout is -1 and scenario tout is changed.
- ◇ Added optional actor argument to `diguyScenario::add_custom_appearance( )` function. This fixes a problem with characters not being able to switch to custom appearances with `diguyCharacter::set_current_appearance( )` because of actors being different.
- ◇ Added `simple_playback_high_precision` and `view_far_position` that demonstrate off the use of high precision position values.

Version 6.1.9

- ◇ Fixed problem with various `diguyWaypoint` "get" functions that return angles (e.g., `diguyWaypoint::get_yaw_in( )`); they weren't properly returning angles in degrees.
- ◇ Fixed problem with merging scenarios: merged waypoints at the origin were not being rotated correctly.
- ◇ Fixed problem with config file appearance syntax. The item "shapaset:shape" format caused a crash; it now works again.
- ◇ Fixed problems with required plug-in list not saving or resetting properly.

Version 6.1.8

- ◇ Far positions rendering are now off by default, though DI-Guy Scenario and DI-Guy Scenario UI V6 turn them on. Practically this means that while internally things are done at a higher precision format, the render camera being in local space will be a change that needs to be explicitly asked for.
- ◇ Added the following functions for controlling far position state:
 - `diguy_ogl_get_use_far_positions( )`
 - `diguy_ogl_set_use_far_positions( )`
 - `diguy_graphics_get_use_far_positions( )`
 - `diguy_graphics_set_use_far_positions( )`

◇ Added accessors for setting character and scene object culling on/off.

- `diguyScenario::set_character_culling_enabled()`
- `diguyScenario::get_character_culling_enabled()`
- `diguyScenario::set_scene_object_culling_enabled()`
- `diguyScenario::get_scene_object_culling_enabled()`

◇ Added accessors for setting character and scene object bounding volume visualization on/off. (only works for OpenGL)

- `diguyScenario::set_character_visualize_bounds_enabled()`
- `diguyScenario::get_character_visualize_bounds_enabled()`
- `diguyScenario::set_scene_object_visualize_bounds_enabled()`
- `diguyScenario::get_scene_object_visualize_bounds_enabled()`

◇ Fixed a bug with

`diguyCharacter::set_current_appearance()`.

◇ Added warning message if a proposed appearance change is null; fixes crash bug.

◇ `diguyScenario::trigger_script_event()` deprecated; use `diguyScenario::trigger_script()`.

◇ Added function `diguyScenario::trigger_decision()`.

Version 6.1.7

◇ Added new experimental functions. (Experimental functions are typically for probationary features whose final functionality or API has not been fully worked out.)

- `diguyCharacter::set_weapon_projectile_is_visible()`
- `diguyCharacter::get_weapon_projectile_is_visible()`
- `diguyCharacter::get_label_text()`
- `diguyCharacter::set_label_text()`
- `diguyCharacter::get_label_is_visible()`
- `diguyCharacter::set_label_is_visible()`
- `diguyCharacter::set_label_text_color()`
- `diguyCharacter::get_label_text_color()`
- `diguyCharacter::set_label_background_color()`
- `diguyCharacter::get_label_background_color()`
- `diguyScenario::find_character_at_screen_coords()`
- `diguyScenario::set_playback_mode()`
- `diguyScenario::set_playback_mode_stop()`
- `diguyScenario::set_playback_mode_play()`

◇ Removed the function `diguyLoadManager::set_enable_culling()`.

◇ Added optional arguments `initial_posture` and `initial_variant` to `diguyScenario::set_iguy_character()`.

◇ Added function `diguyScenario::get_time_of_day()`.

◇ Fixed bug in `diguyCharacter::is_speaking()`. It was sometimes improperly returning FALSE.

◇ Added the following functions that control whether the scenario will be reset (as if `diguyScenario::reset()` were called) when a

merge happens:

- `diguyScenarioMergeSettings::set_reset_flag()`
- `diguyScenarioMergeSettings::get_reset_flag()`

- ◇ Now incoming XC_MERGE_SCENARIO PDUs do not cause the running scenario to stop and reset.
- ◇ Now incoming XC_MERGE_SCENARIO PDUs cause a XC_MERGE_SCENARIO_COMPLETE PDU to be sent when the merge operation is done.
- ◇ Now incoming XC_LOAD_SCENARIO PDUs cause a XC_LOAD_SCENARIO_COMPLETE PDU to be sent when the load operation is complete.
- ◇ Added the following functions:

- `diguyExerciseInterface::send_pdu_xc_eval_script()`
- `diguyExerciseInterface::send_pdu_xc_time_now()`

- ◇ Now incoming XC_TIME_NOW messages cause the scenario to chase sync.
- ◇ Now incoming XC_EVAL_SCRIPT messages cause the script therein to be evaluated.
- ◇ `diguyScenario::load()` has been changed to not accept .flt filenames. The new function `diguyScenario::create_scene_object()` should be used instead.
- ◇ Added the following functions:

- `diguyScenario::create_scene_object()`
- `diguyScenario::destroy_scene_object()`

- ◇ Fixed problem with DX99 version of the library that prevented gazing from working correctly.
- ◇ Added better detection of and error message for missing DI-Guy Data installation.
- ◇ Renamed experimental "shot" class, function names, and callback ids to use the more generic term "impact":

- `diguyShot`
 - `diguyShot` (class name) -> `diguyImpact`
 - `get_shooter_name()` -> `get_attacker_name()`
 - `get_shooter()` -> `get_attacker()`
 - `get_shot_number()` -> `get_impact_number()`
 - `set_shot_number()` -> `set_impact_number()`
 - `get_shot_been_processed()` -> `get_impact_been_processed()`

DI-Guy SDK Version 12.5.2 Index

- `set_shot_been_processed()` ->
`set_impact_been_processed()`

· `diguyCharacter`

- `CALLBACK_ID_GOT_SHOT` ->
`CALLBACK_ID_IMPACT`
- `CALLBACK_ID_SHOT_SUCCESS`
->
`CALLBACK_ID_FIRE_WEAPON_SUCCESS`
- `get_last_shot_record()` ->
`get_last_impact_record()`
- `get_last_shot_number()` ->
`get_last_impact_number()`
- `get_last_shot_been_processed()` ->
`get_last_impact_been_processed()`
- `set_last_shot_been_processed()` ->
`set_last_impact_been_processed()`
- `get_last_shot_link()` ->
`get_last_impact_link()`
- `get_last_shot_link_is()` ->
`get_last_impact_link_is()`
- `set_max_shot_hits()` ->
`set_max_impact_hits()`

· `diguyScenario`

- `CALLBACK_ID_ENVIRONMENTAL_DETONATION`
->
`CALLBACK_ID_SCENE_OBJECT_IMPACT`
- `get_last_env_shot_info()` ->
`get_last_env_impact_info()`

◇ Added function

`diguyExerciseInterface::send_pdu_xc_time_now()`.

◇ Added code to properly respond to incoming XC_TIME_NOW PDUs.

Version 6.1.6

◇ Added functions and `diguyExerciseInterface` class for enhanced networking support:

- `diguyApp::get_exercise_interface()`
- `diguyExerciseInterface::send_pdu_xc_load_scenario()`
- `diguyExerciseInterface::send_pdu_xc_merge_scenario()`
- `diguyExerciseInterface::send_pdu_xc_trigger_script()`
- `diguyExerciseInterface::send_pdu_xc_control()`
- `diguyScenario::set_tin_time_of_day()`
- `diguyScenario::get_tin_time_of_day()`

DI-Guy SDK Version 12.5.2 Index

- ◇ Added new functions `diguyView::get_width()` and `diguyView::get_height()`.
- ◇ Experimental plugin infrastructure has been reworked a fair amount.

- Function `diguyScenario::add_plugin()` has been changed to `diguyApp::load_plugin()`.
- Function `diguyScenario::remove_plugin()` has been changed to `diguyApp::unload_plugin()`.
- The following functions will be imported when a plugin is loaded. Note that not only `diguy_version()` must be implemented:

- `void diguy_version(int* version_major, int* version_minor, int* version_point)`
- `void initialize_plugin(diguyApp* app)`
- `void deinitialize_plugin(diguyApp* app)`
- `void initialize(diguyScenario* s)`
- `void deinitialize(diguyScenario* s)`
- `void pre_update(diguyScenario* s)`
- `void post_update(diguyScenario* s)`
- `void pre_draw(diguyScenario* s)`
- `void post_draw(diguyScenario* s)`
- `void reset(diguyScenario* s)`
- `diguyPluginMouseCursor`
`get_mouse_cursor(diguyScenario* s)`
- `void mouse_press(diguyScenario* s, diguyPluginMouseEvent* mouse_event)`
- `void mouse_drag(diguyScenario* s, diguyPluginMouseEvent* mouse_event)`
- `void mouse_release(diguyScenario* s, diguyPluginMouseEvent* mouse_event)`
- `void mouse_move(diguyScenario* s, diguyPluginMouseEvent* mouse_event)`
- `void save_review_data(diguyScenario* s, const char* filename)`
- `void load_review_data(diguyScenario* s, const char* filename)`

- ◇ Added an additional argument, `merge_settings`, to the `diguyScenario::merge()` function.

DI-Guy SDK Version 12.5.2 Index

- ◇ Added the following functions to enable better control of merging scenarios:

- `diguyScenario::create_merge_settings()`
 - `diguyScenario::destroy_merge_settings()`

- ◇ Added the following functions:

- `diguyCharacter::revive_now()`

- ◇ Added the following functions in support of 3D positional sound:

- `diguyCharacter::set_default_weapon_sound()`
 - `diguyCharacter::get_default_weapon_sound()`
 - `diguyCharacter::set_default_weapon_sound_gain()`
 - `diguyCharacter::get_default_weapon_sound_gain()`
 - `diguyCharacter::play_3d_sound()`
 - `diguySoundInstance::set_is_3d_sound_flag()`
 - `diguySoundInstance::get_is_3d_sound_flag()`

- ◇ Added new optional argument "gain" to `diguyCharacter::play_sound()` function.
- ◇ Made following functions available to Decision Beads and Decision Events:

- `diguyCharacter::aim_at_character()`
 - `diguyCharacter::end_aim()`
 - `diguyCharacter::end_gaze()`
 - `diguyCharacter::end_lpoint()`
 - `diguyCharacter::gaze_at_character()`
 - `diguyCharacter::lpoint_at_character()`
 - `diguyCharacter::resume_interrupted_path()`
 - `diguyCharacter::play_3d_sound()`

- ◇ Removed unused variable "working_directory" from `bdiExFaceInit` structure. Replaced with variable "placeholder", leaving the option open for future init variables. Calls passing NULL for the `bdiExFaceInit` structure to `diguy_exface_initialize()` will not need to change.
- ◇ Fixed problem with Decision Bead that incorrectly put function `variable_less_than_string()` in variable query functions instead of character query functions.
- ◇ Added `script_type` argument to `diguyScenario::eval_script()` and `diguyScenario::eval_script_file()` functions.

Version 6.1.5

- ◇ Fixed problem that newly created characters were not appearing until the next scenario tick dt boundary was crossed.
- ◇ `diguyCharacter::set_t()` deprecated; use `diguyCharacter::update()`.

DI-Guy SDK Version 12.5.2 Index

- ◊ Added new function `diguyCharacterPath::get_length()`.
- ◊ Fixed potential crash from call to function `diguyCharacterPath::update()`.
- ◊ Character positions now stored in higher precision format.

· Added new functions:

- `diguyCharacter::set_position_double()`
- `diguyCharacter::get_position_double()`
- `diguyCharacter::get_link_position_double()`
- `diguyCharacter::get_link_position_with_offset_double()`
- `diguyGraphicsLink::get_translation_double()`
- `diguyViewCamera::get_position_double()`

· New local space camera. DI-Guy can now render in the local space of the camera. This allows DI-Guy to avoid putting large values on the rendering stack, avoiding numerical imprecision on objects that are far from the origin.

· Added new local space camera functions:

- `diguyScenario::set_render_camera()`
- `diguyViewCamera::get_local_fix()`
- `diguyViewCamera::get_local_position()`

In OpenGL, use
`gluLookAt(local_position_x,
local_position_y, local_position_z,
local_fix_x, local_fix_y, local_fix_z, 0.0,
0.0, 1.0);` to emulate the DI-Guy camera.

Version 6.1.4

- ◊ Added new function `diguyGraphicsShape::get_is_hidden()`.
- ◊ Added new functions to `diguyCharacterGroup` class:

- `diguyCharacterGroup::get_num_members()`
- `diguyCharacterGroup::get_member_at_index()`

- ◊ Added new functions to `diguySensorRegion` class:

- `diguySensorRegion::add_characters_in_region_to_group()`
- `diguySensorRegion::add_candidate_characters_in_region_to_group()`

- ◊ Removed unused functions from `diguySensorRegion` class:

- `diguySensorRegion::get_characters_in_region()`
- `diguySensorRegion::update_characters_in_region()`
- `diguySensorRegion::update_candidate_characters_in_region()`

Version 6.1.3

DI-Guy SDK Version 12.5.2 Index

- ◇ Added new utility function `diguy_convert_euler_angle_order()`. This function should help DI-Guy API users convert the DI-Guy standard Euler angle order of ZXY (yaw then roll then pitch) into other Euler angle orders.
- ◇ Added new argument `rampup_interval` to function `diguyCharacter::add_motion_texture()`. This function makes it possible to more smoothly enter a motion texture. Previously the rampup was hard-coded to 0.2 seconds. If the function is to end existing motion textures, it also uses the passed rampup interval as the rampdown interval for ending the existing motion textures.
- ◇ Fixed annoying error message that was printed when motion textures ended.
- ◇ Added new variants that can be added to custom motions:

- `DIGUY_MOTION_VARIANT_MISC0`
- `DIGUY_MOTION_VARIANT_MISC1`
- `DIGUY_MOTION_VARIANT_MISC2`
- `DIGUY_MOTION_VARIANT_MISC3`
- `DIGUY_MOTION_VARIANT_MISC4`
- `DIGUY_MOTION_VARIANT_MISC5`
- `DIGUY_MOTION_VARIANT_MISC6`
- `DIGUY_MOTION_VARIANT_MISC7`
- `DIGUY_MOTION_VARIANT_MISC8`
- `DIGUY_MOTION_VARIANT_MISC9`

Version 6.1.2

- ◇ Changed win32 programming examples so that when examples are compiled for different versions of LIBC (i586_md_debug as opposed to i586_md, for example), the object files will be placed in separate directories.
- ◇ Added new function calls that will free memory allocated by the DI-Guy Graphics API loader once platform-specific objects have been created:
 - `diguyGraphicsMaterial::free_loader_memory()`
 - `diguyGraphicsMesh::free_loader_memory()`
 - `diguyGraphicsState::free_loader_memory()`
 - `diguyGraphicsTexture::free_loader_memory()`
- ◇ Fixed a problem with gazing in which a character with orientation change enabled would do an extra shuffle to the left and back again when the character was already facing the gaze angle at gaze start.
- ◇ Added new side-effects to characters that die, including those for which the `diguyCharacter::die_now()` function is called:
 - all aiming is ended
 - all gazing is ended
 - all pointing is ended
 - head nodding and shaking are stopped

- all gestures are aborted
- all sounds originated by the character are stopped

◊ Added new function call limitations to dead characters. The following function calls will have no effect on dead characters:

- `diguyCharacter::add_motion_texture()`
- `diguyCharacter::aim_at_angle()`
- `diguyCharacter::aim_at_angle_local()`
- `diguyCharacter::aim_at_character()`
- `diguyCharacter::aim_at_point()`
- `diguyCharacter::aim_at_point_local()`
- `diguyCharacter::create_and_force_bridge_path()`
- `diguyCharacter::execute_gesture()`
- `diguyCharacter::execute_1stage_gesture()`
- `diguyCharacter::execute_2stage_gesture()`
- `diguyCharacter::execute_3stage_gesture()`
- `diguyCharacter::fire_screen_space_bullet()`
- `diguyCharacter::fire_weapon()`
- `diguyCharacter::fire_weapon_n_times()`
- `diguyCharacter::force_action()`
- `diguyCharacter::force_action_and_path_shape()`
- `diguyCharacter::force_action_with_duration()`
- `diguyCharacter::force_partial_path()`
- `diguyCharacter::force_path()`
- `diguyCharacter::gaze_at_angle()`
- `diguyCharacter::gaze_at_angle_local()`
- `diguyCharacter::gaze_at_character()`
- `diguyCharacter::gaze_at_point()`
- `diguyCharacter::gaze_at_point_local()`
- `diguyCharacter::lpoint_at_angle()`
- `diguyCharacter::lpoint_at_angle_local()`
- `diguyCharacter::lpoint_at_character()`
- `diguyCharacter::lpoint_at_point()`
- `diguyCharacter::lpoint_at_point_local()`
- `diguyCharacter::nod_head()`
- `diguyCharacter::play_sound()`
- `diguyCharacter::push_path()`
- `diguyCharacter::resume_interrupted_path()`
- `diguyCharacter::set_aim()`
- `diguyCharacter::set_desired_action()`
- `diguyCharacter::set_nod()`
- `diguyCharacter::shake_head()`

◊ Added

`diguyCharacterPath::get_point_at_distance()`
function that makes it possible to sample path points. The function is very similar to

`diguyPathShape::get_point_at_distance()`.

◊ Changed `diguyScenario::create_waypoint()` function to take an additional argument. Waypoints created with the function can

now have different initial weights for entering and leaving the waypoint.

Version 6.1.1

◊ Added new member function:

- `diguyLoadManager::remove_all_cameras()`

◊ Added new member functions to `diguyVariable` that allow for setting and getting the initial value of the variable, and for retrieving the scenario and/or character the variable belongs to:

- `diguyVariable::get_scenario()`
- `diguyVariable::get_character()`
- `diguyVariable::get_initial_value_as_int()`
- `diguyVariable::get_initial_value_as_float()`
- `diguyVariable::get_initial_value_as_string()`
- `diguyVariable::set_initial_value_as_int()`
- `diguyVariable::set_initial_value_as_float()`
- `diguyVariable::set_initial_value_as_string()`

◊ Added character variables to characters. Character variables are almost exactly like scenario variables but are local to each character instead of being shared across the scenario. For example, all characters in a scenario could be given a variable named "panic_level" that tracks how panicked the character is. Character variables use the same class as scenario variables: `diguyVariable`. The following functions are available for accessing character variables:

- `diguyCharacter::get_num_variables()`
- `diguyCharacter::get_variable_at_index()`
- `diguyCharacter::find_variable()`
- `diguyCharacter::find_or_create_variable()`
- `diguyCharacter::destroy_variable()`
- `diguyCharacter::variable_equal_to_int()`
- `diguyCharacter::variable_greater_than_int()`
- `diguyCharacter::variable_less_than_int()`
- `diguyCharacter::variable_equal_to_float()`
- `diguyCharacter::variable_greater_than_float()`
- `diguyCharacter::variable_less_than_float()`
- `diguyCharacter::variable_equal_to_string()`
- `diguyCharacter::variable_greater_than_string()`
- `diguyCharacter::variable_less_than_string()`

Version 6.1.0

◊ Added new functions to the DI-Guy Graphics API to make it easier to know which character links or shapes belong to, and to easily get a pointer to the character's base link:

- `diguyGraphicsLink::get_character()`

- `diguyGraphicsShape::get_character()`
- `diguyCharacter::get_position_link()`

◇ Added new function

`diguyGraphicsLink::get_transformation_matrix()`.

This function provides access to the link transformation in matrix form rather than separate vectors for translation, rotation, etc.

◇ Added new blend functions "StepHalfway" and "Waver". The "StepHalfway" blend function makes an instant step from 0 to 1 halfway through the blend. The "Waver" blend function wavers from 0 to 1 back to 0 and back to 1 using a three-quarter sine wave.

◇ Added functions to `diguyScenario` API:

- `diguyScenario::get_character_type_num_gestures()`
- `diguyScenario::get_character_type_gesture_at_index()`

◇ Added functions to `diguyCharacterGesture` API:

- `diguyCharacterGesture::force_table_action()`

Version 6.0.8

◇ Changed DI-Guy Graphics API to not print a warning if a geometry file is not found. This allows external loaders to load their own version of a geometry file without warnings, and without the file having to be present in \$DIGUY/geometry.

◇ Added function `diguyCharacter::is_group_member()`.

◇ For Windows, added the DI-Guy version number as a suffix to the .dll that gets installed in the bin directory.

Version 6.0.6

◇ Changed default for gaze parameter "orientation enabled for gaze". In DI-Guy 5.x this parameter defaulted to 0 for all characters except the sergeant. In earlier versions of 6.0 this was changed to "1 if character has turning actions, else 0". This change has, however, resulted in too much confusion and user astonishment. The new (and hopefully final) change is to default the parameter to 0 for **all** characters. This fixes a backwards incompatibility problem where some version 5 characters that were gazing would leave their path on version 6 and thus not complete the actions specified by their path.

◇ Implemented function `diguyCharacter::end_aim()`. This function had been omitted from the DI-Guy API.

◇ Fixed problem with initial value of scenario variables, where the integer value of each variable was initialized to zero, even if the intended initial value was nonzero.

◇ Fixed problem when merging two scenarios with different values for tick dt. Decision beads, Script event beads, Aim beads and Gaze beads were being improperly scheduled.

Version 6.0.5

◇ Fixed `diguyCharacter::get_aim_azimuth()` and `diguyCharacter::get_aim_elevation()`. They were

returning zero. Now they return meaningful values if the character is aiming.

Version 6.0.4

- ◇ DI-Guy can now use OpenGL's multitexturing mechanism. The OpenGL texture unit that DI-Guy will use can be set by setting the `bdiGraphicsInitOgl::use_multitexture_extension` and `bdiGraphicsInitOgl::texture_unit` initialization variables.

Version 6.0.3

- ◇ Fixed a problem with DI-Guy Graphics API functions not being correctly declared for DLL use.
- ◇ Fixed a problem with calling path edit functions on a path that is in use. The change was not being fully propagated to the character's motions.
- ◇ Eliminated GoThere camera instability at low frame rates.
- ◇ Eliminated aiming instability at low frame rates and at low desired locate times.

Version 6.0.2

- ◇ No changes.

Version 6.0.1

- ◇ Fixed problem with motion scaling where the scale factor was being applied twice in some cases. (PR 3029)
- ◇ Removed deprecated `diguyCharacter` functions:
 - `diguyCharacter::force_action_by_name( )`
 - `diguyCharacter::get_desired_action_name( )`
 - `diguyCharacter::set_desired_action_by_name( )`
 - `diguyCharacter::force_partial_path_by_name( )`
 - `diguyCharacter::force_path_by_name( )`
 - `diguyCharacter::push_path_by_name( )`
 - `diguyCharacter::set_parent_by_name( )`
- ◇ Removed deprecated `diguySensorRegion` function `diguySensorRegion::contains_character_by_name( )`.
- ◇ Removed deprecated `diguySignal` function `diguySignal::play_sound( )`.
- ◇ Removed deprecated `diguyScenario` function `diguyScenario::find_or_create_face_expression( )`.

Version 6.0.0

- ◇ Made aiming precise. Characters now actually aim at the targets specified by aim beads, rather than giving an approximation of proper aim.
- ◇ Made aiming dynamic. Character aiming motion is now much more realistic, homing in on the specified target.
- ◇ Added new `diguyCharacter` functions:

DI-Guy SDK Version 12.5.2 Index

- `diguyCharacter::aim_at_angle()`
- `diguyCharacter::aim_at_angle_local()`
- `diguyCharacter::aim_at_point()`
- `diguyCharacter::aim_at_point_local()`
- `diguyCharacter::aim_at_character()`
- `diguyCharacter::set_aim_param_desired_locate_time()`
- `diguyCharacter::get_aim_param_desired_locate_time()`
- `diguyCharacter::get_aim_is_acquired()`
- `diguyCharacter::get_aim_is_steady()`
- `diguyCharacter::set_weapon_near_range()`
- `diguyCharacter::set_weapon_far_range()`

◇ Added `diguyLoadManager` class. Members are:

- `diguyLoadManager::add_camera()`
- `diguyLoadManager::add_view_camera()`
- `diguyLoadManager::move_camera()`
- `diguyLoadManager::change_view_camera_settings()`
- `diguyLoadManager::remove_camera()`
- `diguyLoadManager::set_enable_culling()`
- `diguyLoadManager::set_nonvisible_update_rate()`
- `diguyLoadManager::set_zone_bounds()`
- `diguyLoadManager::set_zone_parameters()`
- `diguyLoadManager::set_max_update_period()`

◇ Added new `diguyApp` functions:

- `diguyApp::create_load_manager()`
- `diguyApp::get_load_manager()`
- `diguyApp::destroy_load_manager()`

◇ Added new `diguyCharacter` functions:

- `diguyCharacter::get_bounding_radius()`
- `diguyCharacter::get_default_bounding_radius()`
- `diguyCharacter::set_bounding_radius()`

◇ Added new `diguyApp` class. There is only one `diguyApp` object per application. It is available after the `diguy_*_initialize()` call by calling the static function `diguyApp::get_app()`. The `diguyApp` object will hold data that is application-wide, which in the past has been somewhat artificially placed in the `diguyScenario` class.

◇ Removed support for character history types `DIGUY_HISTORY_TYPE_FIRST` and `DIGUY_HISTORY_TYPE_LAST`. Characters saved using these history types will have their history type set to `DIGUY_HISTORY_TYPE_COMPLETE`.

◇ Moved some static functions from `diguyScenario` to `diguyApp`:

- `diguyScenario::add_default_callback()`
deprecated; use
`diguyApp::add_default_scenario_callback()`
- `diguyScenario::remove_default_callback()`
deprecated; use
`diguyApp::remove_default_scenario_callback()`
- `diguyScenario::remove_default_callback_with_user_data()`
deprecated; use
`diguyApp::remove_default_scenario_callback_with_user_data()`
- `diguyScenario::add_default_callback_script()`
deprecated; use
`diguyApp::add_default_scenario_callback_script()`
- `diguyScenario::remove_default_callback_script()`
deprecated; use
`diguyApp::remove_default_scenario_callback_script()`

◊ Deprecated two `diguyCharacterPath` callback ids. A decision bead or script bead at the appropriate place in the path should be used instead of the deprecated callbacks:

- `CALLBACK_ID_TIME_LEFT` has been changed to
`CALLBACK_ID_TIME_LEFT_DEPRECATED`
- `CALLBACK_ID_TIME_INT0` has been changed to
`CALLBACK_ID_TIME_INT0_DEPRECATED`

◊ Fixed problem with gestures not range checking stage argument. If 0 was passed as the stage in some functions, memory overwrites could have occurred or garbage results could have been returned. Also updated doc of affected functions to specify that stages are between 1 and 3.

◊ Added functions to character for manually applying path bead effects:

- `diguyCharacter::apply_aim_bead()`
- `diguyCharacter::apply_decision_bead()`
- `diguyCharacter::apply_gaze_bead()`
- `diguyCharacter::apply_script_bead()`

◊ Added new `diguyCharacterPathDecisionBead` class.
Members are:

- `diguyCharacterPathDecisionBead::get_name()`
- `diguyCharacterPathDecisionBead::set_name()`
- `diguyCharacterPathDecisionBead::get_distance_into_path()`
- `diguyCharacterPathDecisionBead::set_distance_into_path()`
- `diguyCharacterPathDecisionBead::get_length()`
- `diguyCharacterPathDecisionBead::set_length()`
- `diguyCharacterPathDecisionBead::get_tin()`
- `diguyCharacterPathDecisionBead::set_tin()`
- `diguyCharacterPathDecisionBead::get_duration()`
- `diguyCharacterPathDecisionBead::set_duration()`

- ◊ Added new `diguyCharacterPathAimBead` class. Members are:

- `diguyCharacterPathAimBead::get_name()`
- `diguyCharacterPathAimBead::set_name()`
- `diguyCharacterPathAimBead::get_distance_into_path()`
- `diguyCharacterPathAimBead::set_distance_into_path()`
- `diguyCharacterPathAimBead::get_length()`
- `diguyCharacterPathAimBead::set_length()`
- `diguyCharacterPathAimBead::get_tin()`
- `diguyCharacterPathAimBead::set_tin()`
- `diguyCharacterPathAimBead::get_duration()`
- `diguyCharacterPathAimBead::set_duration()`

- ◊ Added new `diguyCharacterPathGazeBead` class. Members are:

- `diguyCharacterPathGazeBead::get_name()`
- `diguyCharacterPathGazeBead::set_name()`
- `diguyCharacterPathGazeBead::get_distance_into_path()`
- `diguyCharacterPathGazeBead::set_distance_into_path()`
- `diguyCharacterPathGazeBead::get_length()`
- `diguyCharacterPathGazeBead::set_length()`
- `diguyCharacterPathGazeBead::get_tin()`
- `diguyCharacterPathGazeBead::set_tin()`
- `diguyCharacterPathGazeBead::get_duration()`
- `diguyCharacterPathGazeBead::set_duration()`

- ◊ Added function `diguyCharacter::find_waypoint()`. This function finds a waypoint by name.
- ◊ Added functions to access new bead types. Also added find functions for action beads and script beads. The following functions have been added to `diguyCharacterPath`:

- `diguyCharacterPath::find_waypoint()`
- `diguyCharacterPath::find_action_bead()`
- `diguyCharacterPath::find_script_bead()`
- `diguyCharacterPath::get_num_aim_beads()`
- `diguyCharacterPath::get_aim_bead_at_index()`
- `diguyCharacterPath::find_aim_bead()`
- `diguyCharacterPath::get_num_decision_beads()`
- `diguyCharacterPath::get_decision_bead_at_index()`
- `diguyCharacterPath::find_decision_bead()`
- `diguyCharacterPath::get_num_gaze_beads()`
- `diguyCharacterPath::get_gaze_bead_at_index()`
- `diguyCharacterPath::find_gaze_bead()`

- ◊ Added functions to `diguyCharacterGesture` API:

- `diguyCharacterGesture::set_table_action_begin()`
- `diguyCharacterGesture::get_table_action_begin()`

- `diguyCharacterGesture::set_table_action_end()`
- `diguyCharacterGesture::get_table_action_end()`
- `diguyCharacterGesture::get_table_action_now()`
- `diguyCharacterGesture::get_table_action_next()`
- `diguyCharacterGesture::get_rampup_duration()`
- `diguyCharacterGesture::set_rampup_duration()`
- `diguyCharacterGesture::get_rampup_blend_function_name()`
- `diguyCharacterGesture::set_rampup_blend_function_name()`
- `diguyCharacterGesture::get_rampdown_duration()`
- `diguyCharacterGesture::set_rampdown_duration()`
- `diguyCharacterGesture::get_rampdown_blend_function_name()`
- `diguyCharacterGesture::set_rampdown_blend_function_name()`
- `diguyCharacterGesture::get_table_action_rep_duration()`
- `diguyCharacterGesture::get_table_transition_duration()`
- `diguyCharacterGesture::get_default_blend_duration()`
- `diguyCharacterGesture::get_num_table_actions()`
- `diguyCharacterGesture::get_table_action_at_index()`

◇ Added functions for setting initial motion texture parameters:

- `diguyCharacter::set_initial_motion_texture()`
- `diguyCharacter::get_initial_motion_texture()`
- `diguyCharacter::set_initial_motion_texture_gain()`
- `diguyCharacter::get_initial_motion_texture_gain()`

◇ Plugins require a license.

◇ Moved to flexlm 10.1 on Windows for license management.

◇ Particles now work in locked scenarios.

◇ Made characters with non default actors display properly in Vega Prime.

◇ Saving review data now saves all generated data, not just up to the time slider.

◇ Fixed a bug with history saving.

Version 5.3.5

◇ Fixed problem with .flt loader not dealing well with .flt files that contain multiple nodes that have the same name. The .flt loader now more vigorously checks for name collisions.

◇ Removed one unused level of Motion LODs. The unused LOD 6 was not mentioned in documentation nor was it used in motion generation code. This change requires use of DI-Guy Data 5.3.5.

◇ Worked on `diguyCharacterGesture` API:

- `diguyCharacterGesture::get_num_stages()`
deprecated; use
`diguyCharacterGesture::get_num_progression_stages()`
- `diguyCharacterGesture::set_stage_duration()`
deprecated; use
`diguyCharacterGesture::set_progression_stage_duration()`
- `diguyCharacterGesture::get_stage_duration()`
deprecated; use

`diguyCharacterGesture::get_progression_stage_duration()`
`diguyCharacterGesture::get_stage_optimal_duration()`
 deprecated; use
`diguyCharacterGesture::get_progression_stage_optimal_dura`

- ◊ Worked on `diguyCharacterGesture` documentation. Added more which functions can be called for which gesture types.
- ◊ Added type `diguyCharacterGestureType` and corresponding function `diguyCharacterGesture::get_type()`.
- ◊ New experimental shooting functionality:

- Added callback `diguyCharacter::CALLBACK_ID_GOT_SHOT`.
- Added callback `diguyCharacter::CALLBACK_ID_SHOT_SUCCESS`.
- Added new find all hits functionality.
- Added new `diguyShot` class accessible from scripts, records who got shot and where.
- Characters can now shoot multiple targets; added `max_shot_hits` to control the maximum number of shots to report. By default scene objects stop bullets.
- Weapon spread shots test against the octree first (speed up).

- ◊ Fixed some bugs with review data.
- ◊ Experimental plugin infrastructure:

- Added functions `diguyScenario::add_plugin()` and `diguyScenario::remove_plugin()`.
- The following functions can be exported from the dll to plug into DI-Guy code:

- `void initialize(diguyScenario* s)`
- `void deinitialize(diguyScenario* s)`
- `void pre_draw(diguyScenario* s)`
- `void post_draw(diguyScenario* s)`
- `void update(diguyScenario* s)`
- `void reset(diguyScenario* s)`
- `void save_review_data(const char* filename)`
- `void load_review_data(const char* filename)`

- See Documentation for more information.

Version 5.3.4

- ◊ Added new callbacks for use with the `diguyCharacter::add_callback()` function:

- `CALLBACK_ID_CURRENT_APPEARANCE_CHANGED`
- `CALLBACK_ID_CURRENT_HEAD_APPEARANCE_CHANGED`

- ◇ Added new gesture type: table gestures. Table gestures are much like character action tables, and allow a single gesture to employ many different motions.

Version 5.3.3

- ◇ Improved SDK's initialization time. Version 5.3.3 should take about half the time of version 5.3.2 to initialize.
- ◇ Changed default override stage of function `diguyCharacter::create_pose_override()` from 2 to 3.
- ◇ Improved documentation for `diguyCharacter` "Pose Functions" section.
- ◇ Improved documentation for `diguyScenario::set_automatic_motion_lod_switching()` function.
- ◇ Added function `diguyCharacter::end_all_gestures()`, a gentler way than `diguyCharacter::abort_all_gestures()` to stop a character's active gestures.
- ◇ Fixed problem with changing an existing character's type via the `diguyCharacter::set_character_type()` function. The geometry of the character wasn't displaying properly until the scenario was reset. This has been fixed.
- ◇ Fixed documentation for the `diguyCharacterPoseOverride::get_pose_in_radians()` function; it was incorrectly implying that it could be used to read the current pose of the character, when it really only returns the most recent pose as set by `diguyCharacterPoseOverride::set_pose_in_radians()`.
- ◇ Fixed problem where specifying a non-existent default appearance for a character caused a crash.
- ◇ Fixed problem where characters with no links due to config file syntax errors caused a crash.

Version 5.3.2

- ◇ Added missing initial position and orientation functions to `diguyCharacter`:
 - `diguyCharacter::set_initial_position()`
 - `diguyCharacter::get_initial_position()`
 - `diguyCharacter::set_initial_orientation()`
 - `diguyCharacter::get_initial_orientation()`
- ◇ Added `set_name()` functions to most DI-Guy API objects.
- ◇ Added better access to path data such as waypoints and action beads. The following functions have been added to `diguyCharacterPath`:
 - `diguyCharacterPath::update()`
 - `diguyCharacterPath::get_num_waypoints()`
 - `diguyCharacterPath::get_waypoint_at_index()`
 - `diguyCharacterPath::create_waypoint()`
 - `diguyCharacterPath::get_num_action_beads()`

- `diguyCharacterPath::get_action_bead_at_index()`
- `diguyCharacterPath::create_action_bead()`
- `diguyCharacterPath::destroy_action_bead()`
- `diguyCharacterPath::get_num_script_beads()`
- `diguyCharacterPath::get_script_bead_at_index()`
- `diguyCharacterPath::create_script_bead()`
- `diguyCharacterPath::destroy_script_bead()`

◇ Added better access to path shape data such as waypoints. The following functions have been added to `diguyPathShape`:

- `diguyPathShape::update()`
- `diguyPathShape::get_num_waypoints()`
- `diguyPathShape::get_waypoint_at_index()`
- `diguyPathShape::create_waypoint()`

◇ Added new `diguyCharacterPathActionBead` class.
Members are:

- `diguyCharacterPathActionBead::get_name()`
- `diguyCharacterPathActionBead::get_action()`
- `diguyCharacterPathActionBead::set_action()`
- `diguyCharacterPathActionBead::get_distance_into_path()`
- `diguyCharacterPathActionBead::set_distance_into_path()`
- `diguyCharacterPathActionBead::get_length()`
- `diguyCharacterPathActionBead::set_length()`
- `diguyCharacterPathActionBead::get_derive_length_flag()`
- `diguyCharacterPathActionBead::set_derive_length_flag()`
- `diguyCharacterPathActionBead::get_tin()`
- `diguyCharacterPathActionBead::set_tin()`
- `diguyCharacterPathActionBead::get_duration()`
- `diguyCharacterPathActionBead::set_duration()`
- `diguyCharacterPathActionBead::get_derive_duration_flag()`
- `diguyCharacterPathActionBead::set_derive_duration_flag()`
- `diguyCharacterPathActionBead::get_desired_duration()`
- `diguyCharacterPathActionBead::set_desired_duration()`
- `diguyCharacterPathActionBead::get_derive_desired_duration()`
- `diguyCharacterPathActionBead::set_derive_desired_duration()`
- `diguyCharacterPathActionBead::get_motion_reps()`
- `diguyCharacterPathActionBead::set_motion_reps()`
- `diguyCharacterPathActionBead::get_derive_motion_reps_flag()`
- `diguyCharacterPathActionBead::set_derive_motion_reps_flag()`
- `diguyCharacterPathActionBead::get_transition_before_bead_time()`
- `diguyCharacterPathActionBead::set_transition_before_bead_time()`

◇ Added new `diguyCharacterPathScriptBead` class.
Members are:

- `diguyCharacterPathScriptBead::get_name()`
- `diguyCharacterPathScriptBead::set_name()`
- `diguyCharacterPathScriptBead::get_script()`

- `diguyCharacterPathScriptBead::set_script()`
- `diguyCharacterPathScriptBead::get_distance_into_path()`
- `diguyCharacterPathScriptBead::set_distance_into_path()`
- `diguyCharacterPathScriptBead::get_length()`
- `diguyCharacterPathScriptBead::set_length()`
- `diguyCharacterPathScriptBead::get_tin()`
- `diguyCharacterPathScriptBead::set_tin()`
- `diguyCharacterPathScriptBead::get_duration()`
- `diguyCharacterPathScriptBead::set_duration()`

- ◇ Changed API of `diguyWaypoint` object to have more intuitive function names. Old function names have been deprecated but will still work.
- ◇ Improved behavior of reflected characters when the remote scenario is reset. After remote reset, on remote play, local reflected characters now immediately perform their desired actions rather than gradually transitioning into them.
- ◇ Switched to using asynchronous I/O for DIS networking. This improves DI-Guy's robustness on networks with lots of entities.
- ◇ Fixed problem with `diguyPathShape::get_length()`. The `diguyPathShape::update()` function must be called after waypoints have been added to the path shape for the length to be updated. After the `update()` call, `get_length()` will return a valid length. (PR 2852)
- ◇ Fixed problem with `diguyCharacter::get_current_path_index()` function. The value 0 was being returned when the character was not on a path, and when the character was on the first path. The return value for "not on a path" has been changed to -1. (PR 2856)
- ◇ Fixed problem with `diguyCharacter::get_path_name_at_index()` function. An empty string was being returned when the path at the specified index had a zero-length name, and when there was no path at the specified index. The return value for no path at index has been changed to NULL.
- ◇ Fixed problem which, under DIS using DI-Guy custom information, caused newly created reflected characters not to receive custom action information for up to five seconds. Now custom action information is sent immediately after the first entity state PDU is sent.

Version 5.3.1

- ◇ Added support for "live reckoning". Live reckoning provides more accurate display of the behavior of remote characters, using less networking bandwidth.
- ◇ Added network support for multiple concurrent scenario instances. Characters in all scenarios are published to the network. Incoming entities are created in the scenario which is passed to `diguy_dis_go_online()` or `diguy_hla_go_online()`.
- ◇ Improved behavior of reflected entities when the local scenario is reset.

DI-Guy SDK Version 12.5.2 Index

- ◇ Removed some options from network.cfg. It is recommended that the current network.cfg be deleted and let DI-Guy recreate it.
- ◇ Improved behavior of reflected entities when the remote scenario is paused. Now they can move to their correct location even when paused, rather than staying still in a location that is suboptimal.
- ◇ Fixed bug preventing incoming (reflected) non-soldier characters from walking when DI-Guy custom action information is not available.
- ◇ Removed Review Data dt data member from `diguyScenario` objects and from the DI-Guy Scenario UI. Any scenario review data is now recorded at the scenario's tick dt.
- ◇ Added access to the DirectX 9 world stack used by DI-Guy through the new function `diguy_dx9_get_world_stack( )`. This allows users of DI-Guy for DirectX 9 to offset the origin used by characters from 0,0,0. Use of the new function is demonstrated in the example program `$DIGUYprogramming_examplesdiguy_dx9/offset_example`.
- ◇ Made it possible to set the tick dt of a scenario using the function `diguyScenario::set_tick_dt( )`. This was previously possible only in the DI-Guy Scenario UI.
- ◇ Made it possible to turn off review data generation for scenarios. By default, scenarios created in the DI-Guy Scenario editor will have review data enabled; scenarios created using the DI-Guy API will have review data disabled. Turning on review data can add significantly to memory usage. Changing review data settings can be done using the function `diguyScenario::set_history_type( )`.
- ◇ Turned off review data by default for characters created using the DI-Guy API. This significantly reduces the memory overhead of each character. Review data can be re-enabled using the function `diguyCharacter::set_history_type( )`.
- ◇ Added new callbacks for use with the `diguyCharacter::add_callback( )` function:
 - `CALLBACK_ID_PRE_DIE`
 - `CALLBACK_ID_POST_DIE`
 - `CALLBACK_ID_PRE_FIRE_WEAPON`
 - `CALLBACK_ID_POST_FIRE_WEAPON`
- ◇ Fixed problem with setting character's current appearance too soon after the character was created. This caused a crash.

Version 5.2.2

- ◇ New function `diguyCharacter::set_weapon_fires_live_rounds( )` sets whether firing causes collision search for characters to be killed. Default is to fire blanks. Old scenarios that used this functionality need to call this function to re-enable firing to kill other characters.
- ◇ Fixed problem for expressive faces in Vega. If a model without texture or material was drawn just before the expressive face, the expressive face would inherit its color.

- ◇ Fixed various memory leaks.

Version 5.2.1

- ◇ Improved DIS/HLA Networking:
 - Motion behavior improved when receiving standard PDUs in DIS.
 - Motion behavior improved when not using DI-Guy Custom FOM in HLA.
- ◇ Improved vpNet/DI-Guy for Vega Prime interaction:
 - DI-Guy for Vega Prime now shares the network connection created by vpNet (MaK Technologies networking add-on to MultiGen-Paradigm's Vega Prime product).
 - Special programming no longer required to have DI-Guy for Vega Prime/vpNet work together.
 - Users need to maintain two separate modelmaps, but mapping to NULL is now implemented for both vpNet and DI-Guy, so that entities being drawn by other modules in Vega will not be drawn a second time by DI-Guy for Vega Prime and vice-versa.
 - Programming examples have been expanded to show simple examples of vpNet/DI-Guy for Vega Prime interaction. Note that these programming examples are copied to the samples directory of the Vega Prime installation.
 - Added function call `diguy_net_get_character_from_id( )` which returns the `diguyCharacter` associated with the entity id. Works with both DIS and HLA. (see ref. manual).
 - Added option `hla_advisories` to `network.cfg`. When using HLA, if the RTI uses advisories, `hla_advisories` should be set to 1. If not, set this to zero. Mismatch between the settings should be avoided.
- ◇ Fixed problem for expressive faces in Vega. The normals on the head were being scaled incorrectly, causing strange shading artifacts.
- ◇ Fixed problem for expressive faces in Vega. If more than one expressive face was in a scene, all expressive faces would use the texture of the first expressive face.

Version 5.2.0

- ◇ Added dynamic (dll) versions of DI-Guy libraries for win32.
- ◇ Discontinued support of MT and MTd versions of DI-Guy libraries for win32.
- ◇ Fixed nmake-based makefiles for win32 programming examples so that they'll work if DI-Guy is installed in a directory with spaces.

Version 5.1.25

- ◇ Initial release of version 5.1.

Versions 5.1.0 – 5.1.24

- ◇ These versions are 5.1 pre-releases.

DI-Guy Scenario

Note that most changes to the DI-Guy API also apply to DI-Guy Scenario.

Version 9.1.4

- ◇ Scenario can now be stopped, run, and reset in I-Guy Input Mode by hitting Shift-Space (toggle stop and run) an Shift-Ctrl-Space (reset). Keyboard reset can be disabled in I-Guy mode in the I-Guy Page.

Version 9.1.0

- ◇ Large rework of Object pages. There is now one Scenario Objects page that contains all of the pages from previous versions' Scenario Elements, Character Elements, View Settings, Event Handlers, and DI-Guy AI Elements. Pages are separated into multiple tab groups.

Version 9.0.2

- ◇ Fixed bug where network page wouldn't remember current coordinate converter settings when closed and reopened.

Version 9.0.0

- ◇ Near and far parameters of character weapon parameters are now always enabled, and will always be taken into consideration. Before they only had an effect if the "spread" of a weapon was greater than 0.
- ◇ New command line flag added: `-runtime_only`. If this flag is specified, DI-Guy Scenario will not attempt to check out the development license feature but instead will remain in runtime mode, leaving the development license available to another program.

Version 8.6.3

- ◇ Changing a character's character type no longer deletes all action beads. The actions of the existing beads are changed to the nearest equivalent of the new character type.
- ◇ Region/crowd painting now works on scene object grids. delayed loading of grids should be more responsive as well.
- ◇ Fixed issues with feelers: changes made via UI now change all crowd members, 3D rep of feelers now updated immediately when changed and on reset.
- ◇ Fixed `set_playback_mode_play( )` so that it works in decisions.
- ◇ Ctrl-a will now restore previous visibility settings if hit 2 times in a row.
- ◇ Added experimental feeler callback option.
- ◇ Minor UI preference bug fixes.

Version 8.6.2

- ◇ Improved speed of particle system rendering.
- ◇ Fixes to scene object grid rendering.
- ◇ Fixed problem that character initial path could not be deleted.
- ◇ If last character path is deleted, initial position and orientation controls now become enabled as they should.
- ◇ Added preference for changing the fast-forward playback factor. Default is 4. Can be changed in the Preferences dialog.
- ◇ Symbolic View: Characters merging into a crowd symbol should now work outside of ortho_xy view.
- ◇ 3d UI: Added right click pop-up menu to character mode.
- ◇ Added smoothing to ground clamp function.

Version 8.6.1

- ◇ Added one click movie creation to DI-Guy Scenario.
- ◇ More of the UI has up/down sorting arrows.
- ◇ Added is_speaking as a decision bead query.
- ◇ Fixed bug with "this_character" not properly having sub-objects listed in decision beads.
- ◇ Added the ability for labels to reflect character state, accessible in the character label UI.
- ◇ Fixed math issues with scene object sub-culling, feature should be significantly more robust with scaled and rotated scene objects.
- ◇ Character's added to dynamic octtree and scene object characters should require less memory, and be more efficient.
- ◇ Minor improvements to advanced light UI.
- ◇ Action bead page now lists the current action's information.
- ◇ Fixed bug where 1st person I-Guy could not be killed.
- ◇ Better performance when populating or repopulating crowds.

Version 8.5.0

- ◇ Initial DI-Guy SAF release.
- ◇ Added new SAF Elements palette that provides access to DI-Guy SAF objects.
- ◇ Added shortcut ctrl-a: turns off **all** supplemental visuals, providing better viewing for non-demo scenario run.
- ◇ Added "Ortho XY" Projection Mode to Camera settings. This provides an alternative planview that uses an orthonormal projection rather than a perspective-based one.
- ◇ Added "Symbolic View" checkbox to Camera settings. When this is checked characters will be represented as flat polygons rather than their full animated graphics. This, along with the Ortho XY Projection Mode, provides a good SAF-like view of a scenario.

Version 8.0.3

- ◇ Fixed issue with I-Guy not using a joystick for input, even through the UI says it is enabled.
- ◇ Fixed issue with children's bounding volumes not being factored into culling calculation.
- ◇ User interface polish:

DI-Guy SDK Version 12.5.2 Index

- Added more context sensitive input mode buttons.
- Labels identifying what character and path is selected have been replaced with combo boxes, allowing for faster editing of complex scenarios.
- Fixed various other small UI issues.

Version 8.0.2

- ◇ Fixed issue with sound not playing properly when scenario is fast-forwarded.
- ◇ Fixed issues with jittery camera in I-Guy 3rd person mode.
- ◇ Multisync masters now send and receive characters over the network.

Version 8.0.0

- ◇ Initial 8.0 release candidate.
- ◇ Improved plugin interface so that existing plugin name can be changed instead of just being deleted and a new one created.
- ◇ DI-Guy plugin libraries can now be browsed for in the Plugins page, instead of just typing in the library name.
- ◇ Added limits to the Tick dt parameter in the Performance page. The Tick dt must now be ≥ 0.01 seconds, and ≤ 0.1 seconds.

Version 7.1.19

- ◇ Made improvements to DI-Guys sound system, sounds will now pause along with the scenario and can now be fast forwarded and rewind (to a degree).

Version 7.1.16

- ◇ Added 'Init. Up Vector' combo box to Character page. Allows user to set the up vector that the character will use. Previously this could only be done via script of decision.

Version 7.1.15

- ◇ Added '-safe_mode' flag to command line. Passing this flag on the command line will cause DI-Guy Scenario to start up using default preferences, which may resolve some unexpected issues.

Version 7.1.14

- ◇ Added chain settings to Merge Settings UI.
- ◇ Fixed issues with selecting a different path by clicking on one of its waypoints in Path Edit Input Mode.

Version 7.1.12

- ◇ Review data for new scenarios is now turned off by default.
- ◇ New characters created in DI-Guy scenario now have no review data by default. Characters from existing scenarios will not be affected. Note that this removes the ability to rewind scenarios unless some review data is turned back on.

Version 7.1.11

- ◇ Sounds now stop when a scenario loops or is rewind. Added a 'Stop All Sounds' button to the Scenario palette Sound page, to make it

easier to stop playing long sounds.

- ◇ Window menu now always opens UI windows instead of toggling whether they are shown or hidden. If a window is already open, it is brought to the top of the window stack.
- ◇ The F-key shortcuts for opening UI windows have changed behavior. Pressing F2 for the Time Control window, for example, will make the Time Control window visible by either opening it or bringing it to the top. Old behavior toggled whether windows were shown or hidden. Use Shift-F-key to use old toggle behavior (hide if showing, show if hidden).
- ◇ Character Labels Visibility is now set to 'None' in the default .dss file.

Version 7.1.8

- ◇ Fixed problem with restoring all event mappings when reloading a previously saved scenario.

Version 7.1.7

- ◇ Switch to new installer setup. Should better detect and remove previous versions, and in general be more stable.
- ◇ Added preliminary support for TerraPage .txp files in scene objects.
- ◇ Fixed issue with OpenAL not being able to play sounds on some machines.

Version 7.1.6

- ◇ Added 3rd Parameter to Decision Beads.
- ◇ Added diguyCharacter::create_and_force_bridge_spath() and diguyCharacter::force_local_path to Decision Beads editor.
- ◇ Application now asks user to confirm whether an existing file can be overwritten when Save As is selected.
- ◇ Added Comment field to Decisions and Decision Beads.
- ◇ Added new cursor to represent current input mode.
- ◇ Added events page to character as well as signal and sensor region. Same basic functionality as the event mapper but more distributed UI design.

Version 7.1.4

- ◇ Added functionality to the character elements page that allows character list to be reordered using up and down arrows next to the list box.
- ◇ Character list box will now display disabled characters in gray.

Version 7.1.3

- ◇ Added offset XYZ line edits for aiming at characters on Aim Bead page.

Version 7.1.2

- ◇ Added new "Character" Input Mode. While in this mode clicks and drags will have the following effects:

DI-Guy SDK Version 12.5.2 Index

click and drag translate character
drag
shift-click and drag rotate character

- ◇ Made 'Delete' key work on more object types in the 3D window, depending on the current Input Mode:

Character or PeopleBlitzer delete current character

Path Edit or Path Insert delete current waypoint

Action Edit or Action Insert delete current action bead

Sensor Edit or Sensor Insert delete current sensor region

Group delete current group

Version 7.1.1

- ◇ Added new "Group" Input Mode. While in the mode clicks and drags will have the following effects:

click and drag translate all group characters

shift-click and drag rotate all group characters

ctrl-click add or remove clicked character to group

- ◇ Added Chain Sim Settings page to the Scenario Elements palette.
- ◇ Added edit boxes to event beads (Aim, Decision, Gaze, and Script beads) that allow beginning and end of beads to be tied to specific waypoints or action beads.
- ◇ Added ability to set character label colors on the Character page.
- ◇ Added optional labels for action beads in the 3D window that show the action name and the time at which the action begins.
- ◇ Current selected character index is now saved in .dss file and restored when the scenario is loaded.
- ◇ Revamped Plugins page. Each plugin now has these parameters:

Required By Scenario If checked, this plugin is required for proper execution of this scenario. When the scenario is loaded this plugin will be loaded as well if it is not already active. Default is unchecked.

Unload on Scenario Close If checked, this plugin will be unloaded if this scenario is closed or a new scenario is opened. Default is unchecked.

DI-Guy SDK Version 12.5.2 Index

Accepts Mouse Input If checked, this plugin will receive mouse events via the mouse plugin functions when the input mode is "Plugin". Default is checked.

- ◇ Added new camera frustum culling functionality that works on the subshapes of a scene object. The functionality can be turned on via a check box in the performance page, or via `diguyScenario::set_scene_object_sub_culling_enabled()` and `diguyScenario::get_scene_object_sub_culling_enabled()`. Potentially this can lead to 100%–200% rendering speed increase for large well segmented terrains. Note currently scene object culling must be on for this to work.

Version 7.1.0

- ◇ Revamped Cut and Paste operations to use the merge functionality that makes sure objects added to a scenario have unique names.

Version 7.0.8

- ◇ Camera speed UI input was broken, should be responsive again
- ◇ Hitting cancel in particle system texture map property widget will no longer delete the currently selected texture.
- ◇ Fixed Camera Affected by Script events check box; was actually an SDK fix.

Version 7.0.5

- ◇ Enhanced character path following so that waypoints that have a roll parameter specified will roll the character as it travels along the path.
- ◇ Enhanced character path following so that characters can now pitch "upside-down" along paths.

Version 7.0.3

- ◇ Look At button now works for reflected characters.

Version 7.0.1

- ◇ Velocity Noise on particles will now get transformed with the parent's orientation.
- ◇ Fixed problems with various guide algorithms not working well if the desired z coordinate is not at the same altitude as the character.
- ◇ Re-enabled "-C cfg_file" command line flag.

Version 6.1.10

- ◇ Fixed problem that prevented some parameters from being editable in the Decision editor; in particular, when param 2 was an appearance or a string.
- ◇ Fixed aim trajectory rendering to more consistently render and use the proper color.
- ◇ Fixed problem with I-Guy characters not being able to move forward and back.

DI-Guy SDK Version 12.5.2 Index

- ◇ Particles are more aggressive about requesting that octtree is rebuilt.
- ◇ High velocity particles, stuck particles and bouncing particles all interact better after colliding with scene geometry.
- ◇ **BDI_LOG_FATAL** errors will behave better in DI-Scenario, giving the option to save before exiting.
- ◇ Fixed decision beads pseudocode description crash.

Version 6.1.9

- ◇ Fixed problem with OpenGL renderer that prevented objects with subfaces from rendering correctly.
- ◇ Fixed problem with Draw Mode radio buttons in the Preferences dialog not having the correct effect on rendering.
- ◇ Changed info popups to show up as dialog windows separate from the main window. This prevents the Primary 3D Window from un-maximizing when an info popup pops up.
- ◇ Fixed problem that scene objects did not correctly look in custom directory for models.
- ◇ Added code to check positions of DI-Guy Scenario windows; if windows will not be visible (are off-screen, etc.), they are moved to screen coordinate 0,0.
- ◇ Updated Networking Settings page to be consistent with V6 UI.
- ◇ Numerous fixes to Networking Settings page.
- ◇ Added About box with version information.
- ◇ Fixed bug that preferences weren't correctly saved if app was closed by clicking 'X' button in upper right corner.
- ◇ Fixed various problems with values entered into edit boxes being clobbered.
- ◇ Added plugins page to DI-Guy Scenario to match page from V6 UI.

Version 6.1.8

- ◇ Fixed world space collisions for particles, functionality was temporally broken.
- ◇ Particle UI should be now be almost entirely working in DI-Guy Scenario.
 - Added load particle descriptions button to UI.
 - Added collision plane rendering check box to UI.
 - Added check box to particle page that optionally turns off placeholders.
 - Added orientation flag to particles, useful for parented emitters.
 - Renaming particles should validate names and echo to other appearance menus.
 - Adding or removing particle descriptions now properly updates the appearance section of the character page and people blitzer.
 - Particle manager was getting updated unnecessarily causing particles to playback 2x as fast.
- ◇ Fixed a focus out problem that yielded a hard lock up if a load manager error message was produced.

DI-Guy SDK Version 12.5.2 Index

- ◇ Waypoints and sensor regions visuals weren't being properly updated when data changed.
- ◇ Editing a Scene object triggers the unsaved flag more often.
- ◇ Fixed a glitch where shadows and the profiler wouldn't work together.
- ◇ Added UI for setting character and scene object culling active.
- ◇ Added UI for setting character and scene object bounding volume visualization active.

Version 6.1.7

- ◇ Added UI for setting the character's label text and controlling whether the label is visible to the Character page.
- ◇ Added new visibility option for Character Labels on the Visibility page: Per Object.
- ◇ Major reworking of the DI-Guy Scenario user interface in preparation for the DI-Guy 7 release. A version of the program with the old interface (henceforth called the V6 UI) will be available in releases for awhile with a different filename. As time goes on new features that appear in DI-Guy Scenario may not appear in the V6 UI.
- ◇ Changed the sound module of DI-Guy Scenario to be based on OpenAL. This fixed a problem with sounds playing back too fast on certain systems, and allows for 3D position sounds.

Version 6.1.6

- ◇ Added "+plugin " command line argument. This will load the specified plugin as the application is starting up. This is essentially the same as calling `diguyApp::load_plugin( )` before the initial scenario is created.
- ◇ Added "Plugin" page. This allows a scenario designer to specify which plugins are required for proper execution of the scenario.
- ◇ Fixed crash bug that occasionally occurred when the "Trigger Now" button was clicked on the Script and Decision pages.
- ◇ Added UTM Coordinates to the Networking page.
- ◇ Added "Decisions", which are similar to Script Events but have a UI for constructing logic similar to Decision Beads.

Version 6.1.4

- ◇ Fixed minor problems with decision bead page. The parameters for the sensor region function `contains_character_by_type_and_appearance( )` did not include quotes and were not alphabetized.
- ◇ Added buttons "Add All" and "Remove All" to Group page.

Version 6.1.1

- ◇ Added Load Manager page to the Scenario window. Now users can enable load management, and adjust load management parameters from within DI-Guy Scenario.
- ◇ Particle warm up time should now work properly on reset.
- ◇ Added particle system life time, after which it will no longer emit particles.

DI-Guy SDK Version 12.5.2 Index

- ◇ Added new Particle position and velocity noise options, should allow for better control and movement options.
- ◇ Added character variable querying to decision beads, can now query variables with: `variable_equal_to_(float, int, string)`, `variable_less_than_(float, int, string)` and `variable_greater_than_(float, int, string)`.
- ◇ Added character variable actions to decision beads, can now have actions that affect variables, using `variable_set_value_as_(float, int, string)` and `variable_increment` and `decrement`. `string`) and `variable_greater_than_(float, int, string)`.
- ◇ Can now use decision beads to check whether someone's been shot, as well as where the shot hit.
- ◇ Changed how decision beads get pointers to objects; previously if an object was referenced multiple times in a decision bead (e.g., multiple queries were made of character "soldier-1" in the if clauses), the same object was searched for multiple times. Now a pointer to the object is just searched for once.
- ◇ Added Variable page to Character window. This page allows for creation, deletion, and editing of character variables. Character variables are almost exactly the same as scenario variables, but are private to each character instead of being shared across the entire scenario.

Version 6.1.0

- ◇ Ctrl-'u' now pushes a scenario undo onto the undo stack.
- ◇ Space bar toggles between play and pause; Ctrl-space bar does a reset.

Version 6.0.8

- ◇ DI-Guy Scenario now properly sets the network Force ID for outgoing entities based on group membership. Groups `net_friendly`, `net_opposing`, and `net_neutral` should be used to indicate the Force ID of outgoing entities.
- ◇ DI-Guy Scenario now assigns incoming entities to group `net_friendly`, `net_opposing`, or `net_neutral` based on the incoming Force ID.
- ◇ Incoming entities cannot now be assigned to or removed from groups. As well, groups to which incoming entities belong cannot be deleted.

Version 6.0.2

- ◇ Fixed problem with "Deflate" compression method for movie frame capture.
- ◇ Changed label of "Reset" button on Movie Settings page to "Reset Next Frame Number", to differentiate it from the Reset button on the Control palette.

Version 6.0.1

- ◇ Added setting of Scenario Review Data on Scenario Performance page to .dss file. It was not being saved and restored as scenarios were saved and loaded.

DI-Guy SDK Version 12.5.2 Index

- ◇ Changed scenario "Loop" functionality. If no characters have review data, the scenario will reset and play rather than rewind and show review data.
- ◇ Removed unused "Altitude" edit box from Networking property page. (PR 3002)
- ◇ Changed gaze bead logic so that faster gazes are possible. The desired locate time for gaze beads was effectively limiting the maximum angular velocity. Now if a gaze bead overrides gaze parameters, it also turns off use of the desired locate time for each link. (PR 3030)

Version 6.0.0

- ◇ Added "GoThere" camera movement. In Camera mode, if there is neither a Look At character nor a Look From character selected and the camera projection mode is "free camera", if you hold the Shift key and left-click on anything in the scene, the camera will rapidly travel to what you Shift+clicked on and turn around to face the location from which the camera came. This also works from modes other than camera mode, if you press and hold Alt+Shift while left-clicking.
- ◇ Changed Aim Bead UI to match Gaze Bead UI. Note that now aiming is also precise, and dynamic. See DI-Guy aiming change above.
- ◇ Fixed problem with current light and fog settings being overwritten when a .dss file was loaded. (PR 2019)
- ◇ Fixed problem with primary view of scenario returning 0 when `diguyScenarioView::is_showing( )` was called on it. The correct value is now returned. (PR 2954)
- ◇ Added ability to change window title of primary view using the call `diguyView::set_window_title( )`. (PR 2953)
- ◇ Fixed problem with looking at scene objects that are translated away from the origin. (PR 2722)
- ◇ Fixed problem for character paths on load. Paths that weren't the initial path of the character weren't being fully initialized until they were used by the character. Now all paths receive an initial update call on load. (PR 2791)
- ◇ Added new command line argument: `-no_info_popup_server`. Adding this argument to the command line causes DI-Guy Scenario to not attempt to start the program `DIGuyInfoPopupServer`. If the server is not started, info popup windows will not appear.
- ◇ Fixed problem that if the Initial Script Filename on the Script Event property page specified a script that didn't exist, no warning was given. Now a log message at Warn level is printed for a missing script file. (PR 2826)
- ◇ Fixed problem that prevented Performance settings from taking effect unless the scenario was saved and reloaded.
- ◇ Added edit box for setting character's initial motion texture gain. (PR 2845)
- ◇ Added UI preference for turning off the "DI-Guy Scenario by *Boston Dynamics*" overlay.

Version 5.3.5

- ◇ Changed edit box for setting character's initial motion texture into a combo box that shows which motion textures are available.

Version 5.3.2

- ◇ Small improvement to particle system UI, fixed outstanding issues with texture map browser widget. Color Widgets go right to a color picker. Clicking on static text opens subitems.
- ◇ Particle Systems now populate better between ticks, i.e. release intervals smaller than the update time now work.
- ◇ Performance improvements to the particle systems.

Version 5.3.1

- ◇ Removed Review Data dt data member from `diguyScenario` objects and from the DI-Guy Scenario UI. Any scenario review data is now recorded at the scenario's tick dt.
- ◇ Split Visibility / Performance page into two separate pages named Visibility and Performance.

Version 5.3.0

- ◇ Made it possible to turn off review data generation for scenarios. By default, scenarios created in the DI-Guy Scenario editor will have review data enabled; scenarios created using the DI-Guy API will have review data disabled. Turning on review data can add significantly to memory usage. Changing review data settings can be done using the function `diguyScenario::set_history_type( )`, or using the Review Data Type radio buttons on the Visibility / Performance page of the DI-Guy Scenario editor.

Version 5.2.3

- ◇ Added joystick gain settings to Preferences. The gains allow DI-Guy Scenario users to adjust how sensitive the joystick will be. Gains of 1.0 retain previous behaviors. Gains less than 1.0 make joystick control less sensitive; i.e., it will take a larger movement to achieve a response. Gains more than 1.0 make joystick control more sensitive; i.e., it will take a smaller movement to achieve a response.

Version 5.2.1

- ◇ Special Effects: DI-Guy Scenario now offers a new add-on option for particle-based special effects. Examples of particle effects include smoke, fire, debris, missile trails, rotor wash, explosions, and dust trails. A Particles tab is now available in the Scenario palette for easy control of effects parameters include size, number, color, texture map, velocity, wind effect, and gravity.
- ◇ Fixed crashing bug when undoing path changes.
- ◇ Fixed minor issues with using soldier character backward actions and companion waypoints.
- ◇ Fixed problem where character sounds were not consistently playing when a scenario looped.

DI-Guy SDK Version 12.5.2 Index

- ◇ Fixed problem where path shapes were not always being drawn when they should have been.
- ◇ Fixed problem where having path shapes in a scenario caused the program to crash on exit.
- ◇ Fixed crashing bug when shutting down a scenario that contains characters with guides.

Version 5.2.0

- ◇ Added ability to show character names as floating labels next to the character in the 3D window. Name labels can be shown for current character only, all characters, or no characters. Access via the Scenario | Visibility / Performance property page.
- ◇ Added "Look At" buttons for characters, waypoints, and path beads on respective Elements pages.
- ◇ Added special effects such as smoke and fire.
- ◇ Fixed perl script interpreter so that it will work if DI-Guy is installed in a directory with spaces.
- ◇ Fixed problem where LOD range scale factors weren't taking effect for scene objects.

DI-Guy AI

Version 9.0.1

- ◇ Fixed bug where characters were improperly going through walls.
- ◇ Fixed a number of minor crashes with creating behavior paths.

Version 9.0.0

- ◇ To enable DI-Guy AI in programs using the DI-Guy SDK, the header file "diguy_ai.h" must be included, and the functions `diguy_ai_initialize()` and `diguy_ai_deinitialize()` must be called at the appropriate times.
- ◇ New command line flag added to DI-Guy Scenario for optionally disabling DI-Guy AI. By default DI-Guy Scenario will look for the diguy_ai license feature and check it out if it is present. By putting "-module diguy_ai" on the command line, DI-Guy Scenario will not attempt to enable DI-Guy AI.
- ◇ Fixed problem with DI-Guy AI agents having trouble killing animals.
- ◇ Painting a non-continuous region will be more likely to successfully create a behavior path, the longest path from the start point to the mouse up location is used.
- ◇ Fixed issues with feelers: changes made via UI now change all crowd members, 3D rep of feelers now updated immediately when changed and on reset.
- ◇ Fixed problem that agent feeler collide distance could not be greater than turn distance.
- ◇ It is now possible to add a feeler callback to enable DI-Guy AI agents to interact better with static geometry in the SDK. See the following functions for more details:

DI-Guy SDK Version 12.5.2 Index

- `diguyCharacter::agent_set_feeler_function()`
- `diguyCharacter::agent_get_feeler_function()`
- `diguyScenario::agent_set_default_character_feeler_function()`

◇ Crowds and agents can now flee a group. Accessed via:

- `diguyCharacter::agent_flee_group()`
- `diguyCrowd::agents_flee_group()`

Version 8.6.3

◇ Crowd UI Improvements:

- Selected member will now only show postures and variants that it has available.
- Selected edit mode with display better visual cues indicating which crowd members will be affected by changes
- Added look at member button.

◇ Crowd Blitzing: Blitzing a crowd into an area that is non-contiguous will now attempt to find the longest path from the start point to the end point, This should fix a frequent failure case with the Blitzer.

Version 8.6.1

◇ DI-Guy SAF references changed to DI-Guy AI.

Version 8.6.0

◇ New DI-Guy SAF functions:

- `diguyCharacter::agent_set_current_behavior_region_border_`
- `diguyCrowd::set_all_members_enabled()`
- `diguyCrowd::set_current_behavior_region_border_is_solid()`
- `diguyCrowd::is_current_behavior_and_focus_group()`
- `diguyCrowd::is_current_behavior_and_focus_character()`
- `diguyCrowd::is_current_behavior()`
- `diguyCrowd::is_crowd_dead()`
- `diguyScenario::set_create_network_crowds()`
- `diguyScenario::get_create_network_crowds()`

Version 8.5.0

◇ Initial DI-Guy SAF release.

◇ Added/updated DI-Guy SAF classes:

- `diguyAgentParams`
- `diguyCrowdProfile`
- `diguyCrowd` (extensive changes, promoted from experimental)

◇ New DI-Guy SAF functions:

- `diguyScenario::destroy_crowd()`

- `diguyScenario::get_num_crowds()`
- `diguyScenario::find_crowd()`
- `diguyScenario::get_crowd_at_index()`
- `diguyScenario::create_crowd_profile()`
- `diguyScenario::destroy_crowd_profile()`
- `diguyScenario::get_num_crowd_profile()`
- `diguyScenario::find_crowd_profile()`
- `diguyScenario::get_crowd_profile_at_index()`
- `diguyScenario::create_group_in_region()`
- `diguyCharacter::agent_attack_group()`
- `diguyCharacter::agent_flee_character()`
- `diguyCharacter::agent_flee_location()`
- `diguyCharacter::agent_remove_all_flee_objects()`
- `diguyCharacter::agent_mingle_in_region()`
- `diguyCharacter::agent_pursue_character()`
- `diguyCharacter::agent_pursue_group()`
- `diguyCharacter::agent_travel_behavior_path()`
- `diguyCharacter::agent_travel_path_shape()`
- `diguyCharacter::agent_travel_region_border()`
- `diguyCharacter::agent_wander_region()`
- `diguyCharacter::agent_stop_behavior()`
- `diguyCharacter::agent_set_current_focus_character()`
- `diguyCharacter::agent_set_current_focus_group()`
- `diguyCharacter::agent_get_initial_params()`
- `diguyCharacter::agent_get_current_params()`
- `diguyCharacter::agent_set_current_params_from_profile()`
- `diguyCharacter::agent_get_current_focus_character()`

DI-Guy Motion Editor

Version 9.0.1

- ◊ Added sample bvh file to motions directory, should help document how DI-Guy expects bvh files to be structured.

Version 9.0.0

- ◊ If motions are added to an action table, the proper default actor is now set automatically.
- ◊ Improved support for editing offset poses, should be much easier, and not cause abrupt joint behavior.
- ◊ Changing the actor of a motion arc now immediately applies the new scale to the motion, improvements to actor choosing, motion scaling, and importing code.
- ◊ Fixed a number of action table editing issues: including hard crash with undo, "remove unused motions" causing corrupted action tables.
- ◊ General UI improvements:
 - +/- keys behave more intuitively with regards to the yawpitchroll of the poser.
 - Shift key alters magnitude of +/- change.
 - Poser editing gizmo now auto-scales based on camera distance.

DI-Guy SDK Version 12.5.2 Index

- Editing mode is now displayed in 3D window title bar.
- Initial directories for me project, importing motions and saving motions are now set to expected values.

Version 8.6.3

- ◇ Fixed issue with zero length animations causing div by zero.
- ◇ Fixed minor bug with action tables not properly updating when unused motions were removed.

Version 7.1.11

- ◇ Fixed problem that many error messages would appear when saving action tables.

Version 7.1.0

- ◇ Fixed various problems with setting variants and angle extents for 2-axis motions in the Action Table Editor.
- ◇ Added sliders for testing aim actions in the Action Table Editor.

Version 6.1.9

- ◇ Fixed a number of bugs introduced with the upgrade to high precision positions.

Version 6.1.0

- ◇ Added new gesture table editor, allows for the modification of the action tables of table gestures.
- ◇ Null motion arcs are now editable, with blend type and duration as parameters.

Version 6.0.6

- ◇ BVH file import now supports joint translations in addition to rotations.
- ◇ Resultant BDM files contain only relevant variables.

Version 6.0.0

- ◇ Initial 6.0 DI-Guy Motion Editor release.

DI-Guy Character Viewer

Version 8.0.2

- ◇ Added view selection buttons that load new camera settings.

Version 6.1.9

- ◇ Fixed a number of bugs introduced with the upgrade to high precision positions.
- ◇ Fixed character transparency.
- ◇ Fixed selecting the default appearance.

Version 6.0.2

- ◇ Added default settings file that shows Character Type *Appearance* Action selection page.

Version 6.0.1

- ◇ Fixed problem where appearances that used a different actor than the default actor for a character did not correctly display.

Version 6.0.0

- ◇ DI-Guy Character Viewer added to DI-Guy Applications Suite. Allows quick review of character types, appearances, and actions.

DI-Guy for Vega Prime

Version 9.0.3

- ◇ vpNet 2.2 should now be enabled.
- ◇ Fixed issue with skyLight lighting and shaders not being properly applied.
- ◇ Fixed Memory leak when characters were unrefed to zero. Note a slight change to the basics example.

Version 9.0.1

- ◇ Fixed issue with new vpDiguys not having their locations properly set when reading out of an acf.
- ◇ Fixed issue with vpDiguyCharacters parented to vpTransforms not being properly created.

Version 9.0.0

- ◇ DI-Guy for Vega Prime now works with VP 2.2.

Version 8.6.3

- ◇ Fixed crash on exit when a vpDiguyCharacter parent was not a scenario.
- ◇ Fixed bug where characters would multiply render if they were parented to a vpObject
- ◇ parented and translated characters will should start broadcasting over the network with proper values.

Version 8.0.3

- ◇ Moved example_acfs to their own subdirectory instead of in simple_playback directory.
- ◇ Added warning about setting channel LOD scale to zero, can cause character to not draw.
- ◇ Fixed an issue where not having a valid license could cause vp applications to crash.
- ◇ Fixed licensing issue effecting vpDiguyvpNetHLA/stealth.
- ◇ Basics Programming example now demonstrates ground clamping. As well as some other polish to programming examples.

Version 8.0.2

- ◇ Improved reporting of version information during installation.

Version 8.0.0

- ◇ Initial 8.0 release candidate.
- ◇ vpDiguy using vpNet should now support all coordinate conversion routines that vpNet supports.
- ◇ Memory usage and rendering performance improvements.

Version 7.0.8

- ◇ Fixed HLA networking licensing issue. Shouldn't need explicit hla license with vpNet.
- ◇ Added new autogenerated vpDiguy SDK docs.
- ◇ Added code to basics example that demonstrates creating a character after configure.
- ◇ Added documentation about vpDiguy openAL audio module's incompatibility with vpAudio module.
- ◇ Added static versions of vpDiguy libraries, updated autolinking code in vpDiguyCpp.cpp.

Version 7.0.7

- ◇ Fixed a number of vpNet related bugs.
- ◇ NOTE: In order to make vpNet interaction more solid reading of UTM coordinates out of Scenario Objects in vpDiguy has been disabled. UTM coordinates should be specified explicitly in the Net Connection gcf panel and the DI-Guy Scenario Network Page.

Version 7.0.6

- ◇ Fixed a number of recreation/looping related bugs.

Version 7.0.3

- ◇ Turned on vpNet/vpDiguy networking.
- ◇ Updated all the acfs to be vp 2.0 compliant.
- ◇ Updated vpDiguy gcf to include new 7.0 content.
- ◇ Fixed bug in UTM support.
- ◇ Fixed problem if multiple diguy particles attempt to instantiate the same type of vpFX. Fixed memory leak as well.
- ◇ Fixed divide by zero bug in networking code, could cause 0th character to never show up.
- ◇ Fixed bug where a character with an appearance change would not be recreated properly if a scenario was looping.

Version 7.0.0

- ◇ Fixed a couple of vpDiguyScenario function calls to print a warning if the functions are called before configure, instead of crashing.
- ◇ [diguyCharacter](#)'s position syncs with vpDiguyCharacter on creation.
- ◇ Distributed rendering should now support mapping from DI-Guy

Version 6.1.11

DI-Guy SDK Version 12.5.2 Index

- ◇ Consolidated networking examples, moved `vpnet_(send/receive)_(dis/hla).acf` into `simple_playback` because they didn't have unique c++ code.
- ◇ Added `postLoad` function to `vpDiguyCharacter` building code to try and enable sensor prime.
- ◇ Added `vpDiguyScenario::set_time_independant_of_vp_sim_time()` and `vpDiguyScenario::get_time_independant_of_vp_sim_time()` Allows time to be disconnected from sim time.

Version 6.1.10

- ◇ Renamed `vpDiguyScenario::SetSyncCameraEnable()` to `vpDiguyScenario::setSyncCameraEnable()`.

Version 6.1.9

- ◇ Added per scenario time offset code.

Version 6.1.8

- ◇ Code now compiles against Vega 2.0 libraries:
- ◇ Added 2.0 style copy constructor for `diguyCharacters` to use in cloning.
- ◇ Fixed a bug where locked off joints issued warnings.
- ◇ Fixed a bug where some dynamic matrixes weren't being properly updating, fixes gazing inside Vega.
- ◇ Altered how `diguyScenarios` get loaded to accommodate new merging code.
- ◇ Unsupported Vega sound module now uses openAL.

Version 6.1.1

- ◇ `gcfs` and `xsd` no longer use "DI-Guy" in the naming convention; instead "Diguy" is now used. This avoids uncomparable code generated from `lynxPrime's export .cpp` functionality, as well as being more consistent with Vega conventions.

Version 6.0.8

- ◇ Fixed serious bug affecting HLA networking in DI-Guy for Vega Prime. This bug was causing various problems, ranging from missing or extra apparent network entities, to entities appearing at the wrong location.

Version 6.0.7

- ◇ Changed all Vega Prime code/acfs that specify "DI-Guy" to "Diguy". This should allow cpp exports to work better.
- ◇ Renamed `vpDiguy.h/cpp` to `vpDiguySettings` to better match acf info.
- ◇ Added Vega Prime view settings example that shows off camera sync code.
- ◇ Added Vega Prime simple example that shows how to initialize and run with no acf or dss.

Version 6.0.6

- ◇ Fixed problem which inadvertently disabled standard Vega Prime "X" key functionality (enable / disable the position strategy for all

observers) in a vpDiguy application.

Version 6.0.5

- ◇ Fixed problems preventing DI-Guy for Vega Prime from working with DIS/HLA for Vega Prime. DI-Guy for Vega Prime now works with Vega Prime 1.2.2 from MultiGen Paradigm and vpNet 1.2.2 from MAK Technologies. Prior versions of vpNet are not supported.

Version 6.0.2

- ◇ Switched bdiString to bdiString* in vpDiguyCharacter.h, to remove a dependency on a header file that is not distributed.

Version 6.0.1

- ◇ Fixed problem in vpDiguy DIS networking. Outgoing entities were being published with entity numbers which conflicted with those being used by Vega Prime entities. On the receiving end, this would cause characters to pop from place to place.

Version 5.3.4

- ◇ Added a circle example program to programming_examples/vpdiguy.

Version 5.3.3

- ◇ DI-Guy characters types are now listed in a Character Type combo box.
- ◇ A character type sensitive list of appearances are now listed in an Appearance combo box.
- ◇ A character type sensitive list of actions are now listed in an Action combo box.
- ◇ A character type sensitive list of head appearances are now listed in a Head Appearance combo box.
- ◇ Altering the xyz position, heading or scale factor of a character now reflects on the character immediately.
- ◇ Adding a ground clamp to a DI-Guy Scenario node now adds the ground clamp to all of its children that don't currently have a ground clamp.
- ◇ Added new DI-Guy Settings functionality, including turning on/off scripting, sound effects, facial expressions, and the mapping of DI-Guy particles to Vega particles.
- ◇ Added new options to DI-Guy Scenario panel.
- ◇ Added toggle for showing scene objects found in the scenario file.
- ◇ Added Sync Camera option that will slave the observer/motion controls to the DI-Guy camera.
- ◇ Adding a DI-Guy character directly to a scene now creates one without a scenario being necessary.
- ◇ DI-Guy characters dropped into a vpNet model map are now correctly cloned. (Requires the latest version of vpNet.)

Version 5.2.1

- ◇ Better integration with MaK's vpNet module (see above).
- ◇ Installation now occurs as a separate module on the DI-Guy CD.

- ◇ Installation now automatically copies samples, documents to the Vega Prime installation as well as registering the DI-Guy for Vega Prime module.

DI-Guy Data, 9.0 and Earlier

Version 9.0.1

- ◇ Added character soldier_sa80, a British soldier with desert and temperate appearances using a sa80, based on the soldier_07 character. Available appearances:
 - uk_desert_sa80
 - uk_temperate_sa80
- ◇ Added character "projectile" with meters per second motions. Character projectile appearances:
 - rpg_fired
 - sa7_fired
 - stinger_fired
 - javelin_fired
 - 120mm_tank_round_fired
 - at4_fired
 - 40mm_fired
- ◇ Added sound files: m203_hit.wav, fire_m203.wav, arabic_street_sounds.wav sa7_flight.wav, sa7_fired.wav,

DI-Guy SDK Version 12.5.2 Index

stinger_fired.wav, stinger_flight.wav, javelin_flight.wav,
javelin_impact.wav, at4_fired.wav, at4_flight.wav at4_impact.wav,
javelin_fired.wav, pk74_fired.wav, m1a1_fired.wav

Version 9.0.0

◇ Some character types have been made deprecated, and made "legacy" character types. If these character types are used in an existing .dss file or are created by a `diguyScenario::create_character( )` call they will be created, but by default they will not show up in any DI-Guy Scenario character type selection controls. The legacy character types are:

- soldier_m249
- soldier_m4
- soldier_qk
- soldier_signals
- soldier_fire
- soldier_pistol
- soldier_melios
- afghan_child_crowd
- male_pedestrian3
- mped_sweeper
- mped1_misc
- mped2_misc
- mped3_misc
- mped5_crowd2
- mped5_crowd3
- mped5_crowd4
- mped5_crowd5
- mped5_crowd6
- fped2_misc
- fped3_crowd2
- suspect3
- marshaller
- esil
- soldier_precise
- launcher
- suspect
- suspect2
- fped_mother
- sergeant

Version 8.0.0

◇ New character types:

- chain_simulation
- civilian_cc
- soldier_cc
- soldier_at4
- soldier_javelin

DI-Guy SDK Version 12.5.2 Index

- soldier_m249
- soldier_m9
- soldier_m4
- train

◇ New human appearances:

- 3 Security Guards
- Hazmat Suits level A B and C
- Assortment of Hazmat props
- 10 different Close Combat civilians
- Close Combat BDU and DCU with M9, M249, M240, M60, AT4, Javelin M4 with M203, M16 with M203 Stinger weapons
- Close Combat Opfor with SA16, RPG7, AK47 and PK74 weapons
- 3 Iraqi Insurgents
- 2 Iraqi Soldier
- 2 Iraqi Police

The following soldier appearances can have M9, M249, M240, M60, AT4, Javelin M4 with M203 and M16 with M203 weapons:

- 4 BDU with IBA
- 4 ACU with IBA
- 4 USMC with IBA
- 4 USMC desert with IBA

◇ New turret vehicle and technical vehicle appearances:

- gaz_truck_w_ZSU23_turret
- gaz_truck_w_ZSU23_turret_blownd
- toyota_pickup_tech_turret
- toyota_pickup_tech_turret_blownd
- humvee_50cal_grn_turret
- humvee_50cal_grn_turret_blownd
- humvee_50cal_des_turret
- humvee_50cal_des_turret_blownd
- m1a2_abrams_grn_turret
- m1a2_abrams_grn_turret_blownd
- m1a2_abrams_des_turret
- m1a2_abrams_des_turret_blownd
- t72_tank_grn_turret
- t72_tank_grn_turret_blownd
- t72_tank_des_turret
- t72_tank_des_turret_blownd
- m2a3_bradley_grn_turret
- m2a3_bradley_grn_turret_blownd
- m2a3_bradley_des_turret
- m2a3_bradley_des_turret_blownd

DI-Guy SDK Version 12.5.2 Index

- leclerc_tank_descamo_turret
- leclerc_tank_descamo_turret_blownd
- leclerc_tank_camo_turret
- leclerc_tank_camo_turret_blownd

◇ Added Motions to Soldier and Soldier2 characters:

- stand_ready_8th_turn_left
- stand_ready_8th_turn_right
- stand_ready_qtr_turn_left
- stand_ready_qtr_turn_right
- stand_aim_extreme
- kneel_aim_extreme
- walk_aim
- walk_aim_back

Version 7.0.0

◇ New character types:

- chem_bio_pedestrian
- fdc_chock_and_chains
- fdc_fuelman
- fdc_grounding
- fdc_pallet
- fdc_pedestrian
- firefighter
- fireman_silver
- vehicle_technical
- vehicle_turret
- vehicle_wheels

Version 6.1.2

- ◇ Vehicles and props now have more accurate bounding volumes.
- ◇ Fixed problem with config file appearance syntax. The item "shaperset:shape" format caused a crash; it now works again.
- ◇ Replaced default scene object (*setsdefault_stagedefault_stage.flt*) with new default (*setsdefault_stage7flt/default_stage7.flt*). The new *default_stage7.flt* scene object has a number of different areas in which example scenarios can be placed.

Version 6.1.1

- ◇ Added new appearance "invisible_human"; Using this appearance for a human character will remove all geometry from the character, effectively making it invisible.

Version 6.1.0

- ◇ Added ability to specify in config files how long a null transition should take, and which blend function it should use.

Version 6.0.2

DI-Guy SDK Version 12.5.2 Index

- ◇ Modified stroll motions of female_pedestrian character to move hands further away from body.
- ◇ Fixed some facing issues for soldier-type characters in transition motions between stand_aim and kneel_aim actions.

Version 6.0.1

- ◇ Added particle_puff.rgb to list of effects textures.
- ◇ Fixed the standing hip height of actor greg. It had been mistakenly changed to match that of actor bill. This caused problems for motion scaling for some actor / appearance combinations. (PR 3029)

Version 6.0.0

- ◇ Initial 6.0 DI-Guy Data release.

Version 5.3.5

- ◇ Removed one unused level of Motion LODs from data loaders config entries. The removed LOD 6 always had 0 variables in it, which wasn't useful for anything. This change requires use of DI-Guy Applications 5.3.5.

Version 5.3.2

- ◇ Changed format of Motion Texture (motex) entries in DI-Guy config files. The new format gives motion textures their own config entry keyword, separate from motion arcs. This allows DI-Guy to know exactly which motions are going to be used as motion textures. Custom motion textures using the old format need to be updated.
- ◇ Changed format of gesture entries in DI-Guy config files. Custom gestures using the old format need to be updated.

Version 5.3.1

- ◇ Fixed problem with airport_crowd character selecting wrong action for default moving action.
- ◇ Added sergeant2 character, which is similar to the sergeant character but better deals with transitions between actions.
- ◇ Added fped_mother2 character, which is similar to the fped_mother character but better deals with transitions between actions.

Version 5.2.2

- ◇ Running motion has been improved for the following characters: afghan_fighter, cbd, soldier, soldier2, soldier_precise, soldier_qk. Scenarios from previous revisions using these motions may behave slightly differently.

5.99 diguy_changes_dox.h

DI-Guy SDK Version 12.5.2

Contents:

\page changelog DI-Guy Changes and Version History

\tableofcontents

\section Version125Data DI-Guy Data History

For an overview of data changes please see release notes:

- ◆ [DI-Guy_11_feature_list.pdf](#)
- ◆ [DI-Guy_12_feature_list.pdf](#)
- ◆ [DI-Guy_125_feature_list.pdf](#)

\section Version125 DI-Guy Version 12.5

\subsection v125diguyAPI DI-Guy API

Note that most changes to the DI-Guy API also apply to DI-Guy Scenario.

\subsubsection v1251api Version 12.5.1

- ◆ Fixed bug where customer sourced dds files failed to load because we assumed they were lz4'ed.
- ◆ Fixed bug where files with capitalized extensions eg "FLT" failed to load.
- ◆ Fixed bug where invalid xrefs caused parent bounding boxes to fail to register as possibly valid.
- ◆ Fixed hard crash when drawing empty octtree.
- ◆ Fixed matrix state glitch when drawing octtree and scene object bounds
- ◆ Fixed path planning bug in funnel code when far above the z origin.
- ◆ Fixed bug where browsing for a new scene object file name didn't force octtree recreation.
- ◆ Fixed logic bug in check visibility code.
- ◆ Fixed bug where changing appearances could cause single frame muzzle flashes and loss of character skinning matrices.
- ◆ Fixed bug where flt files were unable to have their material edited by the character viewer
- ◆ Fixed bug where changing character type caused all action beads to collapse on a path.
- ◆ Added additional guide example showing differences between similar and dissimilar character types.
- ◆ Fixed bug in built-in OpenGL renderer where fixed function mode didn't properly bind textures. Fixes textureless segmented characters rendering in Motion Editor.
- ◆ Fixed joystick not properly loading in I-Guy mode.
- ◆ Fixed ray testing against scaled scene objects.
- ◆ Fixed off by one error in link combo in character viewer.

- ◆ Fixed bug in Lua debugger. "Step over" and "Step out" buttons were not working correctly
- ◆ General improvements to code that tests `get_enabled()` and `get_dead()` on a character, the code will now also test the character's `t_out` compared to current time
- ◆ Added `diguyAgentParams::set_influence_box_enabled()` (and `get`)
- ◆ Fixed bug with ground clamping on large scenarios.
- ◆ Fixed performance issue with receiving props over the network.
- ◆ Improved shadow pass rendering performance.
- ◆ Shader cleanup.
- ◆ Changed Graphics API shader samples to default point light `#define` to off.
- ◆ Fixed crash when executing gesture that wasn't made for a character.
- ◆ Fixed crash when loading degenerate `flt` file `xrefs`.
- ◆ Improved memory use when decompressing many `dds` files.
- ◆ `diguyCharacter::set_orientation()` now checks if the character has a vehicle controller and if so, it now allows the caller to hard-set the vehicle's orientation. Previously, only the vehicle controller determined this.
- ◆ `diguyCharacter::find_path()` now checks if the `character_type` contains the substring 'vehicle' and if so, the path produced will be wide enough to allow the character to pass, based upon character's bounding-box's width.
- ◆ Added a number of try/catch blocks to deal with running out of memory while loading extremely large scenarios.
- ◆ Added code/UI to allow DI-Guy Scenario to unload textures when not used for a long time. Off by default.
- ◆ Added a number of experimental `diguyApp` memory management functions:

```

◇ diguyApp::set_threshold_to_unload_textures()
◇ diguyApp::get_threshold_to_unload_textures()
◇ diguyApp::set_texture_unload_frequency()
◇ diguyApp::get_texture_unload_frequency()
◇ diguyApp::get_unload_unused_after_scenario_load()
◇ diguyApp::set_unload_unused_after_scenario_load()

```

\subsubsection v1250api Version 12.5.0

CRITICAL api changes

- ◆ C++ callback:

```
float AltitudeFunction( diguyScenario* scenario, float x, float y, float old_z);
```

```
float AltitudeFunction( diguyScenario* scenario, float x, float y, float old_z, int * valid);
```

- ◆ Graphics API

- ◇ `bdiGraphicsInitGraphicsAPI.use_4x3_shader_matrices` must be set to 0 if you aren't planning on upgrading shaders to use 4x3 matrices.
- ◇ As a performance optimization we are no longer transposing our internal matrix representation in C++ before sending to OpenGL. 4x4 matrix bind should get updated to

```
glUniformMatrix4fv(m_bind_location_ufrm_link_matrices,    // uniform var location
num_mats,          // number of matrices
!get_transposed_matrix_data(),    // whether to transpose matrix
mat_data);         // pointer to matrix data
```

New Features

- ◆ New adaptive guide allowing characters to fix dead reckoning errors with appropriate step and strafe motions. Now the default in the networking system as well as featured in the guide example.
- ◆ New texture variation system allowing for infinite variations on next-gen characters. See `diguyCharacter::set_use_texture_variations()`.
- ◆ Next-gen characters now have a number of multiple available head appearances.
- ◆ Motion engine should now be fully multi-threaded, thread control api is part of `diguyApp`.
- ◆ New C++ documentation system, utilizing doxygen.
- ◆ Fixed matrix update issues with motion lod system and skinned characters, motion lod system can be used again with them, same issues had been happening with segmented characters as well. We recommend using multi-threaded pipeline instead of motion lods at this point.
- ◆ New ground clamp option 'g' – approximation based on ground clamping vehicle bounds
- ◆ Scene objects can now be collada DAEs.
- ◆ Added support for bump maps on scene objects.
- ◆ New multi-threaded asynchronous texture loader, controllable via scenario UI and `diguyCharacter::set_texture_load_behavior()` and `diguySceneObject::set_texture_load_behavior()`
- ◆ Collision Infrastructure Improvements:
 - ◇ Improved performance of ray tests with scenes with many props.
 - ◇ Scene Object spatial partitioning system has been overhauled allowing scene objects to be repositioned without recreating the octtree.
 - ◇ Performance improvement for adding/deleting/transforming scene objects.
 - ◇ Octtree memory usage has been reduced by 10%.
 - ◇ Octtree is now available for end users who's objects are not flt files, see scene_object_mesh programming example.
- ◆ New Scene Object collision control API functions:
 - ◇ `diguyScenario::create_user_defined_scene_object()`
 - ◇ `diguySceneObject::add_vertex_to_user_defined()`
 - ◇ `diguySceneObject::add_triangle_to_user_defined()`
 - ◇ `diguySceneObject::add_vertex_array_to_user_defined();`
 - ◇ `diguySceneObject::add_triangle_array_to_user_defined();`
 - ◇ `diguySceneObject::add_triangle_to_user_defined();`
 - ◇ `diguySceneObject::build_user_defined();`
- ◆ New API functions:

```

◇ diguyCharacter::get_facefx_node_value()
◇ diguyCharacter::get_max_shader_quality_level()
◇ diguyCharacter::get_texture_load_behavior()
◇ diguyCharacter::get_use_texture_variations()
◇ diguyCharacter::set_max_shader_quality_level()
◇ diguyCharacter::set_texture_load_behavior()
◇ diguyCharacter::set_use_texture_variations()
◇ diguyImpact::get_impact_velocity(float* x, float* y,
    float* z)
◇ diguyImpact::get_is_from_net_detonation()
◇ diguyImpact::get_DIS_munition_septet_value(int
    septet_index)
◇ diguyImpact::get_DIS_site()
◇ diguyImpact::get_DIS_host()
◇ diguyImpact::get_DIS_event_number()
◇ diguyImpact::get_DIS_munition_number()
◇ diguyImpact::get_DIS_detonation_result()
◇ diguyImpact::get_DIS_burst_warhead_type()
◇ diguyImpact::get_DIS_burst_fuze()
◇ diguyImpact::get_DIS_burst_quantity()
◇ diguyImpact::get_DIS_burst_rate()
◇ diguyScenario::get_character_type_from_appearance();
◇ diguyScenario::get_num_character_types_with_appearance()
◇ diguyScenario::get_weapon_fire_effects_enabled()
◇ diguyScenario::set_default_intersection_function()
◇ diguyScenario::set_intersection_function()
◇ diguyScenario::set_particle_module_disabled()
◇ diguyScenario::set_weapon_fire_effects_enabled()
◇ diguySceneObject::get_max_shader_quality_level()
◇ diguySceneObject::get_texture_load_behavior()
◇ diguySceneObject::set_max_shader_quality_level()
◇ diguySceneObject::set_texture_load_behavior()
◇ diguy_ogl_apply_active_lights();

```

- ◆ Upgraded lua/C++ binding library. Should greatly improve the readability of error warnings when lua improperly calls C++.
- ◆ `diguyCharacter::get_ai_state_string()` signature has changed to have flags for optionally including callstack info.
- ◆ `diguyCharacter::is_within_distance_n_of_member_of_group()` factors in if a character is enabled.
- ◆ Fixes to the bead constraints system (and user interface) renamed beads should keep the script and decision constraints attached to them.
- ◆ Fixed bugs where script and decision beads attached to the end of paths and stop signs didn't always trigger, or triggered later than might be expected. This might introduce slight timing changes to existing scenarios.
- ◆ Characters are now sorted before drawing, and depending on the rendering pass drawn front to back or back to front.
- ◆ Fixed bug in `diguyCharacter::force_partial_path()` when a target action bead is specified.
- ◆ Fixed crash in `diguyScenario::preload_appearance()` when a non-existent appearance is specified.

- ◆ `diguyScenario::preload_appearance()` should now preload textures for appearances that are based on material swaps.
- ◆ References to aim algorithm 7 have been removed.
- ◆ Deprecated Functions Removed:

- ◇ `diguyApp::create_load_manager()` replaced with `diguyScenario::create_load_manager()`
- ◇ `diguyApp::get_load_manager()` replaced with `diguyScenario::get_character_load_manager()`
- ◇ `diguyApp::get_load_manager()` replaced with `diguyScenario::get_character_load_manager()` -> `set_enabled(0)`
- ◇ `diguyCharacter::set_max_impact_hits()` replaced with `diguyCharacter::set_weapon_max_impacts()`
- ◇ `diguyCharacter::get_paused()` replaced with `diguyCharacter::get_is_network_paused()`
- ◇ `diguyCharacter::get_visible()` replaced with `diguyCharacter::get_is_drawn()`
- ◇ `diguyCharacter::set_aim_param_desired_locate_time()`
- ◇ `diguyCharacter::get_aim_param_desired_locate_time()`
- ◇ `diguyCharacter::set_aim_param_seek_tolerance()`
- ◇ `diguyCharacter::get_aim_param_seek_tolerance()`
- ◇ `diguyCharacter::set_aim_interpolation_flag()`
- ◇ `diguyCharacter::get_aim_interpolation_flag()`
- ◇ `diguyCharacter::set_aim_smooth_flag()`
- ◇ `diguyCharacter::get_aim_smooth_flag()`
- ◇ `diguyCharacter::set_aim_divisions()`
- ◇ `diguyCharacter::set_aim_param_time_constant()` replaced with `set_aim_param_acquire_factor()`
- ◇ `diguyCharacter::get_aim_param_time_constant()` replaced with `get_aim_param_acquire_factor()`
- ◇ `diguyCharacter::set_aim_param_time_constant2()` replaced with `set_aim_param_new_aim_duration()`
- ◇ `diguyCharacter::get_aim_param_time_constant2()` replaced with `get_aim_param_new_aim_duration()`
- ◇ `diguyCharacter::set_aim_param_reset_time_constant()` replaced with `set_aim_param_new_aim_factor()`
- ◇ `diguyCharacter::get_aim_param_reset_time_constant()` replaced with `get_aim_param_new_aim_factor()`

- ◆ Deprecated Callbacks Removed:

- ◇ `diguyCrowd::CALLBACK_ID_NEW_FOCUS_CHARACTER` use `diguyCharacter::CALLBACK_ID_AGENT_NEW_FOCUS_CHARACTER`
- ◇ `diguyCrowd::CALLBACK_ID_ATTACKER_OUT_OF_TARGETS` use `diguyCharacter::CALLBACK_ID_AGENT_ATTACK_OUT_OF_TARGETS`
- ◇ `diguyCrowd::CALLBACK_ID_ATTACKER_NO_VISIBLE_TARGETS` use `diguyCharacter::CALLBACK_ID_AGENT_ATTACK_NO_VISIBLE_TARGETS`
- ◇ `diguyCrowd::CALLBACK_ID_PURSUED_CHARACTER_REACHED` use

DI-Guy SDK Version 12.5.2 Index

```
diguyCharacter::CALLBACK_ID_AGENT_PURSUE_TARGET_REACHED
◇ diguyCrowd::CALLBACK_ID_WANDER_TARGET_REACHED
use
diguyCharacter::CALLBACK_ID_AGENT_WANDER_TARGET_REACHED
```

- ◆ Fixed bug with T-in/T-out and review data on scene object characters.
- ◆ Updated perl header file examples with lua equivalents.
- ◆ Character viewer debug shader can now be used on scene objects
- ◆ Fixed bug on blinking characters when t-in was not 0.
- ◆ Fixed Expressive Faces pop when interacting with gaze code.
- ◆ Added the ability to pause the shadow map update and lower the update frequency.
- ◆ Added controls for adjusting the shadow map frustum and shadow acne fudge factor.
- ◆ Character culling is now based on the base link, not the position link.
- ◆ New DI-Guy Author APIs:

```
◇ diguyAuthorInterface
◇ diguyAuthorVisualMaterial
◇ diguyAuthorVisualMultiline
◇ diguyAuthorVisualObjectHandle
◇ diguyAuthorVisualObjectHandle
◇ diguyAuthorVisualRegionMesh
◇ diguyAuthorVisualRegionPaintbrush
◇ diguyAuthorVisualWaypoint
```

\subsection v125digs DI-Guy Scenario

\subsubsection v1250digs Version 12.5.0

- ◆ New lighting system triggered by particle effects and weapons fire.
- ◆ Improvements to 3d visuals for path editing.
- ◆ Merged some subwidgets in character page.
- ◆ Improved interactive Face Expression Editing.
- ◆ Improved determinism of fast forward, and setting current time via the VCR UI.
- ◆ Streamlined performance settings. Crowd Load Manager, Load Manager tabs have been phased out, settings are now located in the performance page.
- ◆ Changing camera settings will now force a motion update on all characters.
- ◆ Property tree UI's now have slider widgets, easier to change values interactively.
- ◆ Labels above character's heads now autoscale and draw z-sorted.
- ◆ Right click menu now contains a shortcut to turning on the debug label.
- ◆ Right click on log window selection can now go to package selected or Character selected.
- ◆ Shader UI includes the ability to edit character variation texture colors.
- ◆ Physx library has been rewritten to use Physx 3.2, this is upgraded from version 2.8, Physx installer should no longer be needed.
- ◆ Fixed a number of sound looping issues.

\subsection v125DIGuyAI DI-Guy AI

\subsubsection v1250AI Version 12.5.0

- ◆ Improved turning motion system, now on by default.

- ◆ Variant system now treats the normal variant as an indicator to automatically pick an appropriate variant for the character. Controlled by `diguyCharacter::agent_set_auto_variant_selection()`
- ◆ System will now cycle through appropriate motions randomly.
- ◆ LuaCharacter – `sleep_ignore_messages` avoids resuming the coroutine.
- ◆ AI movement callbacks are now queued up and triggered at the end of the multi-threaded update.
- ◆ New AI State Inspector – mini lua watch window, activate via right click menu.
- ◆ Added `diguyCrowdProfile::create_character()` short cut for using a profile to create a character.
- ◆ Significantly improved speed of collision influence calculation for crowds.
- ◆ Crowd profiles have been updated to use new variation system.
- ◆ New API functions:

```

◇ diguyAgentParams::set_turning_motions_enabled()
◇ diguyAgentParams::get_turning_motions_enabled()
◇ diguyCharacter::agent_set_auto_variant_selection()
◇ diguyCharacter::agent_get_auto_variant_selection()
◇ diguyCrowdProfile::create_character()

```

\subsection v125gapi Graphics API Changes

\subsubsection v1251gapi Version 12.5.1

- ◆ Added functionality to support color picking intersection testing for scene graph implementations.
- ◆ Added sample code demonstrating intersection testing with OpenSceneGraph.
- ◆ New API Functions:

```

◇ diguyCharacter::trigger_shader_refresh()
◇ diguyGraphicsShape::get_bind_color_array()

```

\subsubsection v1250gapi Version 12.5.0

- ◆ Modified `diguyOsgGraphicsMesh::build()` to load appearance textures on demand instead of pre-loading all appearance textures.
- ◆ Added support for 4x3 skinned matrices, saves 25% of the bandwidth and uniform constants sent per character draw call. Now the default for OpenGL. Configurable via `bdiGraphicsInitOgl.use_4x3_shader_matrices` and `bdiGraphicsInitGraphicsAPI.use_4x3_shader_matrices`. `diguy_graphics_get_osg_defaults()` and `diguy_graphics_get_dx9_defaults()` have been added to make it a bit easier to get the defaults set up.
- ◆ Added sample code demonstrating use of 4x3 matrices for transform data.
- ◆ Greatly expanded shader techniques API, `diguyGraphicsShaderTechnique` can now be subclassed and end users can create new shaders on the fly with `diguyGraphicsShaderTechnique::find_or_create_shader_variation()`. End users can override existing functionality and return alternatively compiled shaders depending on their own criteria, sample code is provided demonstrating conditionally compiling in point light support but only using shader when necessary.
- ◆ Fixed matrix transpose mistake in `diguyOglGraphicsLink.cpp`

- ◆ Cleaned up computation of local space camera.
- ◆ Removed use of display lists from diguyOglGraphicsMaterial.cpp
- ◆ Cleaned up diguyOglGraphicsShaderInstance.cpp to reflect the use of shader techniques.
- ◆ Added sample code demonstrating updating lighting state for simple point lights on characters.
- ◆ Added sample code demonstrating using #defines to control compilation of various shader features.
- ◆ Added support for variation system to sample code, this is shown in diguyOglGraphicsShape and diguyOsgGraphicsShape.
- ◆ Added diguyOglGraphicsShaderTechnique demonstrating how to modify built-in shaders and overriding our existing shader LOD system.
- ◆ New API Functions:

```

◇ diguyGraphicsShaderProgram::get_num_additional_feature_names()
◇ diguyGraphicsShaderProgram::get_additional_feature_name()
◇ diguyGraphicsShaderProgram::has_additional_feature()
◇ diguyGraphicsShaderProgram::get_vertex_shader_filename()
◇ diguyGraphicsShaderProgram::get_pixel_shader_filename()
◇ diguyGraphicsShaderProgram::get_transposed_matrix_data()
◇ diguyGraphicsShaderProgram::load()
◇ diguyGraphicsShaderProgram::get_use_4x3_shader_matrix_data()
◇ diguyGraphicsShaderTechnique::get_shader_by_distance()
◇ diguyGraphicsShaderTechnique::get_program_index_by_distance()
◇ diguyGraphicsShaderTechnique::get_shadow_pass_shader()
◇ diguyGraphicsShaderTechnique::find_or_create_shader_variation()
◇ diguyGraphicsShaderTechnique::pick_shader_program()
◇ diguyGraphicsShaderTechnique::build();
◇ diguyGraphicsShaderTechnique::add_additional_shader()
◇ diguyGraphicsShaderTechnique::get_num_additional_shaders()
◇ diguyGraphicsShaderTechnique::get_additional_shader_at_index()
◇ diguyGraphicsShape::get_variation_texture()
◇ diguyGraphicsShape::get_variation_texture_size()
◇ diguyGraphicsShape::update_variation_texture()
◇ Init params:
    bdiGraphicsInitGraphicsAPI.transpose_shader_matrices
◇ Init params:
    bdiGraphicsInitGraphicsAPI.use_4x3_shader_matrices

```

\subsection v12diguynet DI-Guy Networking

\subsubsection v125 Version 12.5

- ◆ New API Functions:

```

◇ diguyNetInterface::set_use_adaptive_guide()
◇ diguyNetInterface::get_use_adaptive_guide()

```

- ◆ New Custom PDU's (Note: Functionality currently limited to DIS.)

DI-Guy SDK Version 12.5.2 Index

- ◇ `DIGUY_CUSTOM_PDU_ID_SET_SHAPE_VARIATION` replicates variation texture over the network
- ◇ `DIGUY_CUSTOM_PDU_ID_EXECUTE_GESTURE` replicates the `diguyCharacter::execute_gesture()` call over the network
- ◇ `DIGUY_CUSTOM_PDU_ID_CHARACTER_SCALE` replicates the `diguyCharacter::set_scale()` value over the network.
- ◆ `DIGUY_CUSTOM_PDU_ID_SET_CHAR_APPEARANCE` Now contains a second optional string for head appearances

\section Version120 DI-Guy Version 12.0

\subsection v12digs DI-Guy Scenario

\subsubsection v1201 Version 12.0.1

- ◆ Added visualization option for gaze vectors.
- ◆ Restored missing default face expressions and chain settings libraries to new.dss.
- ◆ Added display of more callback information to the Event Mapper User Interface.
- ◆ Fixed bugs with water shader.

\subsection v12diguyAPI DI-Guy API

Note that most changes to the DI-Guy API also apply to DI-Guy Scenario.

\subsubsection v1203 Version 12.0.3

- ◆ Updated network model map for vehicles and also resolved issue with vehicles stuttering on paths.
- ◆ Made fire and detonation PDUs use matching event ids. Added velocities to Fire PDUs. M16 velocity is used by default for all weapons but can be overridden in munition config file.
- ◆ Fixed DI-Guy Scenario bugs in copy/paste paths, action spreadsheet.
- ◆ Fixed minor content bugs involving missing torso on insurgent.

\subsubsection v1202 Version 12.0.2

- ◆ Fixed `diguyApp::reload_shader_programs()` to support the graphics API. See `diguyOglGraphicsShaderProgram.cpp` for minor change: shaders that are already loaded must be detached on reload.
- ◆ Added `diguyCharacter::set_shader_lod()` and `diguyCharacter::get_shader_lod()`.
- ◆ Modified `diguyCharacter::set_face_expression()` to support a strength multiplier.
- ◆ Modified `diguyOsgGraphicsMesh::build()` to load appearance textures on demand instead of pre-loading all appearance textures.
- ◆ Call callbacks with id `CALLBACK_ID_SCENE_OBJECT_IMPACT` if `diguyScenario::trigger_detonation()` or `diguyCharacter::trigger_detonation_on_character()` are called.
- ◆ Values in `diguy_dis_constants.h` now available in scripting.

- ◆ Added a number of DIS detonation data values to `diguyImpact`; e.g., `diguyImpact::get_DIS_detonation_result()`.

\subsubsection v1201 Version 12.0.1

- ◆ AI flee code now properly uses feeler callback to pick the longest available flee route when near buildings.
- ◆ Fleshed out FaceFX api, fixed a number of after action review issues, added the following functions:

```

◇ diguyCharacter::stop_facefx_animations()
◇ diguyCharacter::get_num_facefx_nodes();
◇ diguyCharacter::get_facefx_node_name();
◇ diguyCharacter::find_facefx_node_index();
◇ diguyCharacter::set_facefx_node_value();

```

- ◆ Updates to FaceFX sample content.
- ◆ Added the following functions to help enable advanced shader materials in the graphics api:

```

◇ diguyOglGraphicsShaderProgram::bind_uniform_with_location( )
◇ diguyOglGraphicsShaderProgram::bind_uniform_with_location( )
◇ diguyOglGraphicsShaderProgram::get_attrib_bind_location( )
◇ diguyOglGraphicsShaderProgram::get_uniform_bind_location( )

```

- ◆ Streamlined shader bind and unbind calls.
- ◆ Fixed issue where even with only 1 update thread was used some of the multi-threaded infrastructure was active.
- ◆ Added warnings to various `add_callback_script` functions to warn if the same callback is repeatedly added.
- ◆ Update Middle Eastern AI's to avoid perpetual memory growth.
- ◆ Changed multi-threading to only use the Qt threading framework when more than one thread is allocated to DI-Guy.

\subsubsection v1200 Version 12.0.0

- ◆ In API programs shadow disks are off by default. Now the "set_draw_character_shadow_disks" setting in .dss files will be ignored. To turn shadow disks on or off an explicit call to `diguyScenario::set_draw_character_shadow_disks()` needs to be made.
- ◆ Fixed `diguyCharacter::agent_set_current_params_from_profile()` so arguments are in proper order. **Note:** If you have existing calls to `agent_set_initial_params_from_profile()`, verify that you are getting proper behavior.
- ◆ Added the following functions:

```

◇ diguyAgentParams::set_wander_max_wander_dist()
◇ diguyAgentParams::set_wander_sets_desired_orientation()

```

DI-Guy SDK Version 12.5.2 Index

- ◆ Removed the following experimental functions; should use minds instead:

- ◇ diguyCharacter::add_state_machine_instance()
- ◇ diguyCharacter::remove_state_machine_instance()
- ◇ diguyCharacter::remove_all_state_machine_instances()
- ◇ diguyCharacter::find_state_machine_instance()
- ◇ diguyScenario::get_num_state_machines()
- ◇ diguyScenario::create_state_machine()
- ◇ diguyScenario::destroy_state_machine()
- ◇ diguyScenario::get_state_machine_at_index()
- ◇ diguyScenario::find_state_machine()
- ◇ diguyScenario::get_num_state_machine_instances()
- ◇ diguyScenario::add_state_machine_instance()
- ◇ diguyScenario::remove_state_machine_instance()
- ◇ diguyScenario::get_this_state_machine_instance()
- ◇ diguyCrowd::all_members_add_state_machine_instance()
- ◇ diguyCrowd::all_members_remove_state_machine_instance()

- ◆ Removed the following deprecated functions:

- ◇ diguyApp::set_student_log_callback()
- ◇ diguyApp::student_log_print()
- ◇ diguyApp::set_user_query_yes_no_callback()
- ◇ diguyApp::user_query_yes_no()
- ◇ diguyApp::set_user_query_text_callback()
- ◇ diguyApp::set_user_query_item_callback()
- ◇ diguyApp::user_query_item()
- ◇ diguyApp::set_user_inform_callback()
- ◇ diguyApp::user_inform()

\subsection v12DIGuyAI DI-Guy AI

\subsection v12gapi Graphics API Changes

- ◆ See the 11–12 transition guide in the SDK user manual for a complete overview of the changes between 11 and 12

\section v11diguy DI-Guy Version 11

\subsection v11diguyapi DI-Guy API

Note that most changes to the DI-Guy API also apply to DI-Guy Scenario.

\subsubsection v1160 Version 11.6.0

- ◆ Added [diguyRegionPaintbrushMode](#) enum to diguy_constants.h.
- ◆ Fixed problem with agent_set_current_params_from_profile() use in decision clauses.
- ◆ Made the following functions callable from decisions:

- ◇ [diguyCharacter::agent_move_on\(\)](#)

- ◇ `diguyCharacter::agent_move_to_region()`
- ◇ `diguyCharacter::agent_path_follow()`
- ◇ `diguyCharacter::agent_resume_path_follow()`
- ◇ `diguyCharacter::agent_set_current_focus_character()`
- ◇ `diguyCharacter::agent_set_current_focus_group()`

- ◆ Added the following functions:

- ◇ `diguyApp::set_region_paintbrush_mode()`
- ◇ `diguyApp::get_region_paintbrush_mode()`
- ◇ `diguyApp::set_munition_type_DIS_impact_septet()`
- ◇ `diguyApp::set_munition_type_DIS_tracer_septet()`
- ◇ `diguyCharacter::trigger_detonation_on_character()`
- ◇ `diguyCharacter::get_vehicle_body_orientation()`
- ◇ `diguyScenario::trigger_decision_delayed()`
- ◇ `diguyScenario::trigger_script_delayed()`
- ◇ `diguyScenario::get_altitude_from_point()`
- ◇ `diguyScenario::destroy_region()`
- ◇ `diguyScenario::save()`
- ◇ `diguyScenario::save_as()`
- ◇ `diguyScenario::get_filename()`
- ◇ `diguyScenario::get_filename_without_directory()`
- ◇ `diguyScenario::get_unsaved()`
- ◇ `diguyScenario::set_unsaved(int unsaved)`
- ◇ `diguyRegion::get_random_factor()`

- ◆ Remove unused playback mode enum
`DIGUY_SCENARIO_PLAYBACK_MODE_RECORD`.
- ◆ Made changes to some scenario call operations. Previously the `diguyScenario::post_draw()` call had no effect; it now allows various post draw type operations to be optionally invoked. Previously these operations were automatically done at the end of `diguyScenario::draw_pass2()`.
- ◆ Added the following functions:

- ◇ `diguyScenario::set_overall_history_type()`
- ◇ `diguyScenario::get_overall_history_type()`
- ◇ `diguyScenario::manually_invoke_callbacks_now()`
- ◇ `diguyScenario::set_current_region()`
- ◇ `diguyScenario::get_current_region()`

\subsection gapi Graphics API Changes

\subsubsection v1160 Version 11.6.0

- ◆ Moved the following functions from `diguyGraphicsShaderInstance` to `diguyGraphicsShaderProgram`:

- ◇ `diguyGraphicsShaderInstance::bind_link_matrices()`
- ◇ `diguyGraphicsShaderInstance::bind_color_array()`

DI-Guy SDK Version 12.5.2 Index

Your Graphics API subclasses will need to be updated. `bind_link_matrices()` will now potentially be called multiple times per character depending on its complexity. For example, animated faces from `facefx` will cause multiple matrix binds.

The signature for `diguyGraphicsTexture::bind_now()` has changed to `diguyGraphicsTexture::bind_now(diguyTextureMapType texture_type)` to support multi-texturing. Your Graphics API subclasses need to be updated.

\subsubsection v110x Version 11.0.x

- ◆ Added ability to hear sounds when history is enabled and time is rewound to a point midway in the sound. Note that the function `diguyScenario::set_history_max_sound_instances()` lets users set how many sound instances to be remembered; default is 100.
- ◆ Updated where DI-Guy looks for its license file. It is looked for in this order:
 - ◇ file specified by the environment variable `DIGUY_LICENSE_FILE`
 - ◇ file specified by the environment variable `BDILM_LICENSE_FILE`
 - ◇ file specified by the environment variable `LM_LICENSE_FILE`
 - ◇ `license.dat` in DI-Guy installation `flexlm` subdir
- ◆ Turned off character shadow disks by default when using SDK; they're still on by default in DI-Guy Scenario.

\subsubsection v1104 Version 11.0.4

- ◆ Resolved issues with wheeled vehicles on paths, `force_action`, and `resume_path`
 - ◇ Problem with vehicles not resuming their path and remaining motionless after `force_action("still")`
 - ◇ Problem with vehicles resuming path and wheels not spinning
 - ◇ Problems with position discontinuities when `force_action("still")` is applied to a vehicle on a path

\subsubsection v1102 Version 11.0.2

- ◆ Updated where DI-Guy looks for its license file. It is looked for in this order:
 - ◇ file specified by the environment variable `DIGUY_LICENSE_FILE`
 - ◇ file specified by the environment variable `BDILM_LICENSE_FILE`
 - ◇ file specified by the environment variable `LM_LICENSE_FILE`
 - ◇ `license.dat` in DI-Guy installation `flexlm` subdir
- ◆ Turned off character shadow disks by default when using SDK; they're still on by default in DI-Guy Scenario.
- ◆ Removed use of `glCallList` in materials for the Graphics API sample code and the internal OpenGL renderer. Call lists are deprecated and have more overhead than just calling material functions directly.
- ◆ Added early out sample code for materials to graphics API; end users should now have the tools to skip unnecessary material binds.

DI-Guy SDK Version 12.5.2 Index

- ◆ Added a new geode/mesh merge as part of the bdg creation; should cut number of meshes and draw calls in half for most of the non- skinned characters.
- ◆ Added new draw stats profiler to improve feedback on pipeline bottlenecks. Profiler now available in Character Viewer. (Press Ctrl-P a few times.)
- ◆ Added linear draw pipeline init argument.
- ◆ Updated Benchmark sample program with new options and statistics.
- ◆ Fixed bug where get link positions would not take character scale into effect.
- ◆ Added the following functions to DI-Guy Graphics API:

```
◇ diguyGraphicsLink::get_link_transformation_matrix()  
◇ diguyGraphicsLink::begin_character_draw()  
◇ diguyGraphicsLink::end_character_draw()  
◇ diguyGraphicsLink::get_use_linear_traversal_draw()  
◇ diguyGraphicsLink::set_use_linear_traversal_draw()  
◇ diguyGraphicsTexture::get_last_bound_texture()  
◇ diguyGraphicsTexture::clear_last_bound_texture()
```

- ◆ Added sample code to the OpenGL graphics API reference implementation demonstrating the use of the link local transformation matrix to simplify the number of OpenGL function made.
- ◆ Added sample code to the OpenGL graphics API reference implementation to demonstrate using `diguyGraphicsTexture::get_last_bound_texture()` to minimize state changes.
- ◆ Added the following functions to DI-Guy OpenGL rendering API:

```
◇ diguy_ogl_get_num_frame_stats()  
◇ diguy_ogl_frame_stat_name()  
◇ diguy_ogl_get_frame_stat();  
◇ diguy_ogl_clear_all_frame_stats()
```

\subsubsection v1101 Version 11.0.1

- ◆ Added the following functions to DI-Guy Graphics API:

```
◇ diguyGraphicsMaterial::get_last_bound_material()  
◇ diguyGraphicsMaterial::clear_last_bound_material()
```

- ◆ Added sample code to the OpenGL Graphics API reference implementation to demonstrate using `diguyGraphicsMaterial::get_last_bound_material()` to minimize state changes.
- ◆ Fixed binary geometry serialization bug on x64 machines.
- ◆ Added `diguyScenario::is_valid_character_group_pointer()`

\subsubsection v1100 Version 11.0.0

- ◆ Removed deprecated functions:

```
◇ diguyChainSimulation::set_connection_character()
```

◆ New APIs:

- ◇ [diguyFormation](#)
- ◇ [diguyVehicleController](#)
- ◇ [diguyViewButtonPanel](#) (DI-Guy Scenario only)

\subsubsection v1051 Version 10.5.1

◆ Removed deprecated functions:

- ◇ [diguyScenario::set_tin\(\)](#)
- ◇ [diguyScenario::get_tin\(\)](#)
- ◇ [diguyScenario::trigger_script_event\(\)](#)
- ◇ [diguyScenario::add_custom_appearance\(\)](#)
- ◇ [diguyScenario::add_custom_appearance_based_on_existing_appearance\(\)](#)
- ◇ [diguyScenario::add_named_callback\(\)](#)
- ◇ [diguyScenario::remove_named_callback\(\)](#)
- ◇ [diguyScenario::call_named_callback\(\)](#)
- ◇ [diguyScenario::set_iguy_character\(\)](#)

\subsubsection v1017 Version 10.1.7

◆ Added the following functions:

- ◇ [diguyCharacter::set_max_unprocessed_impacts\(\)](#)
- ◇ [diguyCharacter::get_max_unprocessed_impacts\(\)](#)
- ◇ [diguyScenario::set_max_unprocessed_impacts\(\)](#)
- ◇ [diguyScenario::get_max_unprocessed_impacts\(\)](#)
- ◇ [diguyCharacter::reset_shader_program_to_default\(\)](#)
- ◇ [diguySceneObject::reset_shader_program_to_default\(\)](#)

\subsubsection v1016 Version 10.1.6

◆ Added functions to [diguyViewCamera](#) and [diguyViewCameraSettings](#):

- ◇ [diguyViewCamera::set_speed_fov_degrees_per_wheel_click\(\)](#)
- ◇ [diguyViewCamera::get_speed_fov_degrees_per_wheel_click\(\)](#)
- ◇ [diguyViewCameraSettings::set_speed_fov_degrees_per_wheel_click\(\)](#)
- ◇ [diguyViewCameraSettings::get_speed_fov_degrees_per_wheel_click\(\)](#)

- ◆ Added flag to [diguyScenario](#) that explicitly enables or disables tout ("T Out") for the scenario, instead of using the arbitrary value of 60000 seconds to mean tout should have no effect.
- ◆ Added flag similar to above that enables or disables character tout.

- ◇ [diguyScenario::set_tout_enabled\(\)](#)
- ◇ [diguyScenario::get_tout_enabled\(\)](#)
- ◇ [diguyCharacter::set_tout_enabled\(\)](#)
- ◇ [diguyCharacter::get_tout_enabled\(\)](#)

◆ Added the following functions:

```

◇ diguyCharacter::get_muzzle_link()
◇ diguyCharacter::get_muzzle_offset()
◇ diguyCharacter::stop_appearance_effect()
◇ diguyCharacter::set_weapon_tracers_enabled()
◇ diguyCharacter::get_weapon_tracers_enabled()
◇ diguyCharacter::set_weapon_shell_eject_enabled()
◇ diguyCharacter::get_weapon_shell_eject_enabled()
◇ diguyCharacter::set_weapon_smoke_enabled()
◇ diguyCharacter::get_weapon_smoke_enabled()

```

\subsubsection v1014 Version 10.1.4

- ◆ Added
diguyCharacter::CALLBACK_ID_AGENT_MIND_CURRENT_STATE_CHANGED.
- ◆ Added the following functions for getting and setting mind data.

```

◇ diguyCharacter::get_mind_current_state_name()
◇ diguyCharacter::set_mind_field_as_string()
◇ diguyCharacter::set_mind_field_as_float()
◇ diguyCharacter::get_mind_field_as_float()
◇ diguyCharacter::set_mind_field_as_int()
◇ diguyCharacter::get_mind_field_as_int()

```

- ◆ Added the following functions for setting agent params, and other agent behavior.

```

◇ diguyCharacter::agent_set_initial_params_from_profile()
◇ diguyCrowd::set_initial_params_from_profile()
◇ diguyAgentParams::set_move_on_t_min()
◇ diguyAgentParams::get_move_on_t_min()
◇ diguyAgentParams::set_move_on_t_max()
◇ diguyAgentParams::get_move_on_t_max()
◇ diguyCharacter::agent_move_on()

```

\subsubsection v1013 Version 10.1.3

- ◆ Added the following blink functions to diguyCharacter.

```

◇ diguyCharacter::set_automatic_blink_duration_in()
◇ diguyCharacter::get_automatic_blink_duration_in()
◇ diguyCharacter::set_automatic_blink_duration_out()
◇ diguyCharacter::get_automatic_blink_duration_out()
◇ diguyCharacter::set_automatic_blink_target_in()
◇ diguyCharacter::get_automatic_blink_target_in()
◇ diguyCharacter::set_automatic_blink_target_out()
◇ diguyCharacter::get_automatic_blink_target_out()
◇ diguyCharacter::set_automatic_blinks_enabled()
◇ diguyCharacter::get_automatic_blinks_enabled()

```

- ◆ Added the following gesture functions to diguyCharacter.

```

◇ diguyCharacter::execute_table_gesture()

```

DI-Guy SDK Version 12.5.2 Index

- ◇ `diguyCharacter::set_executing_table_gesture_desired_action()`
- ◇ `diguyCharacter::end_executing_gesture()`
- ◇ `diguyCharacter::abort_executing_gesture()`

- ◆ Added the following misc functions to `diguyCharacter`.

- ◇ `diguyCharacter::get_gaze_is_active()`
- ◇ `diguyCharacter::get_lpoint_is_active()`

\subsubsection v1011 Version 10.1.1

- ◆ Added the following functions to `diguyCharacter`.

- ◇ `diguyCharacter::get_is_aiming()`

\subsubsection v1010 Version 10.1.0

- ◆ Added the following functions for improved LOD handling.

- ◇ `diguyScenario::set_scale_graphics_lod_ranges_from_view_settings`
- ◇ `diguyScenario::get_scale_graphics_lod_ranges_from_view_settings`
- ◇ `diguyLoadManager::set_scale_zone_extents_from_view_settings()`
- ◇ `diguyLoadManager::get_scale_zone_extents_from_view_settings()`

- ◆ Added the following functions to `diguyCharacter`.

- ◇ `diguyCharacter::set_graphics_ptr()`
- ◇ `diguyCharacter::get_num_pose_overrides()`
- ◇ `diguyCharacter::get_pose_override_at_index()`

\subsection v11digs DI-Guy Scenario

\subsubsection v1151 Version 11.5.1

- ◆ Fixed intersection test issue with scene objects
- ◆ Fixed bogus fog changed notification
- ◆ Improved sound history recording and playback

\subsubsection v1101 Version 11.0.1

- ◆ Fixed crash with using `qt_invoke` without `lqt` loaded.

\subsection v11ME DI-Guy Motion Editor

\subsubsection v1101ME Version 11.0.1

- ◆ Fixed Motion Arc extended info handling.
- ◆ Fixed issue with the poser not being available on some skinned character links.

\subsubsection v1100 Version 11.0.0

DI-Guy SDK Version 12.5.2 Index

- ◆ Initial support for handling skinned characters.

\subsection v11diguycv DI-Guy Character Viewer

\subsubsection v11 Version 11

- ◆ Added Shadow and normal previewer.
- ◆ Added LOD picker.

\subsection v11vpdiguy DI-Guy for Vega Prime

\subsubsection v1151 Version 11.5.1

- ◆ Initial support for vegaPrime 5.x.
- ◆ fixed crash on exit using some modules.
- ◆ Changed setScriptPerlEnable default from true to false.

\subsubsection v1101 Version 11.0.1

- ◆ Initial support for vegaPrime 4.x.

\subsection v11diguynet DI-Guy Networking

\subsubsection v1101 Version 11.0.1

- ◆ Fixed a number of bugs with transmitting gazing and aiming behavior over network.
- ◆ Improvements to log messages.

\section v10apps DI-Guy Applications Version 10

\subsection v10api DI-Guy API 10

Note that most changes to the DI-Guy API also apply to DI-Guy Scenario.

\subsubsection v10010 Version 10.0.10

- ◆ Impact processing on characters now considers mind wakeup callbacks when deciding how the impact affects the character, in addition to the straight callbacks that are considered now.
- ◆ Added functions:

- ◇ `diguyAgentParams::set_vehicle_base_influence_box()`
 - ◇ `diguyAgentParams::get_vehicle_base_influence_box()`

- ◆ Added the following `diguyCharacter` callbacks that report when an agent is being push around by other characters of vehicles.

- ◇ `diguyCharacter::CALLBACK_ID_AGENT_NON_ZERO_REPULSION`
 - ◇ `diguyCharacter::CALLBACK_ID_AGENT_ZERO_REPULSION`

\subsubsection v1008 Version 10.0.8

DI-Guy SDK Version 12.5.2 Index

- ◆ Added merging of regions during scenario merge.
- ◆ Fixed merging of crowds to include merging of crowd behavior regions.
- ◆ Added `diguyScenario::CALLBACK_ID_PLAYBACK_MODE_CHANGED`.
- ◆ Added `diguy_set_diguy_base_dir_path()` function.
- ◆ Added `remove_on_scenario_load` argument to most `add_default_*_callback()` functions; allows for "permanent" callbacks to be added that persist through scenario loads. The default is

1. meaning remove the callback when a scenario is loaded.

\subsubsection v1006 Version 10.0.6

- ◆ Fixed visual issues of skinned appearances when the character is scaled or is using a motion LOD greater than 1.
- ◆ Fixed issue where default scenario callbacks were not being called.
- ◆ Added functions:

```
◇ diguyCharacter::set_graphics_api_node_ptr()  
◇ diguyCharacter::get_graphics_api_node_ptr()  
◇ diguyScenario::set_graphics_api_node_ptr()  
◇ diguyScenario::get_graphics_api_node_ptr()
```

\subsubsection v1005 Version 10.0.5

- ◆ Fixed hard crashes with on-screen labels outside of the ogl pipeline.
- ◆ Fixed issues with octree creation and visibility checks.
- ◆ Fix for vc8 mdd examples using lua.
- ◆ Added missing header file for DI-Guy Bench Marking example.
- ◆ Added infrastructure for hooking up callbacks so visibility tests can work in the SDK.
See the following functions:

```
◇ diguyScenario::set_default_point_los_function()  
◇ diguyScenario::set_default_character_los_function()  
◇ diguyCharacter::set_point_los_function()  
◇ diguyCharacter::get_point_los_function()  
◇ diguyCharacter::set_character_los_function()  
◇ diguyCharacter::get_character_los_function()
```

\subsubsection v1004 Version 10.0.4

- ◆ Fixed bug where green boxes were appearing at the origin in some DI-Guy Graphics API examples.
- ◆ Characters with skinned appearances specified will switch to their default unskinned appearance on platforms without skinning support.
- ◆ Added `diguyScenario::get_altitude_function()` and `diguyScenario::set_altitude_function()` mainly needed for force local path via the API.

- ◆ Added

```
bdiGraphicsInitGraphicsAPI::characters_have_world_space_triangle_queries  
and  
bdiGraphicsInitGraphicsAPI::scene_objects_have_world_space_triangle_queries
```

function allow octtree to be built from internally loaded scene objects. **Note:** this functionality requires that `diguyGraphicsMesh::free_loader_memory()` not be called.

\subsubsection v1003 Version 10.0.3

- ◆ Changed `diguyScenario::set_default_lod_ranges()` to accept "all" for the character type. Passing "all" will set the default LOD ranges for all character types, not just one specific one.
- ◆ Added the following functions to DI-Guy Graphics API for more flexibility in handling graphics LODs.

```

◇ diguyGraphicsShape::set_graphics_lod_ranges()
◇ diguyGraphicsShape::set_graphics_lod()
◇ diguyGraphicsShape::get_graphics_lod()

```

\subsubsection v1001 Version 10.0.1

- ◆ Fixed bugs with intersecting characters added to the octtree.
- ◆ Fixed bugs where custom directory env variable was being ignored when looking up lua scripts.
- ◆ Added functions `diguyCharacter::create_complete_pose_override()` and `diguyCharacter::destroy_complete_pose_override()`.
- ◆ Fixed bug where OGL skinned characters were missing some polygons; please see graphics API samples if you are seeing this bug. Your VBO class may need slight changes.

\subsubsection v1000 Version 10.0.0

- ◆ Added support for character "type maps". Type maps add extra data to characters that make it easier to decide which character type and appearance best match the desired role for characters to be added to the scenario. See the class `diguyCharacterTypeMap`, and information about the "Appearance Wizard" for DI-Guy Scenario.
- ◆ Added the following functions to `diguyCharacter` API:

```

◇ diguyCharacter::get_nearest_character_type_map()

```

- ◆ The DI-Guy Graphics API example directories `osg_12_examples` and `osg_22_examples` have been removed. They have been replaced with `osg_2_examples`.
- ◆ The `osg_2_examples` reference implementation now has support for skinned appearances. It uses the DI-Guy built-in geometry loader instead of OSG's, as DI-Guy has some specific requirements for loaded skinned geometry.
- ◆ `diguyScenario::add_callback()` and `diguyScenario::add_callback_script()` have been modified with an addition argument that specifies whether the callback should remain after a new scenario has been loaded. The previous behavior was for the callback to not be removed, which could cause problems. Users will now need to explicitly request that the callback remain.

\subsubsection v915 Version 9.1.5

- ◆ The ability to show bar graph information has been added to the `diguyViewLabel` class. See `diguyViewLabel::enable_bar_graph()`.
- ◆ Added the following functions to `diguyCharacter` API:

```
◇ diguyCharacter::set_weapon_munition_type_override()
◇ diguyCharacter::get_weapon_munition_type_override()
◇ diguyCharacter::get_weapon_muzzle_munition_type()
◇ diguyCharacter::get_default_muzzle_munition_type()
```

- ◆ Added the following functions to DI-Guy Graphics API:

```
◇ diguyGraphicsLink::post_build()
◇ diguyGraphicsLink::set_shader_instance()
◇ diguyGraphicsMaterial::get_is_unlit()
```

- ◆ Added new DI-Guy Graphics API classes: `diguyGraphicsShaderProgram` and `diguyGraphicsShaderInstance`.
- ◆ Added `time_to_live` to `network.cfg` file.
- ◆ Removed the following experimental functions:

```
◇ diguyScenario::load_shader()
◇ diguyScenario::reload_shader_programs()
```

- ◆ Added new DI-Guy functions to allow SDK users to intercept fire requests and plug in their own intersection test; see `diguyScenario::set_fire_weapon_intersection_function()` for more info.

\subsubsection v914 Version 9.1.4

- ◆ Added the following functions to `diguyCharacter` API:

```
◇ diguyCharacter::set_initial_path()
◇ diguyCharacter::get_initial_path()
```

- ◆ The way that weapon sounds are handled has been changed.

◇ The following functions have been deprecated:

```
· diguyCharacter::set_default_weapon_sound()
· diguyCharacter::get_default_weapon_sound()
· diguyCharacter::set_default_weapon_sound_gain()
· diguyCharacter::get_default_weapon_sound_gain()
```

◇ The above deprecated functions are replaced by:

```
· diguyCharacter::set_weapon_sound_override()
· diguyCharacter::get_weapon_sound_override()
· diguyCharacter::set_weapon_sound_gain_override()
```

· `diguyCharacter::get_weapon_sound_gain_override()`

- ◆ Added the following `diguyCharacter` callbacks to make it easier to tie events to editing characters in the UI.

- ◇ `diguyCharacter::CALLBACK_ID_UI_MOUSE_DOWN`
 - ◇ `diguyCharacter::CALLBACK_ID_UI_MOUSE_UP`

- ◆ Added `diguyViewPainter` class that allows the drawing of lines and

other primitives in DI-Guy Scenario 3D view windows. For access to this object call `diguyApp::get_painter()`.

- ◆ Added `diguyScenario::CALLBACK_ID_POST_DRAW`, allowing the scripting engine a chance to draw in the 3D view in DI-Guy Scenario. Characters with minds are also given the opportunity to draw; see the default `luaCharacter` script.

\subsubsection v912 Version 9.1.2

- ◆ Added `diguyImpact::get_impact_type()` and `diguyImpact::get_impact_type_string()` which provide information about how the impact was generated. New accessors for querying normal information are also available.

- ◆ Added new Lua Scripting module:

- ◇ The Lua module has all the functionality of the current Perl module.
 - ◇ Lua is popular in the game development industry for it's speed and efficiency. Lua script execution is much less performance intensive than a similar Perl script.
 - ◇ Lua is the basis for the new DI-Guy AI Mind framework.

- ◆ Scripts are no longer limited in length (was 32K).
- ◆ Deprecated functions:

- ◇ `diguyCharacter::get_base_link()` replaced with `diguyCharacter::get_position_link()`

- ◆ Added the following `diguyCharacter` callbacks to make it easier to tie events to editing characters in the UI.

- ◇ `diguyCharacter::CALLBACK_ID_UI_PRE_TRANSLATION`
 - ◇ `diguyCharacter::CALLBACK_ID_UI_POST_TRANSLATION`

\subsubsection v911 Version 9.1.1

- ◆ Added the following function to `diguyCharacter` API:

- ◇ `diguyCharacter::get_random_factor()`

- ◆ Added the following functions to `diguyCharacter` API for getting and setting character variables with a compact function name:

- ◇ `diguyCharacter::set_ivar()`
- ◇ `diguyCharacter::get_ivar()`
- ◇ `diguyCharacter::set_fvar()`
- ◇ `diguyCharacter::get_fvar()`
- ◇ `diguyCharacter::set_svar()`
- ◇ `diguyCharacter::get_svar()`

◆ Sound Module Improvements:

- ◇ Adding looping flag to
`diguyCharacter::play_3d_sound()` and
`diguyCharacter::play_sound()` allows sounds to repeat
until explicitly stopped with
`diguyCharacter::stop_sound()` or
`diguyCharacter::stop_all_sounds()`
- ◇ 3D Sounds should now Doppler shift depending on the velocity of
the emitter and listener.
- ◇ More a number of `diguySound` functions have been added to allow
querying a sounds properties.

◆ Fixed problem with the OpenFlight geometry loader. Light point records were
causing the loader to fail.

◆ Added two new `diguyCharacter` callback ids:

- ◇ `diguyCharacter::CALLBACK_ID_POST_CREATE_GEOMETRY`
- ◇ `diguyCharacter::CALLBACK_ID_PRE_DESTROY_GEOMETRY`

◆ Added new plugin function, `post_reset()`. Allows plugins to do reset operations at
the end of a scenario reset, instead of only at the beginning.

\subsubsection v910 Version 9.1.0

◆ Added the following functions to `diguyCharacter` API:

- ◇ `diguyCharacter::get_current_traveled_path_is()`
- ◇ `diguyCharacter::get_current_appearance_is()`
- ◇ `diguyCharacter::get_current_head_appearance_is()`
- ◇ `diguyCharacter::execute_table_gesture()`

\subsection v10digs DI-Guy Scenario 10

Note that most changes to the DI-Guy API also apply to DI-Guy Scenario.

\subsubsection v1002 Version 10.0.2

◆ Fixed the appearance of state functions in the code browser.

\subsubsection v1000 Version 10.0.0

◆ Added the "Appearance Wizard" to places where character type and appearance can
be selected. The Appearance Wizard provides an interactive UI for selecting desired

DI-Guy SDK Version 12.5.2 Index

attributes of the character to be created, such as category, age, sex, race, military branch, carried object, etc.

- ◆ Character types and appearances are now marked as "Recommended" and "Not Recommended". Many older character types and appearances have been nominally replaced with newer versions which should be preferred over the older ones. By default DI-Guy Scenario will show only the preferred ones, but can be changed to show all of them. Scenarios that currently use now not-recommended character types and appearances will still work as before.

\subsubsection v913 Version 9.1.3

- ◆ Improvements to autocomplete, including DIGUY_* constants as well as dynamic analysis of lua objects.

\subsubsection v912 Version 9.1.2

- ◆ Added new documentation viewer, accessible under the help menu.

\subsubsection v911 Version 9.1.1

- ◆ When selecting which agents' parameters are going to be edited, multi-select can now be used.
- ◆ Replaced Scenario Elements, Character Elements, View Settings, Event Handlers, and DI-Guy AI Elements windows with a single revamped Scenario Elements window that has collapsible tab groups.
- ◆ Removed experimental, incomplete "Merge Blitz" capability.

\subsubsection v910 Version 9.1.0

- ◆ Script editing now uses an editing component based on Scintilla, this provides syntax highlighting/coloring, line numbering, basic auto-complete, and code folding. Currently supported control keys:

- ◇ Ctrl '{' or Ctrl '}' – jump to matching brace
- ◇ Ctrl f – find dialog
- ◇ F3 – find again

- ◆ Added a number of ui improvements:

- ◇ Beads now display the time they occur instead of just an index.
- ◇ The Action Spreadsheet is available in the character elements page.
- ◇ A number of element types can now be sorted in their selector view at the top of the page. Most widgets can be sorted by index and name. Some add the ability to be sorted by time, appearance, character type, etc.
- ◇ The path page has a sorted list of all beads on a path.

\subsection v10diguyAI DI-Guy AI 10

\subsubsection v1004 Version 10.0.4

DI-Guy SDK Version 12.5.2 Index

- ◆ Added new AI_Populate SDK example; demonstrates populating the environment with lua minds based Agents via the SDK.

\subsubsection v1003 Version 10.0.3

- ◆ Improved behavior of paintbrush far from origin and at different heights.
- ◆ Added bright red log/package indicator on lua syntax errors.
- ◆ Improved behavior of code browser recompile button, pops up log window on syntax errors.

\subsubsection v1002 Version 10.0.2

– Fixed hard crash in path planner when pursuing with no behavior region set. – Fixed a number of bugs with agents colliding with props. – Fixed bug with crowd awareness code far from the origin.

\subsubsection v914 Version 9.1.4

- ◆ Significant improvements to A* path planning performance, 10x speed up in most cases.

\subsubsection v913 Version 9.1.3

- ◆ Improvements to path planning API, added the following functions to [diguyCharacter](#) API for navigating the world

```
◇ diguyCharacter::agent_move_to_point()  
◇ diguyCharacter::agent_move_to_point_via_subregions()  
◇ diguyCharacter::agent_move_to_region()  
◇ diguyCharacter::agent_move_to_region_via_subregions()
```

\subsubsection v912 Version 9.1.2

- ◆ New Lua based mind framework:

- ◇ New Lua Mind Editor in UI
- ◇ The crowd/people blitzers can now create characters with minds.

- ◆ Added the following functions to [diguyCharacter](#) API:

```
◇ diguyCharacter::agent_get_crowd()  
◇ diguyCharacter::agent_flee_crowd()
```

- ◆ Added [diguyCharacter::CALLBACK_ID_AGENT_VEHICLE_COLLISION](#); this callback is triggered on a vehicle when a vehicle hits an agent.
- ◆ Added [diguyRegion](#), an API wrapper for painted meshes. Retrievable with the function [diguyScenario::find_region\(\)](#).
- ◆ Vehicle bounds are now smoothed when the vehicle changes speed.
- ◆ Added simple gazing model into agent focus behavior.
- ◆ Added Idle Behavior, agent will stop moving and be responsive to focus and simple requests but won't spontaneously move.

- ◆ Added `diguyCharacter::agent_begin_behavior()` allows starting a behavior by name. Typically some of the more elaborate functions are preferred, but this can be useful for idling and such.
- ◆ Changes to None behavior. Agent will attempt to do less and clear out some transient state to make the behavior of the behavior more consistent.

\subsubsection v911 Version 9.1.1

- ◆ The following `diguyCrowd` callbacks are deprecated, and have been replaced with `diguyCharacter` equivalents.

- ◇ `diguyCrowd::CALLBACK_ID_NEW_FOCUS_CHARACTER`
- ◇ `diguyCrowd::CALLBACK_ID_ATTACKER_NO_VISIBLE_TARGETS`
- ◇ `diguyCrowd::CALLBACK_ID_ATTACKER_OUT_OF_TARGETS`
- ◇ `diguyCrowd::CALLBACK_ID_PURSUED_CHARACTER_REACHED`
- ◇ `diguyCrowd::CALLBACK_ID_WANDER_TARGET_REACHED`
- ◇ `diguyCrowd::agents_flee_crowd()`

- ◆ Added the following `diguyCharacter` callbacks to make it easier to detect important events in agent behavior.

- ◇ `diguyCharacter::CALLBACK_ID_AGENT_NEW_BEHAVIOR`
- ◇ `diguyCharacter::CALLBACK_ID_AGENT_NEW_FOCUS_CHARACTER`
- ◇ `diguyCharacter::CALLBACK_ID_AGENT_ATTACK_OUT_OF_TARGETS`
- ◇ `diguyCharacter::CALLBACK_ID_AGENT_ATTACK_NO_VISIBLE_TARGETS`
- ◇ `diguyCharacter::CALLBACK_ID_AGENT_PURSUE_TARGET_REACHED`
- ◇ `diguyCharacter::CALLBACK_ID_AGENT_PURSUE_TARGET_LOST`
- ◇ `diguyCharacter::CALLBACK_ID_AGENT_MINGLE_TARGET_REACHED`
- ◇ `diguyCharacter::CALLBACK_ID_AGENT_MINGLE_TARGET_LOST`
- ◇ `diguyCharacter::CALLBACK_ID_AGENT_WANDER_TARGET_REACHED`
- ◇ `diguyCharacter::CALLBACK_ID_AGENT_WANDER_TARGET_LOST`
- ◇ `diguyCharacter::CALLBACK_ID_AGENT_FLEE_AREA_LEFT`
- ◇ `diguyCharacter::CALLBACK_ID_AGENT_FLEE_AREA_ENTERED`
- ◇ `diguyCharacter::CALLBACK_ID_AGENT_TRAVEL_WAYPOINT_REACHED`
- ◇ `diguyCharacter::CALLBACK_ID_AGENT_TRAVEL_FORWARD_DEST_REACHED`
- ◇ `diguyCharacter::CALLBACK_ID_AGENT_TRAVEL_BACKWARD_DEST_REACHED`

- ◆ DI-Guy AI agents can now flee from multiple characters, groups, and locations instead of just one.
- ◆ Functions
`diguyCharacter::agent_set_current_params_from_profile()`
and `diguyCrowd::set_current_params_from_profile()` have a new argument, `retain_focus_objects`. Setting this to 1 allows the specified to be loaded without changing the agents' current focus character or group.

\subsection v10ME DI-Guy Motion Editor

- ◆ Added support for filtering the action table editor, eases editing of large action tables.
- ◆ Added code to reverse motions.
- ◆ Grabbing a pose on a Linear Offset should interpolate properly.
- ◆ Motion arc list and action table motion list are now in sync.

DI-Guy SDK Version 12.5.2 Index

\subsection v10digcv DI-Guy Character Viewer 10

\subsubsection v1000 Version 10.0.0

- ◆ Added "Appearance Wizard" button to make selecting an appropriate character type and appearance easier.

\subsection v10vpdiguy DI-Guy for Vega Prime 10

\subsubsection v1004 Version 10.0.4

- ◆ Initial release compiling with vp 3.0.

\subsubsection v912 Version 9.1.2

- ◆ Added support for turning on DI-Guy AI from vpDiguySettings UI. DI-Guy AI functionality is limited inside of Vega Prime
- ◆ Added support for turning on Lua scripting module from the the UI.
- ◆ Fixed long standing issue with character's having link offsets applied multiple times. (Also in 9.0.4)

\subsection v10diguynet DI-Guy Networking

\subsubsection v916 Version 9.1.6

- ◆ Fire PDUs position should now be the muzzle of the gun, the entity type should match the munition type fired, and the velocity should be the normalized direction of the muzzle.
- ◆ Fixed issue where agents on non-flat terrains were sending erroneous z values.

5.100 DI-Guy API, Constants and Enumerations

diguy_constants.h

DI-Guy SDK Version 12.5.2

Includes: declspec_bdiutil.h

Contents:

- ◆ [Version Numbers](#)
- ◆ [Misc. Constants](#)
- ◆ [General Enumerations](#)
- ◆ [DI-Guy Asynchronous Texture Load Enumerations](#)
- ◆ [DI-Guy AI Enumerations](#)
- ◆ [DI-Guy Physics Enumerations](#)
- ◆ [DI-Guy Author Enumerations](#)
- ◆ [DI-Guy Shader Enumerations](#)

The API is subject to change in future releases.

Version Numbers

5.100.1 constant DIGUY_VERSION_STRING

```
#define DIGUY_VERSION_STRING "12.5.2"
```

5.100.2 constants

```
#define DIGUY_VERSION_MAJOR 12
#define DIGUY_VERSION_MINOR 5
#define DIGUY_VERSION_POINT 2
```

5.100.3 constants

```
#define DIGUY_REQUIRED_DATA_VERSION_MAJOR 12
#define DIGUY_REQUIRED_DATA_VERSION_MINOR 5
#define DIGUY_REQUIRED_DATA_VERSION_POINT 0
```


Misc. Constants

5.100.4 constant DIGUY_PI

```
#define DIGUY_PI (3.14159265358979323f)
```

Description:

DI-Guy radians and degrees conversion factors.

5.100.5 constants

```
#define DIGUY_RAD2DEG (180.0f / DIGUY_PI)
#define DIGUY_DEG2RAD (DIGUY_PI / 180.0f)
```

5.100.6 constant DIGUY_DEFAULT_FLOAT

```
#define DIGUY_DEFAULT_FLOAT (-16384.0f)
```

Description:

This value is a magic number that, when passed to certain functions, means that the function should use or compute an appropriate default value for the argument.

5.100.7 constant DIGUY_DEFAULT_INT

```
#define DIGUY_DEFAULT_INT (-16384)
```

Description:

This value is a magic number that, when passed to certain functions, means that the function should use or compute an appropriate default value for the argument.

General Enumerations

5.100.8 enumeration diguyCallbackReturn

```
typedef enum
{
    DIGUY_CALLBACK_STOP = 0,
    DIGUY_CALLBACK_CONTINUE

} diguyCallbackReturn;
```

Description:

DI-Guy callbacks return a value of type [diguyCallbackReturn](#), which will be [DIGUY_CALLBACK_STOP](#) or [DIGUY_CALLBACK_CONTINUE](#). If the callback returns [DIGUY_CALLBACK_STOP](#), the default handler of the function will not be called; the callback is asserting that it has done everything necessary for the function call. If the callback returns [DIGUY_CALLBACK_CONTINUE](#), the default handler for the function will be called after the callback.

5.100.9 enumeration diguyCharacterClass

```
typedef enum
{
    DIGUY_CHARACTER_CLASS_UNKNOWN = 0,
    DIGUY_CHARACTER_CLASS_HUMAN,
    DIGUY_CHARACTER_CLASS_VEHICLE,
    DIGUY_CHARACTER_CLASS_OBJECT,
    DIGUY_CHARACTER_CLASS_EFFECT,
    DIGUY_CHARACTER_CLASS_ANIMAL,
    DIGUY_CHARACTER_CLASS_HUMAN_ON_VEHICLE,
    DIGUY_CHARACTER_CLASS_HUMAN_ON_OBJECT,
    DIGUY_CHARACTER_CLASS_HUMAN_ON_ANIMAL,
    DIGUY_CHARACTER_CLASS_OTHER,
    DIGUY_CHARACTER_CLASS_COUNT

} diguyCharacterClass;
```

Description:

DI-Guy character classes. This is a high-level description of the type of thing a character represents.

5.100.10 enumeration

```
enum diguyCharacterAppearanceTypes
{
    DIGUY_APPEARANCE_BODY = 0,
    DIGUY_APPEARANCE_HEAD,
    DIGUY_APPEARANCE_EXPRESSIVE_HEAD,
    DIGUY_APPEARANCE_HAND_ITEM,
    DIGUY_APPEARANCE_TYPES_COUNT
};
```

Description:

DI-Guy character appearance query API, this is for
[diguyCharacter::get_num_appearances_of_type\(\)](#)

5.100.11 enumeration diguyCharacterMode

```
typedef enum
{
    DIGUY_CHARACTER_MODE_PATH = 0,
    DIGUY_CHARACTER_MODE_FREE,
    DIGUY_CHARACTER_MODE_COUNT
} diguyCharacterMode;
```

Description:

This is an enumeration of the different modes a DI-Guy character can be in.
 See [diguyCharacter::get_action_mode\(\)](#),
[diguyCharacter::get_position_mode\(\)](#).

5.100.12 enumeration diguyParameterComparison

```
typedef enum
{
    DIGUY_PARAMETER_COMPARISON_NEAREST = 0,
    DIGUY_PARAMETER_COMPARISON_EQUAL,
    DIGUY_PARAMETER_COMPARISON_GREATER,
    DIGUY_PARAMETER_COMPARISON_LESS,
    DIGUY_PARAMETER_COMPARISON_COUNT

} diguyParameterComparison;
```

Description:

This is an enumeration of the different parameter comparison options that can be specified to various DI-Guy function calls, such as [diguyCharacter::get_action_from_description\(\)](#).

5.100.13 enumeration diguyMuzzle

```
typedef enum
{
    DIGUY_MUZZLE_DEFAULT = 0,
    DIGUY_MUZZLE_RIFLE,
    DIGUY_MUZZLE_GRENADE_LAUNCHER,
    DIGUY_MUZZLE_ACCESSORY1,
    DIGUY_MUZZLE_TURRET1,
    DIGUY_MUZZLE_COUNT

} diguyMuzzle;
```

Description:

This is an enumeration of the different muzzles that can be specified to various DI-Guy function calls, such as [diguyCharacter::set_default_weapon_muzzle\(\)](#) and [diguyCharacter::fire_weapon\(\)](#).

5.100.14 enumeration diguyImpactType

```
typedef enum
{
    DIGUY_IMPACT_DIRECT_HIT = 0,
    DIGUY_IMPACT_NEARBY_DETONATION,
    DIGUY_IMPACT_VEHICLE_COLLISION,
    DIGUY_IMPACT_NO_IMPACT,
    DIGUY_IMPACT_USER_GENERATED
} diguyImpactType;
```

Description:

This is an enumeration of the different impact types that are found in [diguyImpact](#).

5.100.15 enumeration diguyViewCameraMoveMode

```
typedef enum
{
    DIGUY_CAMERA_MOVE_FORWARD = 0,
    DIGUY_CAMERA_MOVE_SIDEWAYS,
    DIGUY_CAMERA_MOVE_VERTICAL
} diguyViewCameraMoveMode;
```

Description:

DI-Guy camera move modes.

In DI-Guy Scenario in Camera Input Mode, the movement mode of the camera determines what will happen when mouse buttons are pressed in the view window.

5.100.16 enumeration diguyViewCameraProjectionMode

```
typedef enum
{
    DIGUY_CAMERA_PROJECTION_PERSPECTIVE = 0,
    DIGUY_CAMERA_PROJECTION_PLANVIEW_XY,
    DIGUY_CAMERA_PROJECTION_PLANVIEW_XZ,
    DIGUY_CAMERA_PROJECTION_PLANVIEW_YZ
} diguyViewCameraProjectionMode;
```

Description:

DI-Guy camera projection modes.

5.100.17 enumeration diguyViewFogMode

```
typedef enum
{
    DIGUY_FOG_MODE_DISABLED = 0,
    DIGUY_FOG_MODE_EXP = 1,
    DIGUY_FOG_MODE_EXP2 = 2,
    DIGUY_FOG_MODE_LINEAR = 3,
    DIGUY_FOG_MODE_PIX_EXP = 4,
    DIGUY_FOG_MODE_PIX_EXP2 = 5,
    DIGUY_FOG_MODE_PIX_LINEAR = 6
} diguyViewFogMode;
```

Description:

DI-Guy fog modes.

5.100.18 enumeration diguyHistoryType

```
typedef enum
{
    DIGUY_HISTORY_TYPE_NONE = 0,
    DIGUY_HISTORY_TYPE_COMPLETE,
    DIGUY_HISTORY_TYPE_FIRST,
    DIGUY_HISTORY_TYPE_LAST,
    DIGUY_HISTORY_TYPE_PARTIAL

} diguyHistoryType;
```

Description:

DI-Guy history types.

5.100.19 enumeration

```
enum
{
    DIGUY_CLASSID_START = 0x400,

    DIGUY_CLASSID_CREATURE,
    DIGUY_CLASSID_LINK,
    DIGUY_CLASSID_SHAPE,

    DIGUY_CLASSID_CHARACTER,
    DIGUY_CLASSID_SCENE_OBJECT,
    DIGUY_CLASSID_PATH,
    DIGUY_CLASSID_PATH_SHAPE,
    DIGUY_CLASSID_SENSOR_REGION

} ;
```

Description:

DI-Guy renderer class ids.

5.100.20 enumeration

```
enum diguyRenderPassFlags
{
    DIGUY_RENDER_PASS_SKIP_PARTICLES           = 0x00000001,
    DIGUY_RENDER_PASS_USE_STORED_LOD_SETTINGS = 0x00000002,
    DIGUY_RENDER_PASS_REFLECTION               = 0x00000004,
    DIGUY_RENDER_PASS_SHADOW                   = 0x00000008,
    DIGUY_RENDER_PASS_INTERSECTION             = 0x00000010
};
```

Description:

DI-Guy renderer pass masks, set via `diguyApp::set_render_pass_flags()`.

DIGUY_RENDER_PASS_SKIP_PARTICLES

Particle rendering is skipped.

DIGUY_RENDER_PASS_USE_STORED_LOD_SETTINGS

LOD recalculation is skipped (for built-in OGL pipeline).

DIGUY_RENDER_PASS_REFLECTION

Internal use; water reflection objects are skipped.

DIGUY_RENDER_PASS_SHADOW

Shadow pass assumed; water, environmental effects are skipped.

DIGUY_RENDER_PASS_INTERSECTION

Pick pass assumed; link and shape colors are sent.

DI-Guy Asynchronous Texture Load Enumerations

Description:

This enumeration is used for `diguyCharacter::set_texture_load_behavior()` and `diguySceneObject::set_texture_load_behavior()`

DIGUY_LOAD_INSTANTLY

Textures are loaded instantly when the object is loaded, this is a blocking behavior and represents the pre 12.5 DI-Guy default behavior.

DIGUY_LOAD_PRIORITY_ASYNC

Textures are paged-in when they are drawn, closer diffuse textures are prioritized.

DIGUY_LOAD_ASYNC

Textures are paged-in in the order that they were loaded, all textures will eventually load.

5.100.21 enumeration `diguyGraphicsTextureLoadingBehavior`

```
typedef enum
{
    DIGUY_LOAD_INSTANTLY = 0,
    DIGUY_LOAD_PRIORITY_ASYNC = 1,
    DIGUY_LOAD_ASYNC = 2,
} diguyGraphicsTextureLoadingBehavior;
```

5.100.22 enumeration

```

enum
{
    DIGUY_ERROR                                = -1,

    DIGUY_ERROR_LICENSE_FAILURE                = -100,
    DIGUY_ERROR_LICENSE_SOFTWARE_INITIALIZATION_ERROR = -101,
    DIGUY_ERROR_LICENSE_NOT_FOUND              = -102,
    DIGUY_ERROR_LICENSE_EXPIRED                = -103,
    DIGUY_ERROR_LICENSE_VERSION_MISMATCH      = -104,
    DIGUY_ERROR_LICENSE_COUNT_TOO_LOW         = -105,
    DIGUY_ERROR_LICENSE_FILE_NOT_FOUND        = -106,
    DIGUY_ERROR_LICENSE_SERVER_ERROR          = -107,
    DIGUY_ERROR_LICENSE_NO_DONGLE             = -108,
    DIGUY_ERROR_LICENSE_NO_DONGLE_DRIVER      = -109,
    DIGUY_ERROR_LICENSE_UNKNOWN_ERROR         = -199,

    DIGUY_ERROR_GRAPHICS_SUBSYSTEM_INIT_FAILED = -200,
    DIGUY_ERROR_SIMULATION_SUBSYSTEM_INIT_FAILED = -201,
    DIGUY_ERROR_DATA_INSTALLATION_MISSING     = -202,
    DIGUY_ERROR_DATA_VERSION_TOO_LOW         = -203,
    DIGUY_ERROR_RENDERER_LICENSE_CHECK_FAILED = -204
};

```

Description:

DI-Guy error codes.

5.100.23 enumeration

```

enum
{
    DIGUY_VEC_X = 0,
    DIGUY_VEC_Y,
    DIGUY_VEC_Z,
    DIGUY_VEC_W,

    DIGUY_VEC_RZ = 0,
    DIGUY_VEC_RX,
    DIGUY_VEC_RY,

    DIGUY_VEC_QX = 0,
    DIGUY_VEC_QY,
    DIGUY_VEC_QZ,
    DIGUY_VEC_QW
};

```

Description:

DI-Guy vector indices.

5.100.24 enumeration diguyGraphicsJointType

```
typedef enum
{
    DIGUY_GRAPHICS_JOINT_TYPE_UNKNOWN = 0,
    DIGUY_GRAPHICS_JOINT_TYPE_FROZEN,
    DIGUY_GRAPHICS_JOINT_TYPE_PIN,
    DIGUY_GRAPHICS_JOINT_TYPE_SLIDER,
    DIGUY_GRAPHICS_JOINT_TYPE_CYLINDER,
    DIGUY_GRAPHICS_JOINT_TYPE_UNIVERSAL,
    DIGUY_GRAPHICS_JOINT_TYPE_PLANAR,
    DIGUY_GRAPHICS_JOINT_TYPE_GIMBAL,
    DIGUY_GRAPHICS_JOINT_TYPE_BALL,
    DIGUY_GRAPHICS_JOINT_TYPE_6DOF,
    DIGUY_GRAPHICS_JOINT_TYPE_TRANSLATION_AND_PIN,
    DIGUY_GRAPHICS_JOINT_TYPE_TRANSLATION_AND_GIMBAL,
    DIGUY_GRAPHICS_JOINT_TYPE_COUNT

} diguyGraphicsJointType;
```

Description:

DI-Guy graphics joint types.

5.100.25 enumeration

```
enum
{
    DIGUY_TYPE_UID_BASE_NOTHING = 0,
    DIGUY_TYPE_UID_BASE_CHARACTER,
    DIGUY_TYPE_UID_BASE_CHARACTER_LINK,
    DIGUY_TYPE_UID_BASE_CHARACTER_SHAPE,
    DIGUY_TYPE_UID_BASE_CHARACTER_PATH,
    DIGUY_TYPE_UID_BASE_CHARACTER_PATH_WAYPOINT,
    DIGUY_TYPE_UID_BASE_CHARACTER_BEAD,
    DIGUY_TYPE_UID_BASE_CROWD,
    DIGUY_TYPE_UID_BASE_REGION,
    DIGUY_TYPE_UID_BASE_LABEL,
    DIGUY_TYPE_UID_BASE_PATH_SHAPE,
    DIGUY_TYPE_UID_BASE_PATH_SHAPE_WAYPOINT,
    DIGUY_TYPE_UID_BASE_SCENE_OBJECT,
    DIGUY_TYPE_UID_BASE_SENSOR_REGION,
    DIGUY_TYPE_UID_BASE_OTHER,
    DIGUY_TYPE_UID_BASE_USER,
    DIGUY_TYPE_UID_BASE_COUNT

};
```

Description:

When a scene is rendered in order to select a DI-Guy object from, say, a mouse click, the type of object selected is identified by this enumeration.

5.100.26 enumeration diguyScenarioPlaybackMode

```
typedef enum
{
    DIGUY_SCENARIO_PLAYBACK_MODE_STOP = 0,
    DIGUY_SCENARIO_PLAYBACK_MODE_PLAY,
    DIGUY_SCENARIO_PLAYBACK_MODE_PLAY_REVERSE,
    DIGUY_SCENARIO_PLAYBACK_MODE_FF,
    DIGUY_SCENARIO_PLAYBACK_MODE_FF_REVERSE,
    DIGUY_SCENARIO_PLAYBACK_MODE_SS,
    DIGUY_SCENARIO_PLAYBACK_MODE_SS_REVERSE,
    DIGUY_SCENARIO_PLAYBACK_MODE_REWIND,
    DIGUY_SCENARIO_PLAYBACK_MODE_COUNT
} diguyScenarioPlaybackMode;
```

Description:

This enumeration identifies which playback mode time should follow. Only valid inside DI-Guy Scenario.

5.100.27 enumeration diguyVisibleFlag

```
typedef enum
{
    DIGUY_VISIBLE_NONE = 0,
    DIGUY_VISIBLE_CURRENT,
    DIGUY_VISIBLE_ALL,
    DIGUY_VISIBLE_PER_OBJECT

} diguyVisibleFlag;
```

Description:

This enumeration lists the visibility options for various objects in the DI-Guy Scenario environment.

5.100.28 enumeration diguyAuthorVisualUpdateFlag

```
typedef enum
{
    DIGUY_GRAPHICS_VISUAL_UPDATE_NONE          = 0,
    DIGUY_GRAPHICS_VISUAL_UPDATE_GEOMETRY      = 1,
    DIGUY_GRAPHICS_VISUAL_UPDATE_POSITION      = 2,
    DIGUY_GRAPHICS_VISUAL_UPDATE_MATERIAL      = 4,
    DIGUY_GRAPHICS_VISUAL_UPDATE_ALL           = 255

} diguyAuthorVisualUpdateFlag;
```

Description:

This enumeration lists which parts of a visual object may need to be updated. The values can be or-ed together.

DIGUY_GRAPHICS_VISUAL_UPDATE_NONE

Nothing in the visual needs to be updated.

DIGUY_GRAPHICS_VISUAL_UPDATE_GEOMETRY

The overall geometry of the visual needs to be updated. This means that vertices and other parts of polygons need to be recalculated.

DIGUY_GRAPHICS_VISUAL_UPDATE_POSITION

The position and/or orientation of the visual needs to be updated. The overall geometry of the visual (polygons, lines, etc.) remains unchanged.

DIGUY_GRAPHICS_VISUAL_UPDATE_MATERIAL

The material of the visual needs to be updated. This can include color, fill options, line width, etc. The geometry and position of the visual remain the same.

DIGUY_GRAPHICS_VISUAL_UPDATE_ALL

All aspects of the visual need to be updated.

5.100.29 enumeration diguyEulerAngleOrder

```
typedef enum
{
    DIGUY_EULER_ANGLE_ORDER_ZXY = 0,
    DIGUY_EULER_ANGLE_ORDER_ZYZ,
    DIGUY_EULER_ANGLE_ORDER_XYZ,
    DIGUY_EULER_ANGLE_ORDER_XZY,
    DIGUY_EULER_ANGLE_ORDER_YXZ,
    DIGUY_EULER_ANGLE_ORDER_YZX,
    DIGUY_EULER_ANGLE_ORDER_ZYX
} diguyEulerAngleOrder;
```

Description:

DI-Guy Euler angle orders.

5.100.30 enumeration diguyScenarioMergeFlag

```
typedef enum
{
    DIGUY_MERGE_FLAG_DONT_MERGE = 0,
    DIGUY_MERGE_FLAG_OVERWRITE_EXISTING,
    DIGUY_MERGE_FLAG_KEEP_BOTH,
    DIGUY_MERGE_FLAG_KEEP_EXISTING_ONLY,
    DIGUY_MERGE_FLAG_KEEP_EXISTING_ONLY_AND_WARN,
    DIGUY_MERGE_FLAG_RENAME_MERGED,
    DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN,
    DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL,
} diguyScenarioMergeFlag;
```

Description:

This enumeration is used for various variables in the [diguyScenarioMergeSettings](#) object. In general it tells the [diguyScenario::merge\(\)](#) function whether to load objects of a specified type at all, and if so, what to do if a merged object conflicts with an existing object.

DIGUY_MERGE_FLAG_DONT_MERGE

Objects in the merged scenario are not loaded.

DIGUY_MERGE_FLAG_OVERWRITE_EXISTING

Objects in the merged scenario are loaded.

In the case of a conflict, the exiting object is overwritten with the values of the merge object.

DIGUY_MERGE_FLAG_KEEP_BOTH

Objects in the merged scenario are loaded.

In the case of a conflict, both the exiting object and the merge object are kept, with no modifications. Note that this will leave some ambiguity of referencing objects by name. A warning will be printed.

Note that this value is here to allow for backward compatibility; it is not recommended for general use.

DIGUY_MERGE_FLAG_KEEP_EXISTING_ONLY

Objects in the merged scenario are loaded.

In the case of a conflict, the exiting object is kept as is and the merge object is discarded.

DIGUY_MERGE_FLAG_KEEP_EXISTING_ONLY_AND_WARN

Same as *DIGUY_MERGE_FLAG_KEEP_EXISTING*, but also prints a warning.

DIGUY_MERGE_FLAG_RENAME_MERGED

Objects in the merged scenario are loaded.

In the case of a conflict, both the exiting object and the merge object are kept, but the merged object is renamed.

DIGUY_MERGE_FLAG_RENAME_MERGED_AND_WARN

Same as *DIGUY_MERGE_FLAG_RENAME_MERGED*, but also prints a warning.

DIGUY_MERGE_FLAG_RENAME_MERGED_IF_NOT_EQUAL

Objects in the merged scenario are loaded.

In the case of a conflict, the existing object and merged object are checked for "equality". If they are the same, the existing object will be kept and the merge object will be discarded. If they are different, both objects will be kept, but the merge object will be renamed.

5.100.31 enumeration *diguyScenarioMergeInitialSettings*

```
typedef enum
{
    DIGUY_MERGE_INITIAL_SETTINGS_DEFAULT = 0,
    DIGUY_MERGE_INITIAL_SETTINGS_MERGE_EVERYTHING,
    DIGUY_MERGE_INITIAL_SETTINGS_MERGE_NOTHING,
    DIGUY_MERGE_INITIAL_SETTINGS_DIGUY6_COMPATIBLE,
    DIGUY_MERGE_INITIAL_SETTINGS_PASTE
} diguyScenarioMergeInitialSettings;
```

Description:

This enumeration tells DI-Guy what values to use as defaults when the *diguyScenario::create_merge_settings()* functions is called. See the documentation for that function for details on what the different values mean.

5.100.32 enumeration diguyScenarioMergeResetFlag

```
typedef enum
{
    DIGUY_MERGE_RESET_NONE = 0,
    DIGUY_MERGE_RESET_MERGED,
    DIGUY_MERGE_RESET_ALL

} diguyScenarioMergeResetFlag;
```

Description:

This enumeration tells DI-Guy which scenario elements should be reset after the merge is complete.

5.100.33 enumeration diguyPluginMouseCursor

```
typedef enum
{
    DIGUY_PLUGIN_MOUSE_CURSOR_NONE = 0,
    DIGUY_PLUGIN_MOUSE_CURSOR_LEFT_ARROW,
    DIGUY_PLUGIN_MOUSE_CURSOR_CROSSHAIR,
    DIGUY_PLUGIN_MOUSE_CURSOR_POINTING_HAND,
    DIGUY_PLUGIN_MOUSE_CURSOR_FORBIDDEN

} diguyPluginMouseCursor;
```

Description:

This enumeration lists the cursors returnable by the plugin function `get_cursor( )`.

DIGUY_PLUGIN_MOUSE_CURSOR_NONE

Plugin takes responsibility for drawing cursor, if any.

DIGUY_PLUGIN_MOUSE_CURSOR_LEFT_ARROW

The standard window-system left arrow cursor is shown.

DIGUY_PLUGIN_MOUSE_CURSOR_CROSSHAIR

Crosshair (plus-shaped) cursor is shown.

DIGUY_PLUGIN_MOUSE_CURSOR_POINTING_HAND

Pointing hand cursor is shown.

DIGUY_PLUGIN_MOUSE_CURSOR_FORBIDDEN

Action forbidden cursor is shown.

5.100.34 enumeration diguyChainCrossSection

```
typedef enum
{
    DIGUY_CHAIN_CROSS_SECTION_SQUARE = 0,
    DIGUY_CHAIN_CROSS_SECTION_PLUS,
    DIGUY_CHAIN_CROSS_SECTION_COIL
} diguyChainCrossSection;
```

Description:

This enumeration lists which cross-sections are available for DI-Guy chain simulation visuals.

DIGUY_CHAIN_CROSS_SECTION_SQUARE

The square cross-section is good for tube-like visuals, such as hoses. The normals of the square cross-section point radially outward from the center, giving the visual a rounded, tube-like appearance.

DIGUY_CHAIN_CROSS_SECTION_PLUS

The plus cross-section is good for thin, chain-like visuals.

DIGUY_CHAIN_CROSS_SECTION_COIL

The coil cross-section is good for wires. The chain masses are connected by line segments.

5.100.35 constants

```
#define DIGUY_LOAD_MANAGER_NUM_ZONES (4)
#define DIGUY_LOAD_MANAGER_NUM_NONVISIBLE_ZONE_INDEX (3)
```

Description:

5.100.36 enumeration diguyScenarioInputMode

```
typedef enum
{
    DIGUY_SCENARIO_INPUT_MODE_NONE = 0,
    DIGUY_SCENARIO_INPUT_MODE_CAMERA,
    DIGUY_SCENARIO_INPUT_MODE_PEOPLE_BLITZER,
    DIGUY_SCENARIO_INPUT_MODE_WAYPOINT_MOVE,
    DIGUY_SCENARIO_INPUT_MODE_WAYPOINT_CREATE,
    DIGUY_SCENARIO_INPUT_MODE_BEAD_MOVE,
    DIGUY_SCENARIO_INPUT_MODE_BEAD_CREATE,
    DIGUY_SCENARIO_INPUT_MODE_SENSOR_REGION_MOVE,
    DIGUY_SCENARIO_INPUT_MODE_SENSOR_REGION_CREATE,
    DIGUY_SCENARIO_INPUT_MODE_PLUGIN,
    DIGUY_SCENARIO_INPUT_MODE_POINT_SELECT,
    DIGUY_SCENARIO_INPUT_MODE_GROUP_MOVE,
    DIGUY_SCENARIO_INPUT_MODE_CHARACTER_MOVE,
    DIGUY_SCENARIO_INPUT_MODE_IGUY,
    DIGUY_SCENARIO_INPUT_MODE_CROWD_EDITOR,
    DIGUY_SCENARIO_INPUT_MODE_CROWD_BLITZER,
    DIGUY_SCENARIO_INPUT_MODE_REGION_PAINTER,
    DIGUY_SCENARIO_INPUT_MODE_NAVMESH_PAINTER,
    DIGUY_SCENARIO_INPUT_MODE_COUNT
} diguyScenarioInputMode;
```

Description:

This enumeration lists the input modes available in DI-Guy Scenario views (3D windows) and for DI-Guy Author.

5.100.37 enumeration diguyScenarioWindowId

```
typedef enum
{
    DIGS_WINDOW_ALL_WINDOWS = 0,
    DIGS_WINDOW_TIME_CONTROL,
    DIGS_WINDOW_INPUT_MODE,
    DIGS_WINDOW_ACTION_SPREADSHEET,
    DIGS_WINDOW_SIGNAL_PALETTE,
    DIGS_WINDOW_SCENARIO_ELEMENTS,
    DIGS_WINDOW_CHARACTER_ELEMENTS,
    DIGS_WINDOW_DIGUY_AI_ELEMENTS,
    DIGS_WINDOW_CROWD_ELEMENTS = DIGS_WINDOW_DIGUY_AI_ELEMENTS,
    DIGS_WINDOW_VIEW_SETTINGS,
    DIGS_WINDOW_EVENT_HANDLERS,
    DIGS_WINDOW_NETWORK_SETTINGS,
    DIGS_WINDOW_LOG,
    DIGS_WINDOW_MOVIE_SETTINGS,
    DIGS_WINDOW_SECONDARY_3D_VIEW_0,
    DIGS_WINDOW_SECONDARY_3D_VIEW_1,
    DIGS_WINDOW_SECONDARY_3D_VIEW_2,
    DIGS_WINDOW_SECONDARY_3D_VIEW_3,
    DIGS_WINDOW_SECONDARY_3D_VIEW_4,
    DIGS_WINDOW_SECONDARY_3D_VIEW_5,
    DIGS_WINDOW_SECONDARY_3D_VIEW_6,
    DIGS_WINDOW_SECONDARY_3D_VIEW_7,
}
```

```
DIGS_WINDOW_SECONDARY_3D_VIEW_8,
DIGS_WINDOW_COUNT
```

```
} diguyScenarioWindowId;
```

Description:

This enumeration lists the windows that can be shown in the DI-Guy Scenario application.

5.100.38 enumeration diguyScenarioEditAction

```
typedef enum
{
    // File menu
    DIGS_EDIT_ACTION_FILE_OPEN = 0,
    DIGS_EDIT_ACTION_FILE_MERGE,
    DIGS_EDIT_ACTION_FILE_SAVE,
    DIGS_EDIT_ACTION_FILE_SAVE_AS,
    DIGS_EDIT_ACTION_FILE_CLOSE,
    DIGS_EDIT_ACTION_FILE_SAVE_REVIEW_DATA,
    DIGS_EDIT_ACTION_FILE_LOAD_REVIEW_DATA,
    DIGS_EDIT_ACTION_FILE_EXIT,

    // Edit menu
    DIGS_EDIT_ACTION_EDIT_UNDO,
    DIGS_EDIT_ACTION_EDIT_CUT,
    DIGS_EDIT_ACTION_EDIT_COPY,
    DIGS_EDIT_ACTION_EDIT_PASTE,
    DIGS_EDIT_ACTION_EDIT_DELETE,
    DIGS_EDIT_ACTION_EDIT_PREFERENCES,

    // Help menu
    DIGS_EDIT_ACTION_HELP_KEY_MAPPINGS,
    DIGS_EDIT_ACTION_HELP_ABOUT,
    DIGS_EDIT_ACTION_HELP_API_DOCS,
```

```
// number of edit actions
DIGS_EDIT_ACTION_COUNT

} diguyScenarioEditAction;
```

Description:

This enumeration lists different types of editing actions that can be performed in the DI-Guy Scenario application.

5.100.39 enumeration

```
enum
{
    DIGUY_SPECIAL_KEY_MIN          = 0x1000000,
    DIGUY_SPECIAL_KEY_ESC          = 0x1000000,
    DIGUY_SPECIAL_KEY_TAB          = 0x1000001,
    DIGUY_SPECIAL_KEY_BACKTAB      = 0x1000002,
    DIGUY_SPECIAL_KEY_BACKSPACE    = 0x1000003,
    DIGUY_SPECIAL_KEY_RETURN       = 0x1000004,
    DIGUY_SPECIAL_KEY_ENTER        = 0x1000005,
    DIGUY_SPECIAL_KEY_INSERT       = 0x1000006,
    DIGUY_SPECIAL_KEY_DELETE       = 0x1000007,
    DIGUY_SPECIAL_KEY_PAUSE        = 0x1000008,
    DIGUY_SPECIAL_KEY_PRINT        = 0x1000009,
    DIGUY_SPECIAL_KEY_HOME         = 0x1000010,
    DIGUY_SPECIAL_KEY_END          = 0x1000011,
    DIGUY_SPECIAL_KEY_LEFT         = 0x1000012,
    DIGUY_SPECIAL_KEY_UP           = 0x1000013,
    DIGUY_SPECIAL_KEY_RIGHT        = 0x1000014,
    DIGUY_SPECIAL_KEY_DOWN         = 0x1000015,
    DIGUY_SPECIAL_KEY_PAGE_UP      = 0x1000016,
    DIGUY_SPECIAL_KEY_PAGE_DOWN    = 0x1000017,

    DIGUY_SPECIAL_F1               = 0x1000030,
    DIGUY_SPECIAL_F2               = 0x1000031,
```

DI-Guy SDK Version 12.5.2 Index

```
DIGUY_SPECIAL_F3          = 0x1000032,  
DIGUY_SPECIAL_F4          = 0x1000033,  
DIGUY_SPECIAL_F5          = 0x1000034,  
DIGUY_SPECIAL_F6          = 0x1000035,  
DIGUY_SPECIAL_F7          = 0x1000036,  
DIGUY_SPECIAL_F8          = 0x1000037,  
DIGUY_SPECIAL_F9          = 0x1000038,  
DIGUY_SPECIAL_F10         = 0x1000039,  
DIGUY_SPECIAL_F11         = 0x100003A,  
DIGUY_SPECIAL_F12         = 0x100003B,  
  
DIGUY_SPECIAL_KEY_MAX     = 0x10000FF  
};
```

Description:

This enumeration lists "special" (non-ASCII) keys that DI-Guy plugins may respond to.

(These enumerations currently match the Qt 4 Qt::Key_* values.)

5.100.40 enumeration diguyAimAlgorithm

```
typedef enum  
{  
    DIGUY_AIM_ALGORITHM_7 = 7,  
    DIGUY_AIM_ALGORITHM_8 = 8  
} diguyAimAlgorithm;
```

Description:

DI-Guy aim algorithms.

DI-Guy AI Enumerations

5.100.41 enumeration diguyDynamicAvoidanceMethod

```
typedef enum
{
    DIGUY_DYNAMIC_AVOIDANCE_METHOD_DISABLED = 0,
    DIGUY_DYNAMIC_AVOIDANCE_METHOD_REPULSION_ZONES,
    DIGUY_DYNAMIC_AVOIDANCE_METHOD_SIMPLE
} diguyDynamicAvoidanceMethod;
```

Description:

This enumeration lists the methods available for collision detection and avoidance against dynamic objects.

DIGUY_DYNAMIC_AVOIDANCE_METHOD_DISABLED

Dynamic objects are not detected or avoided. In effect this means that characters will walk through each other.

If a crowd is far away from the viewpoint, it is not obvious that this is happening. Disabling the crowd's dynamic avoidance can reduce the overhead of the crowd on performance.

DIGUY_DYNAMIC_AVOIDANCE_METHOD_REPULSION_ZONES

Characters are surrounded by zones of repulsion. Other characters that enter these zones have their position, orientation, and/or action changed to try to get out of the zones.

DIGUY_DYNAMIC_AVOIDANCE_METHOD_SIMPLE

This is an experimental method that is still being developed. It should not be used for now.

5.100.42 enumeration diguyStaticAvoidanceMethod

```
typedef enum
{
    DIGUY_STATIC_AVOIDANCE_METHOD_DISABLED = 0,
    DIGUY_STATIC_AVOIDANCE_METHOD_FEELERS

} diguyStaticAvoidanceMethod;
```

Description:

This enumeration lists the methods available for collision detection and avoidance against static objects in the environment. These are usually scene objects and non-moving props.

DIGUY_STATIC_AVOIDANCE_METHOD_DISABLED

Static objects are not detected or avoided. In effect this means that characters can walk through walls.

If a crowd is not near any buildings, walls, or solid objects, disabling static avoidance can reduce the overhead of the crowd on performance.

DIGUY_STATIC_AVOIDANCE_METHOD_FEELERS

The character uses 'feelers' to detect static objects and turn away from them.

5.100.43 `typedef diguyCharacterFeelerNumber`

```
typedef int diguyCharacterFeelerNumber;
```

Description:

This enumeration lists the standard feelers used for static object avoidance used by I-Guy and agent characters.

5.100.44 enumeration

```
enum
{
    DIGUY_CHARACTER_FEELER_F = 0,    // front
    DIGUY_CHARACTER_FEELER_FR,      // front right
    DIGUY_CHARACTER_FEELER_FL,      // front left
    DIGUY_CHARACTER_FEELER_L,       // left
    DIGUY_CHARACTER_FEELER_R,       // right
    DIGUY_CHARACTER_FEELER_BL,      // back left
    DIGUY_CHARACTER_FEELER_BR,      // back right
    DIGUY_CHARACTER_FEELER_B,       // back
    DIGUY_CHARACTER_FEELER_COUNT
};
```

5.100.45 enumeration diguyCharacterFeelerStatus

```
typedef enum
{
    DIGUY_CHARACTER_FEELER_STATUS_NO_CONTACT      = 0,
    DIGUY_CHARACTER_FEELER_STATUS_CONTACT_TURN    = 1,
    DIGUY_CHARACTER_FEELER_STATUS_CONTACT_COLLIDE = 2,
    DIGUY_CHARACTER_FEELER_STATUS_CONTACT_TURN_AND_COLLIDE = 3
} diguyCharacterFeelerStatus;
```

Description:

This enumeration lists the state character feelers may be in.

DIGUY_CHARACTER_FEELER_STATUS_NO_CONTACT

Feeler is not in contact with anything.

DIGUY_CHARACTER_FEELER_STATUS_CONTACT_TURN

Feeler is in contact with an object just enough to cause the character to begin to turn.

DIGUY_CHARACTER_FEELEER_STATUS_CONTACT_COLLIDE

Feeler is in close contact with an object. The object should be pushing the character away.

DIGUY_CHARACTER_FEELEER_STATUS_CONTACT_TURN_AND_COLLIDE

Feeler is within both turn and contact distances of an object.

5.100.46 enumeration diguyCharacterFeelerTurnDirection

```
typedef enum
{
    DIGUY_CHARACTER_FEELEER_TURN_DIRECTION_NONE = 0, // as string: "none"
    DIGUY_CHARACTER_FEELEER_TURN_DIRECTION_L,       // as string: "left"
    DIGUY_CHARACTER_FEELEER_TURN_DIRECTION_R,       // as string: "right"
    DIGUY_CHARACTER_FEELEER_TURN_DIRECTION_L_OR_R   // as string: "left_or_right"
} diguyCharacterFeelerTurnDirection;
```

Description:

This enumeration lists the turn directions a feeler can use when its status is [DIGUY_CHARACTER_FEELEER_STATUS_CONTACT_TURN](#).

DIGUY_CHARACTER_FEELEER_TURN_DIRECTION_NONE

Feeler does not turn the character.

DIGUY_CHARACTER_FEELEER_TURN_DIRECTION_L

Feeler turns the character to the left.

DIGUY_CHARACTER_FEELER_TURN_DIRECTION_R

Feeler turns the character to the right.

DIGUY_CHARACTER_FEELER_TURN_DIRECTION_L_OR_R

Feeler turns the character to the left or right, whichever will result in the turn behavior ending more quickly.

5.100.47 enumeration diguyNavPathDirection

```
typedef enum
{
    DIGUY_NAV_PATH_DIRECTION_UNKNOWN = 0,      // as string: "unknown"
    DIGUY_NAV_PATH_DIRECTION_FORWARD,          // as string: "forward"
    DIGUY_NAV_PATH_DIRECTION_BACKWARD,         // as string: "backward"
    DIGUY_NAV_PATH_DIRECTION_COUNT
} diguyNavPathDirection;
```

Description:

Directions a character can be traveling on a navigation path (nav path).

The travel behavior of agents use nav paths.

5.100.48 enumeration diguyNavPathAtEnd

```
typedef enum
{
    DIGUY_NAV_PATH_AT_END_UNKNOWN = 0,           // as string: "unknown"
    DIGUY_NAV_PATH_AT_END_REVERSE,               // as string: "reverse"
    DIGUY_NAV_PATH_AT_END_LOOP,                  // as string: "loop"
    DIGUY_NAV_PATH_AT_END_STOP,                  // as string: "stop"
    DIGUY_NAV_PATH_AT_END_TELEPORT,              // as string: "teleport"

    // internal use only
    DIGUY_NAV_PATH_AT_END_SWITCH_BEHAVIOR,

    DIGUY_NAV_PATH_AT_END_COUNT
} diguyNavPathAtEnd;
```

Description:

What a character following a nav path will do once it reaches the end of a nav path.

The travel behavior of agents use nav paths.

DIGUY_NAV_PATH_AT_END_REVERSE

Agent reverses direction on the nav path, going through waypoints in opposite order.

DIGUY_NAV_PATH_AT_END_LOOP

Agent's next waypoint is waypoint 0. Best when last waypoint and first waypoint are close to each other, as in a loop path.

DIGUY_NAV_PATH_AT_END_STOP

Stop at the end, switching to behavior none.

DIGUY_NAV_PATH_AT_END_TELEPORT

Teleport to the beginning. In general agents should be out of view at the beginning and end of the path, if possible.

5.100.49 enumeration diguyNavPathSearchResults

```
typedef enum
{
    DIGUY_NAV_PATH_ERROR_NONE = 0,
    DIGUY_NAV_PATH_ERROR_OUT_OF_TIME = 1,
    DIGUY_NAV_PATH_ERROR_SEARCH_QUEUED = 2,
    DIGUY_NAV_PATH_ERROR_UNKNOWN = -1,
    DIGUY_NAV_PATH_ERROR_INVALID_CANVAS_OR_REGION = -2,
    DIGUY_NAV_PATH_ERROR_INVALID_START_POINT = -3,
    DIGUY_NAV_PATH_ERROR_INVALID_END_POINT = -4,
    DIGUY_NAV_PATH_ERROR_INVALID_LENGTH = -5,
    DIGUY_NAV_PATH_ERROR_FAILED_TO_FIND_PATH = -6
} diguyNavPathSearchResults;
```

Description:

Errors various nav path functions may return.

5.100.50 enumeration diguyCrowdProfileSelectionMethod

```
typedef enum
{
    DIGUY_CROWD_PROFILE_SELECTION_METHOD_RANDOM = 0,
    DIGUY_CROWD_PROFILE_SELECTION_METHOD_SEQUENTIAL,
    DIGUY_CROWD_PROFILE_SELECTION_METHOD_NUM_METHODS
} diguyCrowdProfileSelectionMethod;
```

Description:

Selection methods for records from crowd profiles.

5.100.51 enumeration diguyAgentRepulsionZone

```
typedef enum
{
    DIGUY_AGENT_REPULSION_NONE = 0,
    DIGUY_AGENT_REPULSION_LOW,
    DIGUY_AGENT_REPULSION_MEDIUM,
    DIGUY_AGENT_REPULSION_HIGH,
    DIGUY_AGENT_REPULSION_SOLID,
    DIGUY_AGENT_REPULSION_NUM_ZONES

} diguyAgentRepulsionZone;
```

Description:

DI-Guy agent repulsion zones for characters with dynamic avoidance method [DIGUY_DYNAMIC_AVOIDANCE_METHOD_REPULSION_ZONES](#).

5.100.52 enumeration diguyAgentBehavior

```
typedef enum
{
    DIGUY_AGENT_BEHAVIOR_UNKNOWN = -1,    // as string: "unknown"
    DIGUY_AGENT_BEHAVIOR_NONE = 0,        // as string: "none"
    DIGUY_AGENT_BEHAVIOR_TRAVEL,          // as string: "travel"
    DIGUY_AGENT_BEHAVIOR_PATH_FOLLOW,     // as string: "path_follow"
    DIGUY_AGENT_BEHAVIOR_WANDER,          // as string: "wander"
    DIGUY_AGENT_BEHAVIOR_FLEE,            // as string: "flee"
    DIGUY_AGENT_BEHAVIOR_PURSUE,          // as string: "pursue"
    DIGUY_AGENT_BEHAVIOR_MINGLE,           // as string: "mingle"
    DIGUY_AGENT_BEHAVIOR_ATTACK,          // as string: "attack"
    DIGUY_AGENT_BEHAVIOR_IDLE,            // as string: "idle"
    DIGUY_AGENT_BEHAVIOR_NUM_BEHAVIORS

} diguyAgentBehavior;
```

Description:

DI-Guy agent behaviors.

Following is a short description of each behavior. See the DI-Guy AI documentation for more details.

DIGUY_AGENT_BEHAVIOR_NONE

Agent functionality is completely inactive. The character's actions are determined by its character path or direct DI-Guy API calls.

DIGUY_AGENT_BEHAVIOR_TRAVEL

Agent attempts to travel close to a path shape. Many agents can use the same path shape for the travel behavior.

DIGUY_AGENT_BEHAVIOR_PATH_FOLLOW

Similar to [DIGUY_AGENT_BEHAVIOR_TRAVEL](#), but agent follows a character path instead of a path shape, and attempts to match the timing of the character path in addition to the path shape. If the agent is interrupted from the path, it will attempt to resume it where it left off.

DIGUY_AGENT_BEHAVIOR_WANDER

Agent wanders a region. In this case wander means it will select a random point within the region and move close to it, and then wait there for a period of time before selecting a new point in the region.

DIGUY_AGENT_BEHAVIOR_MINGLE

Similar to wander, but agents attempt to move together to various places in the region.

DIGUY_AGENT_BEHAVIOR_FLEE

Agent attempts to move away from a specified character or point.

DIGUY_AGENT_BEHAVIOR_PURSUE

Agent attempts to stay close to a specified character or member of a group.

DIGUY_AGENT_BEHAVIOR_ATTACK

Similar to pursue, but once agents are close enough they will attack the target.

DIGUY_AGENT_BEHAVIOR_IDLE

Agent has no particular automatic behavior; it follows its path or moves toward its desired position as set by [diguyCharacter::set_desired_position\(\)](#). It will affect other agents in its crowd, and if pushed away from its desired position it will move back to it.

5.100.53 enumeration diguyAgentUntrapMethod

```
typedef enum
{
    DIGUY_AGENT_UNTRAP_METHOD_NONE = 0,           // as string: "none"
    DIGUY_AGENT_UNTRAP_METHOD_NAV_PATH,           // as string: "nav_path"
    DIGUY_AGENT_UNTRAP_METHOD_RANDOM_TURN,        // as string: "random_turn"
    DIGUY_AGENT_UNTRAP_METHOD_GHOST,              // as string: "ghost"
    DIGUY_AGENT_UNTRAP_METHOD_TELEPORT,           // as string: "teleport"
    DIGUY_AGENT_UNTRAP_METHOD_STOP_MOVING,        // as string: "stop_moving"
    DIGUY_AGENT_UNTRAP_METHOD_NUM_METHODS
} diguyAgentUntrapMethod;
```

Description:

DI-Guy agent untrap methods.

Following is a short description of each untrap method. See the DI-Guy AI documentation for more details.

DIGUY_AGENT_UNTRAP_METHOD_NONE

The agent makes no attempt to become untrapped. This can result in the agent blindly walking into a wall until its desired position changes.

DIGUY_AGENT_UNTRAP_METHOD_NAV_PATH

The agent will try to create a nav path in its current region to its desired position. If the desired position is not inside the region, a valid nav path may not be able to be found.

DIGUY_AGENT_UNTRAP_METHOD_RANDOM_TURN

The agent will stop moving and turn a random amount. This minor course change can sometimes move a character around a small obstacle.

DIGUY_AGENT_UNTRAP_METHOD_GHOST

Agent feelers will stop having an effect for a short time. This allows the agent to walk through a wall toward its desired position.

DIGUY_AGENT_UNTRAP_METHOD_TELEPORT

This is an extreme method, typically used as a last resort. The agent abruptly teleports to its desired position.

DIGUY_AGENT_UNTRAP_METHOD_STOP_MOVING

Somewhat similar to [DIGUY_AGENT_UNTRAP_METHOD_NONE](#), but the current behavior of the agent is set to none, and the agent character's desired

position is set to its current position.

5.100.54 enumeration diguyRegionPaintbrushMode

```
typedef enum
{
    DIGUY_REGION_PAINTBRUSH_MODE_ADDITIVE = 0,
    DIGUY_REGION_PAINTBRUSH_MODE_ERASE,
    DIGUY_REGION_PAINTBRUSH_MODE_UNGROUND_CLAMP,
    DIGUY_REGION_PAINTBRUSH_MODE_MOVE_MESH,
    DIGUY_REGION_PAINTBRUSH_MODE_REACHABLE,
    DIGUY_REGION_PAINTBRUSH_INTERNAL_MODE_UPDATE_BEHAVIOR_PATH,
    DIGUY_REGION_PAINTBRUSH_INTERNAL_MODE_REPLACE,
    DIGUY_REGION_PAINTBRUSH_NUM_MODES
} diguyRegionPaintbrushMode;
```

Description:

DI-Guy region painting modes.

Painting modes identify what happens when a region paintbrush is used in some of the DI-Guy input modes.

DIGUY_REGION_PAINTBRUSH_MODE_ADDITIVE

Painting will add area to the current subregion. This may end up adding area to the base subregion, as well, as it contains at least the areas of all subregions.

DIGUY_REGION_PAINTBRUSH_MODE_ERASE

Painting will remove area from the current subregion. If the base subregion is erased, all matching areas in other subregions will be erased, as well.

DIGUY_REGION_PAINTBRUSH_MODE_UNGROUND_CLAMP

Documentation pending.

DIGUY_REGION_PAINTBRUSH_MODE_MOVE_MESH

Documentation pending.

DIGUY_REGION_PAINTBRUSH_MODE_REACHABLE

Documentation pending.

DIGUY_REGION_PAINTBRUSH_INTERNAL_MODE_UPDATE_BEHAVIOR_PATH

Internal use only.

DIGUY_REGION_PAINTBRUSH_INTERNAL_MODE_REPLACE

Internal use only.

5.100.55 enumeration diguySubregionIndex

```
typedef enum
{
    DIGUY_SUBREGION_BASE = 0,           // as string: "base"
    DIGUY_SUBREGION_POPULATE,          // as string: "populate"

    DIGUY_SUBREGION_RED,               // as string: "red"
    DIGUY_SUBREGION_GREEN,             // as string: "green"
    DIGUY_SUBREGION_BLUE,              // as string: "blue"
    DIGUY_SUBREGION_YELLOW,            // as string: "yellow"
    DIGUY_SUBREGION_ORANGE,            // as string: "orange"
    DIGUY_SUBREGION_WHITE,             // as string: "white"

    DIGUY_SUBREGION_ROAD,              // as string: "road"
    DIGUY_SUBREGION_SIDEWALK,          // as string: "sidewalk"
    DIGUY_SUBREGION_CROSSWALK,         // as string: "crosswalk"

    DIGUY_SUBREGION_NUM_SUBREGIONS

} diguySubregionIndex;
```

Description:

DI-Guy subregions.

Regions are irregularly shaped areas in the scene that can be used in a number of ways.

Regions contain a number of subregions. Each subregion is represented by a different color, and most often the subregions are referred to by their color. Subregions are not mutually exclusive; a point in the scene may be in all of the base, populate, and green subregions, for example.

Many functions that take one subregion as an argument use the string name of the subregion.

Functions that can have multiple subregions specified use a subregion mask made up of [diguySubregionMask](#) values that can be or'ed together to indicate multiple valid subregions. See [diguySubregionMask](#) for more information.

DIGUY_SUBREGION_BASE

This subregion is really a "super-region", in that its bounds contain all of the other subregions, and possibly more.

If any subregion of a top-level region is expanded beyond the bounds of the base subregion, the base subregion will be expanded until it encompasses the expansion.

If, however, a non-base subregion is subsequently shrunk, the base subregion will not shrink.

If the base subregion is shrunk, any portion of each non-base subregion that is no longer in the base subregion will be removed.

DIGUY_SUBREGION_POPULATE

The subregion into which new crowd agents will be placed if the region is populated or repopulated. This subregion is typically purple.

DIGUY_SUBREGION_[color]

The "color" subregions are free to be used in any way a scenario designer wants. Their colors are based on their name.

DIGUY_SUBREGION_ROAD

Road subregions are commonly used as preferred subregions for vehicle travel, and repulsion subregions for human travel. This subregion is typically gray.

DIGUY_SUBREGION_SIDEWALK

Sidewalk subregions are commonly used as preferred subregions for human travel, and repulsion subregions for vehicles. This subregion is typically white.

DIGUY_SUBREGION_CROSSWALK

Humans will tend to prefer crosswalk subregions over road subregions, but prefer sidewalk subregions over crosswalk subregions. This subregion is typically a light yellow.

5.100.56 enumeration diguySubregionMask

```
typedef enum
{
    DIGUY_SUBREGION_MASK_NONE           = 0x00000000,
    DIGUY_SUBREGION_MASK_BASE           = 0x00000020,
    DIGUY_SUBREGION_MASK_POPULATE       = 0x00000040,

    DIGUY_SUBREGION_MASK_RED            = 0x00000080,
    DIGUY_SUBREGION_MASK_GREEN          = 0x00000100,
    DIGUY_SUBREGION_MASK_BLUE           = 0x00000200,
    DIGUY_SUBREGION_MASK_YELLOW         = 0x00000400,
    DIGUY_SUBREGION_MASK_ORANGE         = 0x00000800,
    DIGUY_SUBREGION_MASK_WHITE          = 0x00001000,

    DIGUY_SUBREGION_MASK_ROAD           = 0x00002000,
    DIGUY_SUBREGION_MASK_SIDEWALK       = 0x00004000,
    DIGUY_SUBREGION_MASK_CROSSWALK      = 0x00008000,

} diguySubregionMask;
```

Description:

DI-Guy subregions mask values.

Example:

To specify that a function can use either the red or blue subregions, the following syntax is used.

In Perl or C++:

```
DIGUY_SUBREGION_MASK_RED | DIGUY_SUBREGION_MASK_BLUE
```

In Lua:

```
bit.bor( DIGUY_SUBREGION_MASK_RED, DIGUY_SUBREGION_MASK_BLUE, ...)
```

Lua does not natively support bitfields, but support is automatically added via the "LuaBit" library located in \$(DIGUY)binlua/bit.lua.

5.100.57 enumeration diguyGeometryTag

```
typedef enum
{
    DIGUY_GEOMETRY_TAG_NONE           = 0x00000000,
    DIGUY_GEOMETRY_TAG_DOOR           = 0x00000010,
    DIGUY_GEOMETRY_TAG_INSIDE          = 0x00000020,
    DIGUY_GEOMETRY_TAG_OUTSIDE         = 0x00000040,
    DIGUY_GEOMETRY_TAG_ROOF            = 0x00000080,

} diguyGeometryTag;
```

Description:

DI-Guy geometry tags.

5.100.58 enumeration diguyNavMeshPathPlannerCostMask

```
typedef enum
{
    DIGUY_NAVMESH_SHORTEST_PATH       = 0x00000001,
    DIGUY_NAVMESH_LEAST_VISIBLE        = 0x00000002,
    DIGUY_NAVMESH_MESH_EDGE            = 0x00000004,
    DIGUY_NAVMESH_MESH_EDGE_AND_LEAST_VIS = 0x00000006,
    DIGUY_NAVMESH_MESH_AVOID_EDGES     = 0x00000008,

} diguyNavMeshPathPlannerCostMask;
```

Description:

DI-Guy navigation mesh path planner cost masks.

5.100.59 enumeration diguyTacticsUnitType

```
typedef enum
{
    DIGUY_TACTICS_UNIT_TYPE_UNKNOWN      = 0,

    DIGUY_TACTICS_UNIT_TYPE_MARINE,      // an individual
    DIGUY_TACTICS_UNIT_TYPE_MARINE_FIRETEAM,
    DIGUY_TACTICS_UNIT_TYPE_MARINE_SQUAD,
    DIGUY_TACTICS_UNIT_TYPE_MARINE_PLATOON,
    DIGUY_TACTICS_UNIT_TYPE_MARINE_COMPANY,
    DIGUY_TACTICS_UNIT_TYPE_MARINE_BATTALION,
    DIGUY_TACTICS_UNIT_TYPE_MARINE_REGIMENT,
    DIGUY_TACTICS_UNIT_TYPE_MARINE_DIVISION,
    DIGUY_TACTICS_UNIT_TYPE_MARINE_CORPS,

    DIGUY_TACTICS_UNIT_TYPE_ARMY_SOLDIER,
    DIGUY_TACTICS_UNIT_TYPE_ARMY_SQUAD,
    DIGUY_TACTICS_UNIT_TYPE_ARMY_PLATOON,
    DIGUY_TACTICS_UNIT_TYPE_ARMY_COMPANY,
    DIGUY_TACTICS_UNIT_TYPE_ARMY_BATTALION,
    DIGUY_TACTICS_UNIT_TYPE_ARMY_BRIGADE,
    DIGUY_TACTICS_UNIT_TYPE_ARMY_DIVISION,
    DIGUY_TACTICS_UNIT_TYPE_ARMY_CORPS,
    DIGUY_TACTICS_UNIT_TYPE_ARMY        // two or more corps

} diguyTacticsUnitType;
```

5.100.60 enumeration diguyDatetimeAdvanceMethod

```
typedef enum
{
    DIGUY_DATETIME_ADVANCE_METHOD_MANUAL = 0,
    DIGUY_DATETIME_ADVANCE_METHOD_REAL_TIME_RATE,
    DIGUY_DATETIME_ADVANCE_METHOD_SIMULATION_TIME_RATE

} diguyDatetimeAdvanceMethod;
```

Description:

This enumeration lists the methods in which the datetime of a scenario can advance.

DIGUY_DATETIME_ADVANCE_METHOD_MANUAL

Datetime does not advance automatically. It can only be set manually, e.g. by a `diguyScenario::set_current_datetime()` call.

DIGUY_DATETIME_ADVANCE_METHOD_REAL_TIME_RATE

Datetime advances at the same rate as realtime.

DIGUY_DATETIME_ADVANCE_METHOD_SIMULATION_TIME_RATE

Datetime advances at the same rate as simulation time.

5.100.61 enumeration diguyDatetimeNetworkMode

```
typedef enum
{
    DIGUY_DATETIME_NETWORK_MODE_LOCAL_ONLY = 0,
    DIGUY_DATETIME_NETWORK_MODE_BROADCAST,
    DIGUY_DATETIME_NETWORK_MODE_RECEIVE
} diguyDatetimeNetworkMode;
```

Description:

DIGUY_DATETIME_NETWORK_MODE_LOCAL_ONLY

Datetime data will be handled locally on this host. No data will be sent on the network, and no data will be read from the network.

DIGUY_DATETIME_NETWORK_MODE_BROADCAST

Datetime data will be sent from this host to other hosts in the network. The datetime will be encoded in a custom [DIGUY_CUSTOM_PDU_ID_BROADCAST_EXERCISE_DATETIME](#) PDU. In general there should be only one host on the network broadcasting

datetime data; all others should receive or ignore it.

DIGUY_DATETIME_NETWORK_MODE_RECEIVE

Datetime data will be read from the network from custom

[DIGUY_CUSTOM_PDU_ID_BROADCAST_EXERCISE_DATETIME](#) PDUs.

DI-Guy Physics Enumerations

5.100.62 enumeration diguySpeedUnits

```
typedef enum
{
    DIGUY_SPEED_UNITS_METERS_PER_SECOND = 0,
    DIGUY_SPEED_UNITS_KILOMETERS_PER_HOUR,
    DIGUY_SPEED_UNITS_MILES_PER_HOUR

} diguySpeedUnits;
```

Description:

This enumeration lists units of speed that can be passed to some DI-Guy API functions.

Note that if one of these values is used specifying an acceleration, add an implicit "per second" to the end. e.g., if 10.0 is passed as an acceleration, with units [DIGUY_SPEED_UNITS_MILES_PER_HOUR](#), the resulting acceleration would be 10.0 miles per hour per second; or in other words, every 1 second speed would increase by 10.0 miles per hour.

5.100.63 enumeration diguyCharacterSimulator

```
typedef enum
{
    DIGUY_CHARACTER_SIMULATOR_UNSET = 0,
    DIGUY_CHARACTER_SIMULATOR_DIGUY_MOTION_ENGINE,
    DIGUY_CHARACTER_SIMULATOR_DIGUY_CHAIN_SIMULATOR,
    DIGUY_CHARACTER_SIMULATOR_DIGUY_VEHICLE_SIMULATOR,
    DIGUY_CHARACTER_SIMULATOR_PHYSICS_SIM_MODULE

} diguyCharacterSimulator;
```

Description:

This enumeration lists the ways that data can be generated for DI-Guy characters. Data typically comes from the DI-Guy motion engine, but at times may be generated by other simulation methods.

5.100.64 enumeration diguyPhysicsSimulationMode

```
typedef enum
{
    DIGUY_PHYSICS_SIMULATION_MODE_UNSET = 0,
    DIGUY_PHYSICS_SIMULATION_MODE_SIM_DRIVES_CHARACTER,
    DIGUY_PHYSICS_SIMULATION_MODE_CHARACTER_DRIVES_SIM,
    DIGUY_PHYSICS_SIMULATION_MODE_CHARACTER_DRIVES_SIM_GOAL_DATA // not yet sup

} diguyPhysicsSimulationMode;
```

Description:

This enumeration lists the modes that a DI-Guy character physics sim can be in, when the character simulator is set to [DIGUY_CHARACTER_SIMULATOR_PHYSICS_SIM_MODULE](#).

5.100.65 enumeration diguyPhysicsCollisionGroup

```
typedef enum
{
    DIGUY_PHYSICS_COLLISION_GROUP_CHARACTER0,
    DIGUY_PHYSICS_COLLISION_GROUP_CHARACTER1,
    DIGUY_PHYSICS_COLLISION_GROUP_CHARACTER2,
    DIGUY_PHYSICS_COLLISION_GROUP_CHARACTER3,
    DIGUY_PHYSICS_COLLISION_GROUP_CHARACTER_EVERYTHING,
    DIGUY_PHYSICS_COLLISION_GROUP_CHARACTER_GHOST,

    DIGUY_PHYSICS_COLLISION_GROUP_PROP0,
    DIGUY_PHYSICS_COLLISION_GROUP_PROP1,
    DIGUY_PHYSICS_COLLISION_GROUP_PROP2,
    DIGUY_PHYSICS_COLLISION_GROUP_PROP3,

    DIGUY_PHYSICS_COLLISION_GROUP_SCENE_OBJECT0,
    DIGUY_PHYSICS_COLLISION_GROUP_SCENE_OBJECT1,
    DIGUY_PHYSICS_COLLISION_GROUP_SCENE_OBJECT2,
    DIGUY_PHYSICS_COLLISION_GROUP_SCENE_OBJECT3,

    DIGUY_PHYSICS_COLLISION_GROUP_EFFECT0,

    DIGUY_PHYSICS_COLLISION_GROUP_COUNT

} diguyPhysicsCollisionGroup;
```

Description:

This enumeration lists the collision groups that a DI-Guy physics sim object can be in. Characters in different collision groups may not have collisions calculated between them.

5.100.66 enumeration diguyVehicleSimulationMode

```
typedef enum
{
    DIGUY_VEHICLE_SIMULATION_MODE_DEFAULT = 0,
    DIGUY_VEHICLE_SIMULATION_MODE_NONE,
    DIGUY_VEHICLE_SIMULATION_MODE_REAR_AXLE_FOLLOWING,
    DIGUY_VEHICLE_SIMULATION_MODE_STEERING,
    DIGUY_VEHICLE_SIMULATION_MODE_FULL_PHYSICS
} diguyVehicleSimulationMode;
```

Description:

This enumeration lists the modes that the DI-Guy vehicle simulator can be in.

DI-Guy Author Enumerations

DI-Guy Shader Enumerations

5.100.67 enumeration diguyGraphicsShaderLanguage

```
typedef enum
{
    DIGUY_GRAPHICS_SHADER_LANGUAGE_GLSL = 0,
    DIGUY_GRAPHICS_SHADER_LANGUAGE_HLSL = 1
} diguyGraphicsShaderLanguage;
```

Description:

This enumeration lists supported shader languages.

5.101 DI-Guy API, DIS Constants and Enumerations

diguy_dis_constants.h

DI-Guy SDK Version 12.5.2

Contents:

This file contains constants defined by the DIS networking standard. They should be compatible with other DIS networking applications that follow the DIS standard.

5.101.1 enumeration

```
enum
{
    DIGUY_DIS_LIFEFORM_UNUSED                = 0,
    DIGUY_DIS_LIFEFORM_UPRIGHT_STANDING_STILL = 1,
    DIGUY_DIS_LIFEFORM_UPRIGHT_WALKING        = 2,
    DIGUY_DIS_LIFEFORM_UPRIGHT_RUNNING        = 3,
    DIGUY_DIS_LIFEFORM_KNEELING                = 4,
    DIGUY_DIS_LIFEFORM_PRONE                   = 5,
    DIGUY_DIS_LIFEFORM_CRAWLING                = 6,
    DIGUY_DIS_LIFEFORM_SWIMMING                = 7,
    DIGUY_DIS_LIFEFORM_PARACHUTING             = 8,
    DIGUY_DIS_LIFEFORM_JUMPING                 = 9,

    DIGUY_DIS_LIFEFORM_SITTING                = 10,
    DIGUY_DIS_LIFEFORM_SQUATTING               = 11,
    DIGUY_DIS_LIFEFORM_CROUCHING               = 12,
    DIGUY_DIS_LIFEFORM_WADING                  = 13
};
```

Description:

Lifeform state values from an Entity State PDU.

Note that the values for sitting, squatting, crouching, and wading don't appear to be part of the official DIS standard, but are included for compatibility with MaK Technologies, Inc. VR-Link.

5.101.2 enumeration

```
enum
{
    DIGUY_DIS_PRIMARY_WEAPON_NOT_PRESENT      = 0,
    DIGUY_DIS_PRIMARY_WEAPON_STOWED           = 1,
    DIGUY_DIS_PRIMARY_WEAPON_DEPLOYED         = 2,
    DIGUY_DIS_PRIMARY_WEAPON_IN_FIRING_POSITION = 3
};
```

Description:

Primary weapon position values from an Entity State PDU.

5.101.3 enumeration

```
enum
{
    DIGUY_DIS_SECONDARY_WEAPON_NOT_PRESENT      = 0,
    DIGUY_DIS_SECONDARY_WEAPON_STOWED           = 1,
    DIGUY_DIS_SECONDARY_WEAPON_DEPLOYED         = 2,
    DIGUY_DIS_SECONDARY_WEAPON_IN_FIRING_POSITION = 3
};
```

Description:

Secondary weapon position values from an Entity State PDU.

5.101.4 enumeration

```
enum
{
    DIGUY_DIS_DAMAGE_NONE          = 0,
    DIGUY_DIS_DAMAGE_SLIGHT        = 1,
    DIGUY_DIS_DAMAGE_MODERATE      = 2,
    DIGUY_DIS_DAMAGE_DESTROYED     = 3
};
```

Description:

Damage appearance values from an Entity State PDU.

5.101.5 enumeration

```
enum
{
    DIGUY_DIS_SMOKE_NOT_SMOKING          = 0,
    DIGUY_DIS_SMOKE_SMOKE_PLUME          = 1,
    DIGUY_DIS_SMOKE_ENGINE_SMOKE         = 2,
    DIGUY_DIS_SMOKE_ENGINE_SMOKE_AND_SMOKE_PLUME = 3
};
```

Description:

Smoke appearance values from an Entity State PDU.

5.101.6 enumeration

```
enum
{
    DIGUY_DIS_TRAILING_EFFECT_NONE    = 0,
    DIGUY_DIS_TRAILING_EFFECT_SMALL  = 1,
    DIGUY_DIS_TRAILING_EFFECT_MEDIUM = 2,
    DIGUY_DIS_TRAILING_EFFECT_LARGE   = 3
};
```

Description:

Trailing effect values from an Entity State PDU.

5.101.7 enumeration

```
enum
{
    DIGUY_DIS_FLAMES_NONE    = 0,
    DIGUY_DIS_FLAMES_PRESENT = 1
};
```

Description:

Flames appearance values from an Entity State PDU.

5.101.8 enumeration

```
enum
{
    DIGUY_DIS_POWER_PLANT_OFF = 0,
    DIGUY_DIS_POWER_PLANT_ON  = 1
};
```

Description:

Power plant status values from an Entity State PDU.

5.101.9 enumeration

```
enum
{
    DIGUY_DIS_MISSILE_TRAIL_OFF = 0,
    DIGUY_DIS_MISSILE_TRAIL_ON  = 1
};
```

Description:

This enumeration lists missile trail status values from an Entity State PDU.

5.101.10 enumeration

```
enum
{
    DIGUY_DIS_FOCUS_TARGET = 10240
};
```

Description:

This enumeration lists the Articulated part info for DI-Guy's gazing.

5.101.11 enumeration

```
enum diguyCustomPDUType
{
    DIGUY_CUSTOM_PDU_ID_ESCAPE_RESERVED = 0,
    DIGUY_CUSTOM_PDU_ID_SET_CHAR_TYPE,
    DIGUY_CUSTOM_PDU_ID_SET_CHAR_APPEARANCE,
    DIGUY_CUSTOM_PDU_ID_SET_CHAR_ACTION,
    DIGUY_CUSTOM_PDU_ID_SET_CHAR_VISIBLE,
    DIGUY_CUSTOM_PDU_ID_SET_CHAR_PAUSED, // (no longer used)
    DIGUY_CUSTOM_PDU_ID_SET_SCENARIO_FRAME = 6,
    DIGUY_CUSTOM_PDU_ID_SET_SCENARIO_PLAYBACK_MODE,
    DIGUY_CUSTOM_PDU_ID_LOAD_SCENARIO_CAMERA,
    DIGUY_CUSTOM_PDU_ID_RESET,
    DIGUY_CUSTOM_PDU_ID_HAIL, // 10
    DIGUY_CUSTOM_PDU_ID_LOAD_SCENARIO_FILE,
    DIGUY_CUSTOM_PDU_ID_SET_CAMERA_DETAILS,
    DIGUY_CUSTOM_PDU_ID_XC_LOAD_SCENARIO = 13, // 13
    DIGUY_CUSTOM_PDU_ID_XC_LOAD_SCENARIO_COMPLETE,
    DIGUY_CUSTOM_PDU_ID_XC_MERGE_SCENARIO,
    DIGUY_CUSTOM_PDU_ID_XC_MERGE_SCENARIO_COMPLETE,
    DIGUY_CUSTOM_PDU_ID_XC_SET_TIN_TOD,
    DIGUY_CUSTOM_PDU_ID_XC_TIME_NOW,
    DIGUY_CUSTOM_PDU_ID_XC_CONTROL,
    DIGUY_CUSTOM_PDU_ID_XC_TRIGGER_SCRIPT, // 20
    DIGUY_CUSTOM_PDU_ID_XC_EVAL_SCRIPT,
```

DI-Guy SDK Version 12.5.2 Index

```

DIGUY_CUSTOM_PDU_ID_XC_SET_FF_SPEED,
DIGUY_CUSTOM_PDU_ID_XC_TRIGGER_SIGNAL,
DIGUY_CUSTOM_PDU_ID_XC_RESET_SCENARIO,
DIGUY_CUSTOM_PDU_ID_XC_ENTER_SLAVE_MODE,
DIGUY_CUSTOM_PDU_ID_XC_EXIT_SLAVE_MODE,
DIGUY_CUSTOM_PDU_ID_VARIABLE_STATE,
DIGUY_CUSTOM_PDU_ID_SAVE_SCENARIO,
DIGUY_CUSTOM_PDU_ID_SAVE_SCENARIO_COMPLETE,
DIGUY_CUSTOM_PDU_ID_AI_SEND_MESSAGE_TO_CHARACTER,      // 30
DIGUY_CUSTOM_PDU_ID_AI_SEND_MESSAGE_TO_GROUP,
DIGUY_CUSTOM_PDU_ID_AI_BROADCAST_MESSAGE,
DIGUY_CUSTOM_PDU_ID_AI_BROADCAST_MESSAGE_TO_GROUP,
DIGUY_CUSTOM_PDU_ID_BEGIN_APPEARANCE_EFFECT,
DIGUY_CUSTOM_PDU_ID_END_APPEARANCE_EFFECT,
DIGUY_CUSTOM_PDU_ID_STOP_APPEARANCE_EFFECT,
DIGUY_CUSTOM_PDU_ID_SET_SCENE_OBJECT_BIT,
DIGUY_CUSTOM_PDU_ID_BROADCAST_EXERCISE_DATETIME,
DIGUY_CUSTOM_PDU_ID_REQUEST_EXERCISE_DATETIME_BROADCAST, // unused
DIGUY_CUSTOM_PDU_ID_SET_SHAPE_VARIATION,               // 40
DIGUY_CUSTOM_PDU_ID_EXECUTE_GESTURE,
DIGUY_CUSTOM_PDU_ID_CHARACTER_SCALE,
DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED0,
DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED1,
DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED2,
DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED3,
DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED4,
DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED5,
DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED6,
DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED7,
DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED8,
DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED9,
DIGUY_CUSTOM_PDU_ID_COUNT                             // 51
} ;

\\enum diguyCustomPDUType

```

Description:

This enumeration lists the message IDs for DI-Guy's custom pdus, not all are usable outside of diguyscenario

DIGUY_CUSTOM_PDU_ID_SET_CHAR_TYPE

- ◊ Contents: String
- ◊ Meaning: call [diguyCharacter::set_character_type\(\)](#)

DIGUY_CUSTOM_PDU_ID_SET_CHAR_APPEARANCE

- ◊ Contents: String A, String B
- ◊ Meaning: call [diguyCharacter::set_appearance\(\)](#)
optionally call
[diguyCharacter::set_current_head_appearance\(\)](#)

DIGUY_CUSTOM_PDU_ID_SET_CHAR_ACTION

- ◊ Contents: String A, String B

DI-Guy SDK Version 12.5.2 Index

- ◇ Meaning: Set the character action to A, the second string is either "p" or "" (the empty string). A value of "p" for the second string indicates that the action is paused. Characters are typically paused if the sending application such as DI-Guy Scenario has been playing and the user hits "pause" on the VCR control.

DIGUY_CUSTOM_PDU_ID_SET_CHAR_VISIBLE

- ◇ Contents: Long Boolean

DIGUY_CUSTOM_PDU_ID_SET_SCENARIO_FRAME

- ◇ Contents: Long val
- ◇ Meaning: call `diguyScenario::set_frame_desired(val)`

DIGUY_CUSTOM_PDU_ID_SET_SCENARIO_PLAYBACK_MODE

- ◇ Contents: Long val
- ◇ Meaning: call `diguyScenario::set_playback_mode(val)`

DIGUY_CUSTOM_PDU_ID_LOAD_SCENARIO_CAMERA

- ◇ Contents: string
- ◇ Meaning: call `diguyScenario::get_scenario_camera( )->load_settings(string)`

DIGUY_CUSTOM_PDU_ID_RESET

- ◇ Meaning: call `diguyScenario::reset( )`

DIGUY_CUSTOM_PDU_ID_HAIL

DIGUY_CUSTOM_PDU_ID_LOAD_SCENARIO_FILE

- ◇ Contents: string
- ◇ Meaning: call `diguyScenario::load(string)`

DIGUY_CUSTOM_PDU_ID_SET_CAMERA_DETAILS

DIGUY_CUSTOM_PDU_ID_XC_LOAD_SCENARIO

- ◇ Contents:
 - NetU16 => Target_site - NetU16 => Target_host - String => Filename - String => Reference name

DIGUY_CUSTOM_PDU_ID_XC_LOAD_SCENARIO_COMPLETE

- ◇ Contents: string
- ◇ Meaning: Indicates scenario has finished loading

DIGUY_CUSTOM_PDU_ID_XC_MERGE_SCENARIO

◇ Contents:

– NetU16 => Target_site – NetU16 => Target_host – NetFlt32 => Time of Day – NetFlt32[3] => Refx, Refy, Refz – NetFlt32[3] => x, y, z – NetFlt32[3] => w,r,p – String => Filename – String => Reference name

DIGUY_CUSTOM_PDU_ID_XC_MERGE_SCENARIO_COMPLETE

◇ Contents: string

◇ Meaning: Indicates scenario has finished loading

DIGUY_CUSTOM_PDU_ID_XC_SET_TIN_TOD

◇ Contents:

– NetU16 => Target_site – NetU16 => Target_host – NetFlt32 => Time of Day

◇ Meaning: call `diguyScenario::set_tin_time_of_day()`

DIGUY_CUSTOM_PDU_ID_XC_TIME_NOW

◇ Contents:

– NetFlt32 => seconds since midnight – NetFlt32 => seconds to smooth

◇ Meaning: call `diguyScenario::set_sync_info()`

DIGUY_CUSTOM_PDU_ID_XC_CONTROL

◇ Contents:

– NetU16 => Target_site – NetU16 => Target_host – String => Command ("RUN", "STOP", "RESET", "FF", "REWIND")

DIGUY_CUSTOM_PDU_ID_XC_TRIGGER_SCRIPT

◇ Contents:

– NetU16 => Target_site – NetU16 => Target_host – String Script name

DIGUY_CUSTOM_PDU_ID_XC_EVAL_SCRIPT

◇ Contents:

– NetU16 => Target_site – NetU16 => Target_host – String => Script text

DIGUY_CUSTOM_PDU_ID_XC_SET_FF_SPEED

◇ Contents:

– NetU16 => Target_site – NetU16 => Target_host – NetFlt32 => Speed multiple

DIGUY_CUSTOM_PDU_ID_XC_TRIGGER_SIGNAL

◇ Contents:

– NetU16 => Target_site – NetU16 => Target_host – String => Signal to trigger

DIGUY_CUSTOM_PDU_ID_XC_RESET_SCENARIO

◇ Contents:

– NetU16 => Target_site – NetU16 => Target_host

DIGUY_CUSTOM_PDU_ID_XC_ENTER_SLAVE_MODE

◇ Contents:

– NetU16 => Target_site – NetU16 => Target_host

DIGUY_CUSTOM_PDU_ID_XC_EXIT_SLAVE_MODE

◇ Contents:

– NetU16 => Target_site – NetU16 => Target_host

DIGUY_CUSTOM_PDU_ID_VARIABLE_STATE

◇ Contents:

– NetU16 => Target_site – NetU16 => Target_host – String => variable – String => value

DIGUY_CUSTOM_PDU_ID_SAVE_SCENARIO

◇ Contents:

– NetU16 => Target_site – NetU16 => Target_host – NetU16 => use_user_name – String => file name

DIGUY_CUSTOM_PDU_ID_SAVE_SCENARIO_COMPLETE

◇ Contents:

– String => file name

DIGUY_CUSTOM_PDU_ID_AI_SEND_MESSAGE_TO_CHARACTER

◇ Contents:

– NetU16 => Target_site – NetU16 => Target_host – NetU16 => target AI network number – String => sender – String => message_type – String => message – String => message_params

◇ Meaning: call

```
diguyCharacter::agent_accept_message(sender,
message_type,
message,
message_params);
```

DIGUY_CUSTOM_PDU_ID_AI_SEND_MESSAGE_TO_GROUP

◇ Contents:

– String => group_name – String => sender – String => message_type – String => message – String => message_params

◇ Meaning: call

```
diguyCharacterGroup::send_message_to_all_members(sender,
message_type,
message,
message_params);
```

DIGUY_CUSTOM_PDU_ID_AI_BROADCAST_MESSAGE

◇ Contents:

– String => sender – NetFlt32 => radius – String => message_type – String => message – String => message_params

◇ Meaning: call

```
diguyCharacter::agent_broadcast_message(sender,
radius,
message_type,
message,
message_params);
```

Where character is the sender of message.

DIGUY_CUSTOM_PDU_ID_AI_BROADCAST_MESSAGE_TO_GROUP

◇ Contents:

– String => sender – String => group_name – NetFlt32 => radius – String => message_type – String => message – String => message_params

◇ Meaning: call

```
diguyCharacter::agent_broadcast_message_to_group(group_name,
radius,
message_type,
message,
message_params);
```

Where character is the sender of message.

DIGUY_CUSTOM_PDU_ID_BEGIN_APPEARANCE_EFFECT

◇ Contents:

– String => appearance effect – bool => flag if theres extended data

◇ Optional extended data:

– NetFlt32 => override scale – NetFlt32[3] => override_offset x, y, z –
NetFlt32 => override duration

◇ Meaning: call

```
diguyCharacter::begin_appearance_effect(app_name, link_name, override_scale,
override_offset_x, override_offset_y, override_offset_z,
override_duration,
```

or

```
diguyCharacter::begin_appearance_effect(app_name);
```

DIGUY_CUSTOM_PDU_ID_END_APPEARANCE_EFFECT (Gradual Stop)

◇ Contents:

◇ String => appearance_effect_name

– String => link_name

◇ Meaning: call

```
diguyCharacter::end_appearance_effect(appearance_effect_name, link_name)
```

DIGUY_CUSTOM_PDU_ID_STOP_APPEARANCE_EFFECT (Instant Stop)

◇ Contents:

◇ String => appearance_effect_name

– String => link_name

◇ Meaning: call

`diguyCharacter::stop_appearance_effect(appearance_effect_name, link_name)`

DIGUY_CUSTOM_PDU_ID_SET_SCENE_OBJECT_BIT

- ◇ Contents:
- ◇ bool => true false
- ◇ Meaning: call

`diguyCharacter::set_is_scene_object(value)`

*DIGUY_CUSTOM_PDU_ID_BROADCAST_EXERCISE_DATETIME***internal use only**

- ◇ Contents:

– NetU16 => Target_site – NetU16 => Target_host – NetU16[7] => year, month, day, hour, minute, second, msec

- ◇ Meaning: call

```
bdiScenario::set_network_datetime(network_datetime);
```

DIGUY_CUSTOM_PDU_ID_REQUEST_EXERCISE_DATETIME_BROADCAST

Unused

DIGUY_CUSTOM_PDU_ID_SET_SHAPE_VARIATION

- ◇ Contents:
- ◇ complicated: a series of NetU16 (link index), NetU16 (shape index) followed by a slightly compressed chunk of pixel data. should probably only happen once per character.

```
int offset = 0;
short link_index = 0;
while(pdu->pullShort(link_index, offset) == 0)
{
    short shape_index = 0;
    pdu->pullShort(shape_index, offset);
    bdiLink * link = character->get_link_at_index(link_index);
    if(link)
    {
        bdiShape * shape = link->get_shape(shape_index);
        if(!shape->get_unique_ramp_data())
        {
            const unsigned char * data_blob = NULL;
            int bytes_read;
            if(pdu->pullUnsignedData(data_blob, offset, bytes_read) == 0)
            {
                unsigned char * data_blob_copy = new unsigned char [shape->get_uniq
                int q;
                for(q = 0; q < bytes_read/3; q++)
```

```

        {
            data_blob_copy[q*4] = data_blob[q*3];
            data_blob_copy[q*4+1] = data_blob[q*3+1];
            data_blob_copy[q*4+2] = data_blob[q*3+2];
            data_blob_copy[q*4+3] = 255;;
        }
        // fill the rest of the buffer
        for( ; q < shape->get_unique_ramp_data_size(); q++)
        {
            data_blob_copy[q*4] = 128;
            data_blob_copy[q*4+1] = 128;
            data_blob_copy[q*4+2] = 128;
            data_blob_copy[q*4+3] = 255;;
        }
        shape->set_unique_ramp_data(data_blob_copy);
    }
}

```

DIGUY_CUSTOM_PDU_ID_EXECUTE_GESTURE

◊ Contents:

– String => gesture_name – Long => reps – NetFlt32 => overall_duration – NetFlt32 => channel_A_weight

◊ Meaning: call

`diguyCharacter::execute_gesture`(gesture_name, reps, overall_duration, channel_A_weight);

DIGUY_CUSTOM_PDU_ID_CHARACTER_SCALE

◊ Contents:

– NetFlt32[3] => scale x,y,z

◊ Meaning: call `diguyCharacter::set_scale`(x,y,z);

*DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED0**DIGUY_CUSTOM_PDU_ID_PLUGIN_RESERVED9*

5.102 DI-Guy API, Info Graphics Module

diguy_graphics_api.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_api.h](#) | [diguy_typedefs.h](#) | [bdiGraphicsInitGraphicsAPI.h](#) | [declspec_diguy_graphics.h](#)

Contents:

- ◆ [Graphics API Initialization Functions](#)
- ◆ [DI-Guy Graphics Object Create Functions](#)

Link against: **libdiguy_graphics_api**

Graphics API Initialization Functions

\ingroup diguyGraphicsAPI

5.102.1 function `diguy_graphics_initialize`

Prototype:

```
diguy_graphics_initialize(bdiGraphicsInitGraphicsAPI* init);
```

Description:

This function initializes the DI-Guy Runtime API for accessing information about DI-Guy graphics and geometry.

Use `diguy_graphics_get_defaults()` to fill out the init structure before changing specific entries.

Arguments:

<code>init</code>	pointer to type <code>bdiGraphicsInitGraphicsAPI</code> ; passing NULL will cause defaults to be used for all values

Returns:

0 on success, non-zero error code on failure

Refer to `diguy_constants.h` for error code values.

`\ingroup diguyGraphicsAPI`

5.102.2 function `diguy_graphics_get_defaults`

Prototype:

```
diguy_graphics_get_defaults(bdiGraphicsInitGraphicsAPI* init);
```

Description:

Fills out the init structure with default values. Selected entries can then be changed before calling `diguy_graphics_initialize()`.

The default values are:

```

◇ init->use_array_format_for_vertex_buffer_data = 0;
◇ init->default_vertex_type =
    DIGUY_GRAPHICS_INDEX_TYPE_USHORT;
◇ init->use_shaders = 1;
◇ init->shader_language =
    DIGUY_GRAPHICS_SHADER_LANGUAGE_GLSL;
◇ init->transpose_shader_matrices = 0;
◇ init->use_4x3_shader_matrices = 1;
◇ init->use_cached_optimized_geometry = 1;
◇ init->use_cached_compressed_textures = 1;
◇ init->use_sRGB_corrected_textures = 1;
◇ init->characters_have_world_space_triangle_query = 0;
◇ init->scene_objects_have_world_space_triangle_query = 0;
◇ init->use_linear_traversal_draw = 1;
```

Arguments:

<code>init</code>	pointer to type <code>bdiGraphicsInitGraphicsAPI</code>

Returns:

0 on success, -1 on failure

5.102.3 function `diguy_graphics_get_osg_defaults`

Prototype:

```
diguy_graphics_get_osg_defaults(bdiGraphicsInitGraphicsAPI* init);
```

Description:

Fills out the init structure with default OSG values. Selected entries can then be changed before calling `diguy_graphics_initialize()`.

The differences for the default values are:

```
◇ init->transpose_shader_matrices = 1;
◇ init->use_4x3_shader_matrices = 0;
◇ init->use_sRGB_corrected_textures = 0;
```

Arguments:

`init` pointer to type `bdiGraphicsInitGraphicsAPI`

Returns:

0 on success, -1 on failure

5.102.4 function `diguy_graphics_get_dx9_defaults`

Prototype:

```
diguy_graphics_get_dx9_defaults(bdiGraphicsInitGraphicsAPI* init);
```

Description:

Fills out the init structure with default values. Selected entries can then be changed before calling `diguy_graphics_initialize()`.

The differences for the default values are:

```
◇ init->transpose_shader_matrices = 1
◇ init->use_4x3_shader_matrices = 0
◇ init->shader_language =
    DIGUY_GRAPHICS_SHADER_LANGUAGE_HLSL
```

Arguments:

`init` pointer to type `bdiGraphicsInitGraphicsAPI`

Returns:

0 on success, -1 on failure

5.102.5 function `diguy_graphics_set_use_far_positions`

Prototype:

```
void diguy_graphics_set_use_far_positions(int flag);
```

Description:

This function turns far position rendering on or off. It must be called after `diguy_graphics_initialize()`.

Arguments:

<code>flag</code>	pass 1 to turn far positions on, 0 to turn them off

5.102.6 `diguy_graphics_get_use_far_positions`

Prototype:

```
int diguy_graphics_get_use_far_positions();
```

Description:

This function gets the status of far position rendering. It can only be called after `diguy_graphics_initialize()`.

Returns:

1 if far position rendering is on, 0 if not

DI-Guy Graphics Object Create Functions

`\ingroup diguyGraphicsAPI`

Function responsible for allowing end users to subclass from a `diguyGraphicsLink`

`\ingroup diguyGraphicsAPI`

5.102.7 function `diguy_graphics_set_link_create_func`

Prototype:

```
diguy_graphics_set_link_create_func(diguyGraphicsLinkCreateFunc* func);
```

Function responsible for allowing end users to subclass from a [diguyGraphicsShape](#)

5.102.8 function `diguy_graphics_set_shape_create_func`

Prototype:

```
diguy_graphics_set_shape_create_func(diguyGraphicsShapeCreateFunc* func);
```

Function responsible for allowing end users to subclass from a [diguyGraphicsFile](#)

5.102.9 function diguy_graphics_set_file_create_func

Prototype:

```
diguy_graphics_set_file_create_func(diguyGraphicsFileCreateFunc* func);
```

Function responsible for allowing end users to subclass from a [diguyGraphicsState](#)

5.102.10 function diguy_graphics_set_state_create_func

Prototype:

```
diguy_graphics_set_state_create_func(diguyGraphicsStateCreateFunc* func);
```

Function responsible for allowing end users to subclass from a [diguyGraphicsMaterial](#)

5.102.11 function `diguy_graphics_set_material_create_func`

Prototype:

```
diguy_graphics_set_material_create_func(diguyGraphicsMaterialCreateFunc* func);
```

Function responsible for allowing end users to subclass from a `diguyGraphicsTexture`

5.102.12 function `diguy_graphics_set_texture_create_func`

Prototype:

```
diguy_graphics_set_texture_create_func(diguyGraphicsTextureCreateFunc* func);
```

Function responsible for allowing end users to subclass from a `diguyGraphicsMesh`

5.102.13 function `diguy_graphics_set_mesh_create_func`

Prototype:

```
diguy_graphics_set_mesh_create_func(diguyGraphicsMeshCreateFunc* func);
```

Function responsible for allowing end users to subclass from a `diguyGraphicsShaderProgram`

5.102.14 function `diguy_graphics_set_shader_program_create_func`

Prototype:

```
diguy_graphics_set_shader_program_create_func(diguyGraphicsShaderProgramCreateFunc*
```

Function responsible for allowing end users to subclass from a `diguyGraphicsShaderTechnique`

5.102.15 function**diguy_graphics_set_shader_technique_create_func****Prototype:**

```
diguy_graphics_set_shader_technique_create_func(diguyGraphicsShaderTechniqueCreateFunc)
```

Function responsible for allowing end users to subclass from a [diguyGraphicsShaderInstance](#)

5.102.16 function**diguy_graphics_set_shader_instance_create_func****Prototype:**

```
diguy_graphics_set_shader_instance_create_func(diguyGraphicsShaderInstanceCreateFunc)
```

Function responsible for allowing end users to subclass from a [diguyViewPainter](#)

5.102.17 function diguy_graphics_setPainter_create_func

Prototype:

```
diguy_graphics_setPainter_create_func(diguyViewPainterCreateFunc* func);
```

5.103 DI-Guy API, OpenGL Graphics Module

diguy_graphics_ogl.h

DI-Guy SDK Version 12.5.2

Includes: [diguy_api.h](#) | [graphics_init_ogl.h](#) | [declspec_diguy_graphics.h](#)

Contents:

- ◆ [Functions](#)
- ◆ [Experimental Functions](#)
- ◆ [Experimental Heads-Up Display Functions](#)
- ◆ [Experimental frame stats API](#)

Link against: `libdiguy_graphics_ogl`

Functions

5.103.1 function diguy_ogl_initialize

Prototype:

```
diguy_ogl_initialize(bdiGraphicsInitOgl* init);
```

Description:

This function initializes the DI-Guy Runtime API for the OpenGL environment.

This function should only be called once per program execution.

Do not call this function again, even after calling

diguy_deinitialize().

Use [diguy_ogl_get_defaults\(\)](#) to fill out the init structure before changing specific entries.

Arguments:

<code>init</code>	pointer to type bdiGraphicsInitOgl ; passing NULL will cause defaults to be used for all values

Returns:

0 on success, non-zero error code on failure

Refer to `diguy_constants.h` for error code values.

5.103.2 function `diguy_ogl_get_defaults`

Prototype:

```
diguy_ogl_get_defaults(bdiGraphicsInitOgl* init);
```

Description:

Fills out the init structure with default values. Selected entries can then be changed before calling `diguy_ogl_initialize()`.

Arguments:

<code>init</code>	pointer to type <code>bdiGraphicsInitOgl</code>

Returns:

0 on success, -1 on failure

5.103.3 function `diguy_default_fire_weapon_intersection_function`

Prototype:

```
int diguy_default_fire_weapon_intersection_function(diguyImpact* impact,
float from_x,
float from_y,
float from_z,
float to_x,
float to_y,
float to_z,
diguyScenario* scenario);
```

Description:

This function is the default fire intersection function. It uses a minimal render of the scene and characters into the back buffer to fill out the impact structure.

It may be useful to combine your intersection results with the results from this function.

Experimental Functions

The following functions are experimental and may change or disappear with no warning.

5.103.4 function `diguy_ogl_generate_character_texture_list`

Prototype:

```
int diguy_ogl_generate_character_texture_list(diguyScenario* scenario,
int character_index);
```

Description:

This function queries the character for the OpenGL texture objects it is using and stores them in an internal list. This internal list can then be accessed via the following functions:

```
diguy_ogl_get_character_texture_list_num_textures();
diguy_ogl_get_character_texture_list_texture_object_at_index();
diguy_ogl_get_character_texture_list_texture_filename_at_index();
```

The function `diguy_ogl_release_character_texture_list()` should be called once query calls are complete to avoid a resource leak.

Note that there is currently only one internal texture list; there are not separate lists per character. Calling this function again, even with a different scenario pointer and/or character index, will clobber the previous texture list.

Arguments:

scenario pointer to character's scenario
 character_index index of character to be queried

Returns:

0 on success, -1 on failure

C++ Example:

```
diguy_ogl_generate_character_texture_list(scenario,
    character_index);

int texture_count = diguy_ogl_get_character_texture_list_num_textures(scenario,
    character_index);

if (texture_count > 0)
{
    int texture_index;

    for (texture_index = 0;
        texture_index < texture_count;
        texture_index++)
    {
        GLuint texture_object = diguy_ogl_get_character_texture_list_texture_object(
            character_index,
            texture_index);

        const char* texture_filename = diguy_ogl_get_character_texture_list_texture_filename(
            character_index,
            texture_index);

        bdi_log_printf(BDI_LOG_WARN, "\\ttexture object = %ld, filename = %s\\n",
            texture_object,
            texture_filename);
    }

    GLuint texture_object = diguy_ogl_get_character_texture_list_texture_object_at(
        character_index,
        0); // the first texture in the list

    //
    // (Code to override/replace the texture)
    //

    diguy_ogl_set_character_texture_objects_modified(scenario,
        character_index);

    diguy_ogl_release_character_texture_list(scenario,
        character_index);
}
```

5.103.5 function `diguy_ogl_release_character_texture_list`

Prototype:

```
int diguy_ogl_release_character_texture_list(diguyScenario* scenario,
int character_index);
```

Description:

This function releases a texture list generated by `diguy_ogl_generate_character_texture_list()`.

Arguments:

`scenario` pointer to character's scenario
`character_index` index of character to be queried

Returns:

0 on success, -1 on failure

5.103.6 function `diguy_ogl_get_character_texture_list_num_textures`

Prototype:

```
int diguy_ogl_get_character_texture_list_num_textures(diguyScenario* scenario,
int character_index);
```

Description:

This function returns the number of textures used by the specified character.

The function `diguy_ogl_generate_character_texture_list()` must be called before this function. The `scenario` and `character_index` arguments passed to this function must match those passed to the generate function.

Arguments:

`scenario` pointer to character's scenario
`character_index` index of character to be queried

Returns:

number of textures in texture list

5.103.7 function**diguy_ogl_get_character_texture_list_texture_object_at_index****Prototype:**

```
GLuint diguy_ogl_get_character_texture_list_texture_object_at_index(diguyScenario* s,
    int character_index,
    int texture_list_index);
```

Description:

This function returns the OpenGL texture object at the specified index.

The function

[diguy_ogl_generate_character_texture_list\(\)](#) must be called before this function. The scenario and character_index arguments passed to this function must match those passed to the generate function.

Arguments:

scenario	pointer to character's scenario
character_index	index of character to be queried

`texture_list_index` index of texture in texture list

Returns:

OpenGL texture object

5.103.8 function**`diguy_ogl_get_character_texture_list_texture_filename_at_index`****Prototype:**

```
const char* diguy_ogl_get_character_texture_list_texture_filename_at_index(diguyScenario
    int character_index,
    int texture_list_index);
```

Description:

This function returns the filename of the texture at the specified index.

The function

`diguy_ogl_generate_character_texture_list()` must be called before this function. The `scenario` and `character_index` arguments passed to this function must match those passed to the generate function.

Arguments:

<code>scenario</code>	pointer to character's scenario
<code>character_index</code>	index of character to be queried

`texture_list_index` index of texture in texture list

Returns:

filename of the texture

5.103.9 function

`diguy_ogl_set_character_texture_objects_modified`

Prototype:

```
int diguy_ogl_set_character_texture_objects_modified(diguyScenario* scenario,
int character_index);
```

Description:

This function notifies the character that its texture objects have been modified and that it should take any steps necessary to update its internal state to ensure proper display.

Arguments:

`scenario` pointer to character's scenario
`character_index` index of character to be queried

Returns:

0 on success, -1 on failure

5.103.10 function `diguy_ogl_get_texture_unit`

Prototype:

```
int diguy_ogl_get_texture_unit();
```

Description:

This function returns the texture unit that DI-Guy is currently set to use.

Returns:

integer identifying texture unit; e.g., 0 for texture unit 0

5.103.11 function `diguy_ogl_set_texture_unit`

Prototype:

```
int diguy_ogl_set_texture_unit(int texture_unit);
```

Description:

This function sets the texture unit that DI-Guy should use.

Arguments:

`texture_unit` integer identifying texture unit

Possible values are:

0	use GL_TEXTURE0
1	use GL_TEXTURE1
2	use GL_TEXTURE2
3	use GL_TEXTURE3

Returns:

0 on success, -1 on failure

5.103.12 function `diguy_ogl_set_use_far_positions`

Prototype:

```
void diguy_ogl_set_use_far_positions(int val);
```

Description:

This function turns far position rendering on or off. See [diguyViewCamera](#) for more information on far positions.

Arguments:

`val` pass 1 to use far positions, 0 to not use them

5.103.13 function diguy_ogl_get_use_far_positions**Prototype:**

```
int diguy_ogl_get_use_far_positions();
```

Returns:

1 if far positions are in use, else 0

5.103.14 function diguy_ogl_set_default_shader_source_files**Prototype:**

```
void diguy_ogl_set_default_shader_source_files(const char* shader_name,
const char* vert_filename,
const char* pixel_filename);
```

Description:

This function sets the shader source files that will be used for the specified shader.

The default filenames for the "diguy_character" shader are "diguy_character_glsl.vert" and "diguy_character_glsl.frag".

Arguments:

<code>shader_name</code>	name with which vertex/pixel shader pair can be referenced
<code>vert_filename</code>	name of vertex shader source file
<code>pixel_filename</code>	name of pixel (fragment) shader source file

5.103.15 function diguy_ogl_render_shadow_map**Prototype:**

```
int diguy_ogl_render_shadow_map(unsigned int texture_unit, // 0 to 7, not GL_TEXTURE0
                                diguyViewLight* light,
                                diguyScenario* scenario,
                                float pos_above_terrain);
```

Description:

Renders a shadow pass of the world and binds into the target texture unit.

5.103.16 function diguy_ogl_apply_activeLights**Prototype:**

```
int diguy_ogl_apply_activeLights(diguyScenario* scenario);
```

Description:

Applies active lights to the OpenGL state, also updates default shader uniforms and calls
`diguyScenario::set_num_active_point_lights()`

Returns:

The number of active lights

Experimental Heads-Up Display Functions

The following functions are experimental and may change or disappear with no warning.

5.103.17 function `diguy_ogl_2d_set_color`

Prototype:

```
int diguy_ogl_2d_set_color(float r, float g, float b, float a);
```


5.103.18 function diguy_ogl_2d_draw_filled_rect**Prototype:**

```
int diguy_ogl_2d_draw_filled_rect(int x0, int y0, int x1, int y1);
```

5.103.19 function diguy_ogl_2d_draw_text**Prototype:**

```
int diguy_ogl_2d_draw_text(int x, int y, const char* text);
```

5.103.20 function diguy_ogl_2d_get_text_width**Prototype:**

```
int diguy_ogl_2d_get_text_width(const char* text);
```

5.103.21 function diguy_ogl_2d_get_text_height**Prototype:**

```
int diguy_ogl_2d_get_text_height(const char* text);
```

5.103.22 function diguy_ogl_get_fbo_texture_id**Prototype:**

```
int diguy_ogl_get_fbo_texture_id();
```

5.103.23 function diguy_ogl_reload_changed_textures**Prototype:**

```
int diguy_ogl_reload_changed_textures();
```

Experimental frame stats API

Currently available stats:

- ◇ "characters_drawn"
- ◇ "links_drawn"
- ◇ "shapes_drawn"
- ◇ "nodes_drawn"
- ◇ "mats_bound"
- ◇ "groups_drawn"
- ◇ "textures_bound"
- ◇ "vbos_drawn"
- ◇ "rigid_vbos_drawn",
- ◇ "skinned_vbos_drawn"
- ◇ "tris_drawn"
- ◇ "verts_drawn"

Stats should be cleared once a frame to get accurate results.

5.103.24 function `diguy_ogl_get_num_frame_stats`

Prototype:

```
int diguy_ogl_get_num_frame_stats();
```

5.103.25 function diguy_ogl_frame_stat_name**Prototype:**

```
const char* diguy_ogl_frame_stat_name(int index);
```

5.103.26 function diguy_ogl_get_frame_stat**Prototype:**

```
int diguy_ogl_get_frame_stat(const char* stat_name);
```

5.103.27 function `diguy_ogl_clear_all_frame_stats`

Prototype:

```
int diguy_ogl_clear_all_frame_stats();
```

5.103.28 function `diguy_ogl_draw_frame_stats`

Prototype:

```
int diguy_ogl_draw_frame_stats(int sx, int sy);
```

5.104 DI-Guy API, VR-Link DIS Module

diguy_module_dis.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

◆ [Functions](#)

Link against: libdiguy_module_dis

Deprecated as of DI-Guy 9.0.0; please move to the interfaces in diguy_module_net.h.

Functions

5.104.1 function diguy_dis_set_cfg_file

Prototype:

```
int diguy_dis_set_cfg_file(const char* cfg_file);
```

Description:

This function sets the filename of the VR-Link configuration file that will be read by the DI-Guy software. The default is "network.cfg". This function must be called before [diguy_dis_initialize\(\)](#).

Arguments:

`cfg_file` filename of desired configuration file

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.104.2 function diguy_dis_initialize

Prototype:

```
int diguy_dis_initialize();
```

Description:

This function allows VR-Link to install some callbacks so that .dss file load and save will include VR-Link-specific info.

It should be called BEFORE the scenario object is constructed.

Returns:

0 on success, -1 on failure

Example:

```
diguy_dis_initialize();
```

Callable From:

◇ C++

5.104.3 function diguy_dis_deinitialize

Prototype:

```
int diguy_dis_deinitialize();
```

Description:

This function allows DI-Guy networking to clean up.

It should be called once, on application shutdown, after the scenario object has been destroyed.

Returns:

0 on success, -1 on failure

Example:

```
diguy_dis_deinitialize();
```

Callable From:

◇ C++

5.104.4 function `diguy_dis_go_online`

Prototype:

```
int diguy_dis_go_online(diguyScenario* scenario, void* exconn = NULL);
```

Description:

This function tells DI-Guy that it should start talking over VR-Link.

It should be called when DI-Guy should join an exercise.

Arguments:

<code>scenario</code>	pointer to the <code>diguyScenario</code> which will be connected to the network
<code>exconn</code>	optional pointer to an existing VR-Link exercise connection to use

Returns:

0 on success, -1 on failure

Example:

```
diguy_dis_go_online(the_scenario_app);
```

Callable From:

- ◇ C++
- ◇ Script

5.104.5 function diguy_dis_update

Prototype:

```
int diguy_dis_update();
```

Description:

This function tells DI-Guy that it should read PDUs from the network and process them. Call this function periodically.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.104.6 function diguy_dis_get_use_user_defined_coords

Prototype:

```
int diguy_dis_get_use_user_defined_coords();
```

Description:

This function returns whether DI-Guy is using user-defined latitude/longitude coordinates for the origin of the database, or the coordinates that come from graphics files for scenario scene objects.

Returns:

0 if graphics file coordinates are used; 1 if user-defined coordinates are used

Callable From:

- ◇ C++
- ◇ Script

5.104.7 function diguy_dis_go_offline

Prototype:

```
int diguy_dis_go_offline();
```

Description:

This function tells DI-Guy that it should stop networking with VR-Link.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.104.8 function diguy_dis_get_online

Prototype:

```
int diguy_dis_get_online();
```

Description:

This function queries whether DI-Guy is talking over VR-Link.

Returns:

1 if yes, 0 if no

Callable From:

- ◇ C++
- ◇ Script

5.104.9 function diguy_dis_get_exercise

Prototype:

```
int diguy_dis_get_exercise();
```

Returns:

the exercise id being used by DI-Guy Networking

Callable From:

- ◇ C++
- ◇ Script

5.104.10 function diguy_dis_get_host

Prototype:

```
int diguy_dis_get_host();
```

Returns:

the host id being used by DI-Guy Networking

Callable From:

- ◇ C++
- ◇ Script

5.104.11 function diguy_dis_get_site

Prototype:

```
int diguy_dis_get_site();
```

Returns:

the site id being used by DI-Guy Networking

Callable From:

- ◇ C++
- ◇ Script

5.104.12 function diguy_dis_get_port

Prototype:

```
int diguy_dis_get_port();
```

Description:

the port number being used for DI-Guy Networking

Callable From:

- ◇ C++
- ◇ Script

5.104.13 function diguy_dis_get_lat_lon

Prototype:

```
int diguy_dis_get_lat_lon(double* lat, double* lon);
```

Description:

Gets the latitude and longitude currently being used by DI-Guy networking for the placement of the database origin.

Arguments:

lat	latitude of origin, returned; see below
lon	longitude of origin, returned; see below

lat and lon are char**'s, into which char*'s will be placed. These char*'s point to static data owned by DI-Guy. Do not store these pointers, and do not modify what they point to. Copy the strings needed out of the static buffers. (Assignment of a char* to a Microsoft CString will work very nicely, for example.)

lat and lon are expressed as strings of the form:

`[-]d [N|W|S|E] or [-]d m [N|W|S|E] or [-]d m s [N|W|S|E]`

N and E are taken as positive. S and W will negate the value otherwise computed. d m and s are positive, floating point values, taken as degrees, minutes, and seconds, respectively. A leading minus sign will negate the entire value otherwise computed (so, e.g., a '-' and a 'S' will result in a double-negation).

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.104.14 function diguy_dis_get_utm

Prototype:

```
int diguy_dis_get_utm(int* zone,
    int* southzone,
    double* easting,
    double* northing);
```

Description:

Gets, in UTM coordinates, the location currently being used by DI-Guy Networking for the placement of the database origin.

Arguments:

zone	the UTM zone, returned
southzone	return value; 1 if the origin is in the southern hemisphere, else 0
easting	return value; normalized (positive only) easting of the database origin within the zone
northing	

return value; normalized (positive only) northing of the database origin within the zone, returned

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

◇ Script

5.104.15 function diguy_dis_set_utm

Prototype:

```
int diguy_dis_set_utm(int zone,
    int southzone,
    double easting,
    double northing);
```

Description:

Sets, in UTM coordinates, the location currently being used by DI-Guy Networking for the placement of the database origin.

Arguments:

zone	the UTM zone
southzone	1 if the origin is in the southern hemisphere, else 0
easting	the normalized (positive only) easting of the database origin within the zone
northing	the normalized (positive only) northing of the database origin within the zone

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.104.16 function diguy_dis_set_utm_zone

Prototype:

```
int diguy_dis_set_utm_zone(int zone, int southzone);
```

Description:

Sets the UTM zone of the location currently being used by DI-Guy Networking for the placement of the database origin. The easting and northing are assumed not to change.

Arguments:

zone	the utm zone
southzone	1 if the origin is in the southern hemisphere, else 0

Returns:

0 if successful

Callable From:

◇ C++

◇ Script

5.104.17 function diguy_parse_dms_to_radians

Prototype:

```
double diguy_parse_dms_to_radians(const char* dmsstring);
```

Description:

Parses a string of the form

`[-]d [N|W|S|E] or [-]d m [N|W|S|E] or [-]d m s [N|W|S|E]`

N and E are taken as positive. S and W will negate the value otherwise computed. d m and s are positive, floating point values, taken as degrees, minutes, and seconds, respectively. A leading minus sign will negate the entire value otherwise computed (so, e.g., a '-' and a 'S' will result in a double-negation).

Arguments:

`dmsstring` a string of the format described above

Returns:

a double precision floating point value in radians

Callable From:

- ◇ C++
- ◇ Script

5.104.18 function diguy_format_lat_or_lon

Prototype:

```
void diguy_format_lat_or_lon(char* outbuf,
    double value,
    bool is_lat,
    bool is_decimal,
    bool value_is_degrees);
```

Description:

Returns a string of the form

d [N|W|S|E] or d m s [N|W|S|E]

N and E are used for positive angles. S and W are used for negative angles. In the d m s format, d and m are integers (degrees and minutes) and s (seconds) is a floating point value. In the d format, d (degrees) is a floating point value.

Arguments:

outbuf a buffer into which to copy the resulting string

<code>value</code>	the lat or lon value to be converted into a string
<code>is_lat</code>	boolean indicating latitude (North/South) rather than longitude (East/West)
<code>is_decimal</code>	boolean indicating that decimal degrees are desired, rather than degrees, minutes, and seconds
<code>value_is_degrees</code>	boolean indicating whether the input value is in degrees (as opposed to radians)

Callable From:

- ◇ C++
- ◇ Script

5.104.19 function `diguy_dis_get_ipaddr`

Prototype:

```
int diguy_dis_get_ipaddr(char** ipaddr);
```

Description:

Gets the IP address currently being used by DI-Guy networking. This address is the address to which messages are sent. Normally, it is some kind of broadcast or multicast address.

Arguments:

`ipaddr` IP address, returned; see below

`ipaddr` is a `char**`, into which a `char*` will be placed. This `char*` points to static data owned by DI-Guy. Do not store this pointer, and do not modify what it points to. Copy the strings you need out of the static buffer. (Assignment of a `char*` to a Microsoft CString will work very nicely, for example.)

ipaddr is a string containing 4 integers separated by periods.

Returns:

0 if successful

Callable From:

◇ C++
◇ Script

5.104.20 function diguy_dis_get_publish_local_chars

Prototype:

```
int diguy_dis_get_publish_local_chars();
```

Description:

Returns whether DI-Guy Networking will publish local characters when online.

C++, Script

Arguments:

none

Returns:

1 if enabled, 0 if not

5.104.21 function diguy_dis_get_display_remote_chars**Prototype:**

```
int diguy_dis_get_display_remote_chars();
```

Description:

Returns whether DI-Guy Networking will display remote characters when online.

C++, Script

Arguments:

none

Returns:

1 if enabled, 0 if not

5.104.22 function diguy_dis_set_exercise**Prototype:**

```
int diguy_dis_set_exercise(int exercise);
```

Description:

Sets the exercise id to be used by DI-Guy Networking.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, Script

Arguments:

`exercise` the exercise id to use

Returns:

0 on success, -1 on failure

5.104.23 function diguy_dis_set_host

Prototype:

```
int diguy_dis_set_host(int host);
```

Description:

Sets the host id to be used by DI-Guy Networking.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, Script

Arguments:

host	the host id to use; the host id identifies the local machine

Returns:

0 on success, -1 on failure

5.104.24 function diguy_dis_set_site

Prototype:

```
int diguy_dis_set_site(int site);
```

Description:

Sets the site id to be used by DI-Guy Networking.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, Script

Arguments:

site	the site id to use; the site id identifies where the local machine is

Returns:

0 on success, -1 on failure

5.104.25 function `diguy_dis_set_port`

Prototype:

```
int diguy_dis_set_port(int port);
```

Description:

Sets the network port number to be used by DI-Guy Networking.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, Script

Arguments:

<code>port</code>	the port number to use

Returns:

0 on success, -1 on failure

5.104.26 function `diguy_dis_set_lat_lon`

Prototype:

```
int diguy_dis_set_lat_lon(double lat, double lon);
```

Description:

Sets the latitude and longitude to be used by DI-Guy Networking. The latitude and longitude are taken as the location, on Earth, of the origin of the database.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, Script

Arguments:

<code>lat</code>	the latitude in radians, positive north
<code>lon</code>	the longitude in radians, positive east

Returns:

0 on success, -1 on failure

5.104.27 function diguy_dis_set_ipaddr

Prototype:

```
int diguy_dis_set_ipaddr(const char* ipaddr);
```

Description:

Sets the IP address to be used by DI-Guy Networking. This address is the address to which messages are sent. Normally, it is a broadcast or multicast address.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, *Script*

Arguments:

`ipaddr` the IP address

`ipaddr` is a string containing 4 integers separated by periods, or the null string, which means "let DI-Guy choose the address". An address of "0" or

"0.0.0.0" means broadcast.

Returns:

0 on success, -1 on failure

5.104.28 function diguy_dis_set_display_remote_chars

Prototype:

```
int diguy_dis_set_display_remote_chars(int display);
```

Description:

Sets whether DI-Guy Networking will display remote characters.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, *Script*

Arguments:

<code>display</code>	pass 1 if DI-Guy Networking should display remote characters, 0 if not

Returns:

0 on success, -1 on failure

5.104.29 function diguy_dis_set_publish_local_chars**Prototype:**

```
int diguy_dis_set_publish_local_chars(int publish);
```

Description:

Sets whether DI-Guy Networking will publish local characters to the network.

This function WILL succeed even if DI-Guy is on-line (is participating in an exercise).

C++, Script

Arguments:

<code>publish</code>	pass 1 if DI-Guy Networking should publish local characters to the network, 0 if not

Returns:

0 on success, -1 on failure

5.104.30 function diguy_dis_get_position_offset**Prototype:**

```
int diguy_dis_get_position_offset(float* z, float* yaw, float* roll, float* pitch);
```

Description:

This function gets the position offset which is currently added to each character when it is published, and subtracted from each character when it is reflected. The offset is useful to compensate for inter-application database location and orientation error.

Arguments:

<code>z</code>	z offset in meters
<code>yaw</code>	orientation offset in degrees
<code>roll</code>	orientation offset in degrees
<code>pitch</code>	orientation offset in degrees

Returns:

0 on success, -1 on failure

5.104.31 function diguy_dis_set_position_offset

Prototype:

```
int diguy_dis_set_position_offset(float z, float yaw, float roll, float pitch);
```

Description:

This function sets the position offset which is currently added to each character when it is published, and subtracted from each character when it is reflected. This is useful to compensate for inter-application location error.

THIS FUNCTION DOES NOTHING if called while VR-Link is online.

Arguments:

z	z offset in meters
yaw	orientation offset in degrees
roll	orientation offset in degrees
pitch	orientation offset in degrees

Returns:

0 on success, -1 on failure

5.104.32 function diguy_dis_get_sync_mode_enabled

Prototype:

```
int diguy_dis_get_sync_mode_enabled();
```

Description:

Gets whether sync mode is enabled when online.

C++, *Script*

Arguments:

none

Returns:

1 if sync mode is enabled when online, 0 if not

5.104.33 function diguy_dis_get_sync_mode_master**Prototype:**

```
int diguy_dis_get_sync_mode_master();
```

Description:

Gets whether this machine will serve as sync master when online and in sync mode.

C++, Script

Arguments:

none

Returns:

1 if this machine will serve as sync master when online and in sync mode, 0 if not

5.104.34 function diguy_dis_get_sync_mode_slave_cam**Prototype:**

```
int diguy_dis_get_sync_mode_slave_cam();
```

Description:

Gets whether this machine's camera will be slaved to a remote master when online and in sync mode and not sync master.

C++, Script

Arguments:

none

Returns:

1 if this machine's camera will be slaved to a remote master when online and in sync mode and not sync master, 0 if not

5.104.35 function diguy_dis_set_use_database_coords**Prototype:**

```
int diguy_dis_set_use_database_coords(int use_database_coords);
```

Description:

Sets whether networking should use database coordinates or not. If database coordinates are used, the database origin is placed at the center of the earth. This makes coordinate conversion to and from UTM unnecessary. Using database coordinates renders ineffective any UTM-related settings which would position the database.

C++, Script

Arguments:

`use_database_coords` 1 if database coordinates should be used, else 0

Returns:

0 if successful, -1 on failure

5.104.36 function diguy_dis_set_use_user_defined_coords**Prototype:**

```
int diguy_dis_set_use_user_defined_coords(int x);
```

Description:

Sets whether networking should use user-defined coordinates or not. If user-defined coordinates are not used, the database origin is placed at the earth location specified in the graphics files for the database.

C++, Script

Arguments:

`x` pass 1 if user-defined coordinates should be used, else 0

Returns:

1 if this machine's camera will be slaved to a remote master when online and in sync mode and not sync master; 0 if not

5.104.37 function diguy_dis_set_sync_mode_enabled**Prototype:**

```
int diguy_dis_set_sync_mode_enabled(int enabled);
```

Description:

Sets whether sync mode is enabled when online.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, Script

Arguments:

`enabled` pass 1 if sync mode should be enabled when online, 0 if not

Returns:

0 on success, -1 on failure

5.104.38 function diguy_dis_set_sync_mode_master**Prototype:**

```
int diguy_dis_set_sync_mode_master(int master);
```

Description:

Sets whether this machine will serve as sync master when online and in sync mode.

C++, Script

Arguments:

`master` pass 1 if this machine will serve as sync master when online and in sync mode, 0 if not

Returns:

0 on success, -1 on failure

5.104.39 function diguy_dis_set_sync_mode_slave_cam**Prototype:**

```
int diguy_dis_set_sync_mode_slave_cam(int sl_cam);
```

Description:

Sets whether this machine's camera will be slaved to a remote master when online and in sync mode and not sync master.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, *Script*

Arguments:

sl_cam – pass 1 if this machine's camera will be slaved to a remote master when online and in sync mode and not sync master, 0 if not

Returns:

0 on success, -1 on failure

5.104.40 function diguy_dis_set_sync_mode_camera_offset**Prototype:**

```
int diguy_dis_set_sync_mode_camera_offset(float offset_r, float offset_y, float offs
```

Description:

This function sets the camera orientation offset to be used in sync_mode (when VR-Link is online).

Arguments:

offset_r	orientation offset
offset_y	orientation offset
offset_p	orientation offset

yaw and pitch represent PERCENTAGES of the camera's horizontal and vertical field of view, respectively. The ROLL (which is in degrees) is applied first, then the yaw, then the pitch.

A typical two screen arrangement would use yaws of 50 and -50 for the left screen, and right screen, respectively. For three screens, the yaws would be -100, 0, and 100.

Using percentages this way makes the screens contiguous even if the camera's field of view changes.

THIS FUNCTION DOES NOTHING (and returns -1) if called while VR-Link is online

Returns:

0 on success, -1 on failure (VR-Link is online).

5.104.41 function diguy_dis_get_sync_mode_camera_offset

Prototype:

```
int diguy_dis_get_sync_mode_camera_offset(float* offset_r, float* offset_y, float* offset_p)
```

Description:

This function gets the camera orientation offset to be used in sync_mode (when VR-Link is online).

Arguments:

offset_r	orientation offset
offset_y	orientation offset
offset_p	orientation offset

yaw and pitch represent PERCENTAGES of the camera's horizontal and vertical field of view, respectively. The ROLL (which is in degrees) is applied first, then the yaw, then the pitch.

In the orientation offset the yaw and pitch represent PERCENTAGES of the camera's horizontal and vertical field of view, respectively. The ROLL (which is in degrees) is applied first, then the yaw, then the pitch.

A typical two screen arrangement would use yaws of 50 and -50 for the left screen, and right screen, respectively. For three screens, the yaws would be 100, 0, and -100 going from left to right.

Using percentages this way makes the screens contiguous even if the camera's field of view changes.

Returns:

0 on success, -1 on failure

5.104.42 function diguy_dis_set_sync_mode_camera_slaved

Prototype:

```
int diguy_dis_set_sync_mode_camera_slaved(int slaved);
```

Description:

This function sets whether the camera will be slaved when VR-Link is online, in sync mode, and not sync master.

It can be set even when VR-Link is online.

Arguments:

`slaved` whether the camera will be slaved when VR-Link is
online, in sync mode, and not sync master.

Returns:

0 on success, -1 on failure

5.104.43 function diguy_dis_get_sync_mode_camera_slaved**Prototype:**

```
int diguy_dis_get_sync_mode_camera_slaved();
```

Description:

This function gets whether the camera is currently slaved. It does NOT simply return what was passed to `diguy_dis_set_sync_mode_camera_slaved()`.

Returns:

Whether the camera is currently slaved. This will be 1 only if VR-link is online, is in sync mode, and is not the sync master, and if `sync_mode_camera_slaved` has been set to 1.

5.104.44 function diguy_dis_get_next_participant_name**Prototype:**

```
void* diguy_dis_get_next_participant_name(char* buf, void* handle);
```

Description:

This function allows iteration through the list of names remote VR-Link hosts that DI-Guy knows about.

Arguments:

<code>buf</code>	destination buffer for host name (64 bytes is plenty)
<code>handle</code>	return value from last call to the function; pass NULL to get the first host name; pass previous return value to get the next host name

Returns:

`void*` to be passed as `handle` argument on the next call. NULL when end of list is reached.

5.104.45 function diguy_dis_set_camera_details_out_of_date**Prototype:**

```
int diguy_dis_set_camera_details_out_of_date();
```

Description:

Lets VR-Link know that the user has moved the camera, so that if it is in sync mode, and is the sync master, it can let other hosts know the new camera position so they can move their cameras.

Returns:

0 on success, -1 on failure

5.104.46 function diguy_dis_get_model_from_septet**Prototype:**

```
int diguy_dis_get_model_from_septet(const char** chartype, const char** appearance,
```

Description:

Fetches what character type and what appearance should be used for a given DIS entity type septet.

Arguments:

<code>chartype</code>	address of <code>char*</code> , for returning a string which names the DI-Guy character type
<code>appearance</code>	address of <code>char*</code> , for returning a string which names the DI-Guy character appearance
<code>septet</code>	array of seven integers specifying the kind, domain, country, category, subCategory, specific, and extra of the entity in question

Returns:

0 on success, -1 on failure

5.104.47 function diguy_dis_get_septet_from_model

Prototype:

```
int diguy_dis_get_septet_from_model(int septet[7], const char* chartype, const char*
```

Description:

Fetches what DIS entity type septet should be used for a given character type and appearance.

Arguments:

<code>chartype</code>	a string which names the DI-Guy character type
<code>appearance</code>	a string which names the DI-Guy character appearance
<code>septet</code>	array of seven integers, returned, indicating the kind, domain, country, category, subCategory, specific, and extra of the entity in question

Returns:

0 on success, -1 on failure

5.104.48 function diguy_dis_get_earth_coord

Prototype:

```
int diguy_dis_get_earth_coord(char* outbuf, const float x, const float y, const float z)
```

Description:

Converts a 3d database-origin-relative location into a string describing it in earth coordinates. This string could be in a number of different formats including decimal lat/lon, dms lat/lon, and UTM.

Arguments:

outbuf	buffer into which to put the output string
x, y, z	3 floats specifying location

Returns:

0 on success, -1 on failure. Fails if DI-Guy Scenario is not currently on the network, meaning that a database-to-earth coordinate conversion cannot be performed.

5.104.49 function diguy_dis_toggle_earth_coord_format**Prototype:**

```
void diguy_dis_toggle_earth_coord_format();
```

Description:

Changes the format of the output of diguy_dis_get_earth_coord. Toggles between UTM, dms lat/lon, decimal lat/lon, and "punting", meaning that the caller of diguy_dis_get_earth_coord should fend for itself. This generally means that the caller will use database-origin-relative coordinates.

Arguments:

none

Returns:

void

5.104.50 function diguy_dis_get_character_from_id**Prototype:**

```
diguyCharacter* diguy_dis_get_character_from_id(char const* id);
```

Description:

returns the [diguyCharacter](#) associated with the entity id specified.

Arguments:

<code>id</code>	entity id

Under HLA, two formats are permitted: the HLA object name, or the DIS entity identifier ("site:application:entity", where each is an integer).

Under DIS, only the DIS Entity Identifier is permitted.

5.104.51 function diguy_dis_enter_vpnet_mode**Prototype:**

```
void diguy_dis_enter_vpnet_mode();
```

Description:

Turns the 'never create' character bit on.

5.104.52 function diguy_dis_get_name_from_entity**Prototype:**

```
char* diguy_dis_get_name_from_entity(void* reflected_entity);
```

Description:

This function returns the name corresponding to the passed entity.

The user must not delete the returned string.

Arguments:

`reflected_entity` entity to be named

Returns:

entity name

5.104.53 function diguy_dis_establish_new_offsets

Prototype:

```
void diguy_dis_establish_new_offsets(int deprecated = 0);
```

5.104.54 function diguy_dis_set_send_custom_pdus

Prototype:

```
void diguy_dis_set_send_custom_pdus(int val);
```

5.104.55 function diguy_dis_get_send_custom_pdus

Prototype:

```
int diguy_dis_get_send_custom_pdus();
```

5.104.56 function diguy_dis_set_client_mode

Prototype:

```
void diguy_dis_set_client_mode();
```

5.104.57 function diguy_dis_set_server_mode

Prototype:

```
void diguy_dis_set_server_mode();
```

5.105 DI-Guy API, FaceFx Expressive Module

diguy_module_facefx.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h | [bdiFaceFxInit.h](#)

Contents:

◆ [Functions](#)

Link against: libdiguy_module_facefx

Functions

5.105.1 function `diguy_facefx_get_defaults`

Prototype:

```
int diguy_facefx_get_defaults(bdiFaceFxInit* init);
```

Description:

Fills out the init structure with default values. Selected entries can then be changed before calling `diguy_facefx_initialize()`.

The `bdiFaceFxInit` init structure has the following entries:

<code>int</code>	a placeholder value; ignored
placeholder	

Returns:

0 on success, -1 on failure

5.105.2 function `diguy_facefx_initialize`

Prototype:

```
int diguy_facefx_initialize(bdiFaceFxInit* init);
```

Description:

This function tells DI-Guy that it should make use of expressive face software for animating character faces.

It should be called along with the other `diguy_initialize_*` calls.

Arguments:

<code>init</code>	initial settings for the expressive face software; passing NULL for the init structure will cause defaults to be used for all values

Use `diguy_facefx_get_defaults()` to fill out the init structure before changing specific entries.

Returns:

0 on success, -1 on failure

Example:

```
diguy_ogl_initialize(NULL);  
diguy_facefx_initialize(NULL);
```

5.105.3 function diguy_facefx_deinitialize

Prototype:

```
int diguy_facefx_deinitialize();
```

Description:

This function tells DI-Guy that it should shut down the expressive face software extensions. It should be called along with the other diguy_deinitialize_* calls.

Returns:

0 on success, -1 on failure

Example:

```
diguy_deinitialize();  
diguy_facefx_deinitialize();
```


5.106 DI-Guy API, VR-Link HLA Module

diguy_module_hla.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

◆ [Functions](#)

Link against: libdiguy_module_hla

Deprecated as of DI-Guy 9.0.0, please move to the interfaces in diguy_module_net.h.

Functions

5.106.1 function diguy_hla_set_cfg_file

Prototype:

```
int diguy_hla_set_cfg_file(const char* cfg_file);
```

Description:

This function sets the filename of the VR-Link configuration file that will be read by the DI-Guy software. The default is "network.cfg". This function must be called before [diguy_hla_initialize\(\)](#).

Arguments:

`cfg_file` filename of desired configuration file

Returns:

0 on success, -1 on failure

Callable From:

◇ C++

5.106.2 function diguy_hla_initialize

Prototype:

```
int diguy_hla_initialize();
```

Description:

This function allows VR-Link to install some callbacks so that .dss file load and save will include VR-Link-specific info.

It should be called BEFORE the scenario object is constructed.

Returns:

0 on success, -1 on failure

Example:

```
diguy_hla_initialize();
```

Callable From:

◇ C++

5.106.3 function diguy_hla_deinitialize

Prototype:

```
int diguy_hla_deinitialize();
```

Description:

This function allows DI-Guy networking to clean up.

It should be called once, on application shutdown, after the scenario object has been destroyed.

Returns:

0 on success, -1 on failure

Example:

```
diguy_hla_deinitialize();
```

Callable From:

◇ C++

5.106.4 function `diguy_hla_go_online`

Prototype:

```
int diguy_hla_go_online(diguyScenario* scenario, void* exconn = NULL);
```

Description:

This function tells DI-Guy that it should start talking over VR-Link.

It should be called when DI-Guy should join an exercise.

Arguments:

<code>scenario</code>	pointer to the <code>diguyScenario</code> which will be connected to the network
<code>exconn</code>	optional pointer to an existing VR-Link exercise connection to use

Returns:

0 on success, -1 on failure

Example:

```
diguy_hla_go_online(the_scenario_app);
```

Callable From:

- ◇ C++
- ◇ Script

5.106.5 function diguy_hla_update

Prototype:

```
int diguy_hla_update();
```

Description:

This function tells DI-Guy that it should read PDUs from the network and process them. Call this function periodically.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.106.6 function diguy_hla_get_use_user_defined_coords**Prototype:**

```
int diguy_hla_get_use_user_defined_coords();
```

Description:

This function returns whether DI-Guy is using user-defined latitude/longitude coordinates for the origin of the database, or the coordinates that come from graphics files for scenario scene objects.

Returns:

0 if graphics file coordinates are used; 1 if user-defined coordinates are used

Callable From:

- ◇ C++
- ◇ Script

5.106.7 function diguy_hla_go_offline**Prototype:**

```
int diguy_hla_go_offline();
```

Description:

This function tells DI-Guy that it should stop networking with VR-Link.

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.106.8 function diguy_hla_get_online

Prototype:

```
int diguy_hla_get_online();
```

Description:

This function queries whether DI-Guy is talking over VR-Link.

Returns:

1 if yes, 0 if no

Callable From:

- ◇ C++
- ◇ Script

5.106.9 function diguy_hla_get_exercise

Prototype:

```
int diguy_hla_get_exercise();
```

Returns:

the exercise id being used by DI-Guy Networking

Callable From:

- ◇ C++
- ◇ Script

5.106.10 function diguy_hla_get_host

Prototype:

```
int diguy_hla_get_host();
```

Returns:

the host id being used by DI-Guy Networking

Callable From:

- ◇ C++
- ◇ Script

5.106.11 function diguy_hla_get_site

Prototype:

```
int diguy_hla_get_site();
```

Returns:

the site id being used by DI-Guy Networking

Callable From:

- ◇ C++
- ◇ Script

5.106.12 function diguy_hla_get_port

Prototype:

```
int diguy_hla_get_port();
```

Description:

the port number being used for DI-Guy Networking

Callable From:

- ◇ C++
- ◇ Script

5.106.13 function diguy_hla_get_lat_lon

Prototype:

```
int diguy_hla_get_lat_lon(double* lat, double* lon);
```

Description:

Gets the latitude and longitude currently being used by DI-Guy networking for the placement of the database origin.

Arguments:

lat	latitude of origin, returned; see below
lon	longitude of origin, returned; see below

lat and lon are char**'s, into which char*'s will be placed. These char*'s point to static data owned by DI-Guy. Do not store these pointers, and do not modify what they point to. Copy the strings needed out of the static buffers. (Assignment of a char* to a Microsoft CString will work very nicely, for example.)

lat and lon are expressed as strings of the form:

`[-]d [N|W|S|E] or [-]d m [N|W|S|E] or [-]d m s [N|W|S|E]`

N and E are taken as positive. S and W will negate the value otherwise computed. d m and s are positive, floating point values, taken as degrees, minutes, and seconds, respectively. A leading minus sign will negate the entire value otherwise computed (so, e.g., a '-' and a 'S' will result in a double-negation).

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.106.14 function diguy_hla_get_utm

Prototype:

```
int diguy_hla_get_utm(int* zone,
    int* southzone,
    double* easting,
    double* northing);
```

Description:

Gets, in UTM coordinates, the location currently being used by DI-Guy Networking for the placement of the database origin.

Arguments:

zone	the UTM zone, returned
southzone	return value; 1 if the origin is in the southern hemisphere, else 0
easting	return value; normalized (positive only) easting of the database origin within the zone
northing	

return value; normalized (positive only) northing of the database origin within the zone, returned

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.106.15 function diguy_hla_set_utm

Prototype:

```
int diguy_hla_set_utm(int zone,
    int southzone,
    double easting,
    double northing);
```

Description:

Sets, in UTM coordinates, the location currently being used by DI-Guy Networking for the placement of the database origin.

Arguments:

zone	the UTM zone
southzone	1 if the origin is in the southern hemisphere, else 0
easting	the normalized (positive only) easting of the database origin within the zone
northing	the normalized (positive only) northing of the database origin within the zone

Returns:

0 on success, -1 on failure

Callable From:

- ◇ C++
- ◇ Script

5.106.16 function diguy_hla_set_utm_zone

Prototype:

```
int diguy_hla_set_utm_zone(int zone, int southzone);
```

Description:

Sets the UTM zone of the location currently being used by DI-Guy Networking for the placement of the database origin. The easting and northing are assumed not to change.

Arguments:

zone	the utm zone
southzone	1 if the origin is in the southern hemisphere, else 0

Returns:

0 if successful

Callable From:

◇ C++

◇ Script

5.106.17 function `diguy_parse_dms_to_radians`

Prototype:

```
double diguy_parse_dms_to_radians(const char* dmsstring);
```

Description:

Parses a string of the form

`[-]d [N|W|S|E]` or `[-]d m [N|W|S|E]` or `[-]d m s [N|W|S|E]`

N and E are taken as positive. S and W will negate the value otherwise computed. d m and s are positive, floating point values, taken as degrees, minutes, and seconds, respectively. A leading minus sign will negate the entire value otherwise computed (so, e.g., a '-' and a 'S' will result in a double-negation).

Arguments:

`dmsstring` a string of the format described above

Returns:

a double precision floating point value in radians

Callable From:

- ◇ C++
- ◇ Script

5.106.18 function diguy_format_lat_or_lon

Prototype:

```
void diguy_format_lat_or_lon(char* outbuf,
    double value,
    bool is_lat,
    bool is_decimal,
    bool value_is_degrees);
```

Description:

Returns a string of the form

d [N|W|S|E] or d m s [N|W|S|E]

N and E are used for positive angles. S and W are used for negative angles. In the d m s format, d and m are integers (degrees and minutes) and s (seconds) is a floating point value. In the d format, d (degrees) is a floating point value.

Arguments:

outbuf a buffer into which to copy the resulting string

<code>value</code>	the lat or lon value to be converted into a string
<code>is_lat</code>	boolean indicating latitude (North/South) rather than longitude (East/West)
<code>is_decimal</code>	boolean indicating that decimal degrees are desired, rather than degrees, minutes, and seconds
<code>value_is_degrees</code>	boolean indicating whether the input value is in degrees (as opposed to radians)

Callable From:

- ◇ C++
- ◇ Script

5.106.19 function `diguy_hla_get_ipaddr`

Prototype:

```
int diguy_hla_get_ipaddr(char** ipaddr);
```

Description:

Gets the IP address currently being used by DI-Guy networking. This address is the address to which messages are sent. Normally, it is some kind of broadcast or multicast address.

Arguments:

`ipaddr` IP address, returned; see below

`ipaddr` is a `char**`, into which a `char*` will be placed. This `char*` points to static data owned by DI-Guy. Do not store this pointer, and do not modify what it points to. Copy the strings you need out of the static buffer. (Assignment of a `char*` to a Microsoft CString will work very nicely, for example.)

ipaddr is a string containing 4 integers separated by periods.

Returns:

0 if successful

Callable From:

- ◇ C++
- ◇ Script

5.106.20 function diguy_hla_get_publish_local_chars

Prototype:

```
int diguy_hla_get_publish_local_chars();
```

Description:

Returns whether DI-Guy Networking will publish local characters when online.

C++, Script

Arguments:

none

Returns:

1 if enabled, 0 if not

5.106.21 function diguy_hla_get_display_remote_chars**Prototype:**

```
int diguy_hla_get_display_remote_chars();
```

Description:

Returns whether DI-Guy Networking will display remote characters when online.

C++, Script

Arguments:

none

Returns:

1 if enabled, 0 if not

5.106.22 function diguy_hla_set_exercise**Prototype:**

```
int diguy_hla_set_exercise(int exercise);
```

Description:

Sets the exercise id to be used by DI-Guy Networking.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, Script

Arguments:

`exercise` the exercise id to use

Returns:

0 on success, -1 on failure

5.106.23 function diguy_hla_set_host

Prototype:

```
int diguy_hla_set_host(int host);
```

Description:

Sets the host id to be used by DI-Guy Networking.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, Script

Arguments:

host	the host id to use; the host id identifies the local machine

Returns:

0 on success, -1 on failure

5.106.24 function diguy_hla_set_site

Prototype:

```
int diguy_hla_set_site(int site);
```

Description:

Sets the site id to be used by DI-Guy Networking.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, Script

Arguments:

site	the site id to use; the site id identifies where the local machine is

Returns:

0 on success, -1 on failure

5.106.25 function diguy_hla_set_port

Prototype:

```
int diguy_hla_set_port(int port);
```

Description:

Sets the network port number to be used by DI-Guy Networking.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, Script

Arguments:

<code>port</code>	the port number to use

Returns:

0 on success, -1 on failure

5.106.26 function diguy_hla_set_lat_lon

Prototype:

```
int diguy_hla_set_lat_lon(double lat, double lon);
```

Description:

Sets the latitude and longitude to be used by DI-Guy Networking. The latitude and longitude are taken as the location, on Earth, of the origin of the database.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, Script

Arguments:

<code>lat</code>	the latitude in radians, positive north
<code>lon</code>	the longitude in radians, positive east

Returns:

0 on success, -1 on failure

5.106.27 function diguy_hla_set_ipaddr

Prototype:

```
int diguy_hla_set_ipaddr(const char* ipaddr);
```

Description:

Sets the IP address to be used by DI-Guy Networking. This address is the address to which messages are sent. Normally, it is a broadcast or multicast address.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, *Script*

Arguments:

`ipaddr` the IP address

`ipaddr` is a string containing 4 integers separated by periods, or the null string, which means "let DI-Guy choose the address". An address of "0" or

"0.0.0.0" means broadcast.

Returns:

0 on success, -1 on failure

5.106.28 function diguy_hla_set_display_remote_chars

Prototype:

```
int diguy_hla_set_display_remote_chars(int display);
```

Description:

Sets whether DI-Guy Networking will display remote characters.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, *Script*

Arguments:

<code>display</code>	pass 1 if DI-Guy Networking should display remote characters, 0 if not

Returns:

0 on success, -1 on failure

5.106.29 function diguy_hla_set_publish_local_chars**Prototype:**

```
int diguy_hla_set_publish_local_chars(int publish);
```

Description:

Sets whether DI-Guy Networking will publish local characters to the network.

This function WILL succeed even if DI-Guy is on-line (is participating in an exercise).

C++, Script

Arguments:

<code>publish</code>	pass 1 if DI-Guy Networking should publish local characters to the network, 0 if not

Returns:

0 on success, -1 on failure

5.106.30 function diguy_hla_get_position_offset**Prototype:**

```
int diguy_hla_get_position_offset(float* z, float* yaw, float* roll, float* pitch);
```

Description:

This function gets the position offset which is currently added to each character when it is published, and subtracted from each character when it is reflected. The offset is useful to compensate for inter-application database location and orientation error.

Arguments:

<code>z</code>	z offset in meters
<code>yaw</code>	orientation offset in degrees
<code>roll</code>	orientation offset in degrees
<code>pitch</code>	orientation offset in degrees

Returns:

0 on success, -1 on failure

5.106.31 function diguy_hla_set_position_offset

Prototype:

```
int diguy_hla_set_position_offset(float z, float yaw, float roll, float pitch);
```

Description:

This function sets the position offset which is currently added to each character when it is published, and subtracted from each character when it is reflected. This is useful to compensate for inter-application location error.

THIS FUNCTION DOES NOTHING if called while VR-Link is online.

Arguments:

z	z offset in meters
yaw	orientation offset in degrees
roll	orientation offset in degrees
pitch	orientation offset in degrees

Returns:

0 on success, -1 on failure

5.106.32 function diguy_hla_get_sync_mode_enabled

Prototype:

```
int diguy_hla_get_sync_mode_enabled();
```

Description:

Gets whether sync mode is enabled when online.

C++, Script

Arguments:

none

Returns:

1 if sync mode is enabled when online, 0 if not

5.106.33 function diguy_hla_get_sync_mode_master**Prototype:**

```
int diguy_hla_get_sync_mode_master();
```

Description:

Gets whether this machine will serve as sync master when online and in sync mode.

C++, Script

Arguments:

none

Returns:

1 if this machine will serve as sync master when online and in sync mode, 0 if not

5.106.34 function diguy_hla_get_sync_mode_slave_cam**Prototype:**

```
int diguy_hla_get_sync_mode_slave_cam();
```

Description:

Gets whether this machine's camera will be slaved to a remote master when online and in sync mode and not sync master.

C++, Script

Arguments:

none

Returns:

1 if this machine's camera will be slaved to a remote master when online and in sync mode and not sync master, 0 if not

5.106.35 function diguy_hla_set_use_database_coords**Prototype:**

```
int diguy_hla_set_use_database_coords(int use_database_coords);
```

Description:

Sets whether networking should use database coordinates or not. If database coordinates are used, the database origin is placed at the center of the earth. This makes coordinate conversion to and from UTM unnecessary. Using database coordinates renders ineffective any UTM-related settings which would position the database.

C++, Script

Arguments:

`use_database_coords` 1 if database coordinates should be used, else 0

Returns:

0 if successful, -1 on failure

5.106.36 function diguy_hla_set_use_user_defined_coords**Prototype:**

```
int diguy_hla_set_use_user_defined_coords(int x);
```

Description:

Sets whether networking should use user-defined coordinates or not. If user-defined coordinates are not used, the database origin is placed at the earth location specified in the graphics files for the database.

C++, Script

Arguments:

`x` pass 1 if user-defined coordinates should be used, else 0

Returns:

1 if this machine's camera will be slaved to a remote master when online and in sync mode and not sync master; 0 if not

5.106.37 function diguy_hla_set_sync_mode_enabled**Prototype:**

```
int diguy_hla_set_sync_mode_enabled(int enabled);
```

Description:

Sets whether sync mode is enabled when online.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, Script

Arguments:

`enabled` pass 1 if sync mode should be enabled when online, 0 if not

Returns:

0 on success, -1 on failure

5.106.38 function diguy_hla_set_sync_mode_master**Prototype:**

```
int diguy_hla_set_sync_mode_master(int master);
```

Description:

Sets whether this machine will serve as sync master when online and in sync mode.

C++, Script

Arguments:

`master` pass 1 if this machine will serve as sync master when online and in sync mode, 0 if not

Returns:

0 on success, -1 on failure

5.106.39 function diguy_hla_set_sync_mode_slave_cam**Prototype:**

```
int diguy_hla_set_sync_mode_slave_cam(int sl_cam);
```

Description:

Sets whether this machine's camera will be slaved to a remote master when online and in sync mode and not sync master.

The function will fail if DI-Guy is on-line (is participating in an exercise).

C++, *Script*

Arguments:

sl_cam – pass 1 if this machine's camera will be slaved to a remote master when online and in sync mode and not sync master, 0 if not

Returns:

0 on success, -1 on failure

5.106.40 function diguy_hla_set_sync_mode_camera_offset**Prototype:**

```
int diguy_hla_set_sync_mode_camera_offset(float offset_r, float offset_y, float offs
```

Description:

This function sets the camera orientation offset to be used in sync_mode (when VR-Link is online).

Arguments:

offset_r	orientation offset
offset_y	orientation offset
offset_p	orientation offset

yaw and pitch represent PERCENTAGES of the camera's horizontal and vertical field of view, respectively. The ROLL (which is in degrees) is applied first, then the yaw, then the pitch.

A typical two screen arrangement would use yaws of 50 and -50 for the left screen, and right screen, respectively. For three screens, the yaws would be -100, 0, and 100.

Using percentages this way makes the screens contiguous even if the camera's field of view changes.

THIS FUNCTION DOES NOTHING (and returns -1) if called while VR-Link is online

Returns:

0 on success, -1 on failure (VR-Link is online).

5.106.41 function diguy_hla_get_sync_mode_camera_offset

Prototype:

```
int diguy_hla_get_sync_mode_camera_offset(float* offset_r, float* offset_y, float* offset_p)
```

Description:

This function gets the camera orientation offset to be used in sync_mode (when VR-Link is online).

Arguments:

offset_r	orientation offset
offset_y	orientation offset
offset_p	orientation offset

yaw and pitch represent PERCENTAGES of the camera's horizontal and vertical field of view, respectively. The ROLL (which is in degrees) is applied first, then the yaw, then the pitch.

In the orientation offset the yaw and pitch represent PERCENTAGES of the camera's horizontal and vertical field of view, respectively. The ROLL (which is in degrees) is applied first, then the yaw, then the pitch.

A typical two screen arrangement would use yaws of 50 and -50 for the left screen, and right screen, respectively. For three screens, the yaws would be 100, 0, and -100 going from left to right.

Using percentages this way makes the screens contiguous even if the camera's field of view changes.

Returns:

0 on success, -1 on failure

5.106.42 function diguy_hla_set_sync_mode_camera_slaved

Prototype:

```
int diguy_hla_set_sync_mode_camera_slaved(int slaved);
```

Description:

This function sets whether the camera will be slaved when VR-Link is online, in sync mode, and not sync master.

It can be set even when VR-Link is online.

Arguments:

`slaved` whether the camera will be slaved when VR-Link is
online, in sync mode, and not sync master.

Returns:

0 on success, -1 on failure

5.106.43 function diguy_hla_get_sync_mode_camera_slaved**Prototype:**

```
int diguy_hla_get_sync_mode_camera_slaved();
```

Description:

This function gets whether the camera is currently slaved. It does NOT simply return what was passed to `diguy_hla_set_sync_mode_camera_slaved()`.

Returns:

Whether the camera is currently slaved. This will be 1 only if VR-link is online, is in sync mode, and is not the sync master, and if `sync_mode_camera_slaved` has been set to 1.

5.106.44 function diguy_hla_get_next_participant_name**Prototype:**

```
void* diguy_hla_get_next_participant_name(char* buf, void* handle);
```

Description:

This function allows iteration through the list of names remote VR-Link hosts that DI-Guy knows about.

Arguments:

<code>buf</code>	destination buffer for host name (64 bytes is plenty)
<code>handle</code>	return value from last call to the function; pass NULL to get the first host name; pass previous return value to get the next host name

Returns:

`void*` to be passed as `handle` argument on the next call. NULL when end of list is reached.

5.106.45 function diguy_hla_set_camera_details_out_of_date**Prototype:**

```
int diguy_hla_set_camera_details_out_of_date();
```

Description:

Lets VR-Link know that the user has moved the camera, so that if it is in sync mode, and is the sync master, it can let other hosts know the new camera position so they can move their cameras.

Returns:

0 on success, -1 on failure

5.106.46 function diguy_hla_get_model_from_septet**Prototype:**

```
int diguy_hla_get_model_from_septet(const char** chartype, const char** appearance,
```

Description:

Fetches what character type and what appearance should be used for a given DIS entity type septet.

Arguments:

<code>chartype</code>	address of char*, for returning a string which names the DI-Guy character type
<code>appearance</code>	address of char*, for returning a string which names the DI-Guy character appearance
<code>septet</code>	array of seven integers specifying the kind, domain, country, category, subCategory, specific, and extra of the entity in question

Returns:

0 on success, -1 on failure

5.106.47 function diguy_hla_get_septet_from_model

Prototype:

```
int diguy_hla_get_septet_from_model(int septet[7], const char* chartype, const char*
```

Description:

Fetches what DIS entity type septet should be used for a given character type and appearance.

Arguments:

<code>chartype</code>	a string which names the DI-Guy character type
<code>appearance</code>	a string which names the DI-Guy character appearance
<code>septet</code>	array of seven integers, returned, indicating the kind, domain, country, category, subCategory, specific, and extra of the entity in question

Returns:

0 on success, -1 on failure

5.106.48 function diguy_hla_get_earth_coord

Prototype:

```
int diguy_hla_get_earth_coord(char* outbuf, const float x, const float y, const float z)
```

Description:

Converts a 3d database-origin-relative location into a string describing it in earth coordinates. This string could be in a number of different formats including decimal lat/lon, dms lat/lon, and UTM.

Arguments:

outbuf	buffer into which to put the output string
x, y, z	3 floats specifying location

Returns:

0 on success, -1 on failure. Fails if DI-Guy Scenario is not currently on the network, meaning that a database-to-earth coordinate conversion cannot be performed.

5.106.49 function diguy_hla_toggle_earth_coord_format**Prototype:**

```
void diguy_hla_toggle_earth_coord_format();
```

Description:

Changes the format of the output of diguy_hla_get_earth_coord. Toggles between UTM, dms lat/lon, decimal lat/lon, and "punting", meaning that the caller of diguy_hla_get_earth_coord should fend for itself. This generally means that the caller will use database-origin-relative coordinates.

Arguments:

none

Returns:

void

5.106.50 function diguy_hla_get_character_from_id**Prototype:**

```
diguyCharacter* diguy_hla_get_character_from_id(char const* id);
```

Description:

returns the [diguyCharacter](#) associated with the entity id specified.

Arguments:

<code>id</code>	entity id

Under DIS, two formats are permitted: the HLA object name, or the DIS entity identifier ("site:application:entity", where each is an integer).

Under DIS, only the DIS Entity Identifier is permitted.

5.106.51 function diguy_hla_enter_vpnet_mode**Prototype:**

```
void diguy_hla_enter_vpnet_mode();
```

Description:

Turns the 'never create' character bit on.

5.106.52 function diguy_hla_get_name_from_entity**Prototype:**

```
char* diguy_hla_get_name_from_entity(void* reflected_entity);
```

Description:

This function returns the name corresponding to the passed entity.

The user must not delete the returned string.

Arguments:

`reflected_entity` entity to be named

Returns:

entity name

5.106.53 function diguy_hla_establish_new_offsets

Prototype:

```
void diguy_hla_establish_new_offsets(int deprecated = 0);
```

5.106.54 function diguy_hla_set_send_custom_pdus

Prototype:

```
void diguy_hla_set_send_custom_pdus(int val);
```

5.106.55 function diguy_hla_get_send_custom_pdus

Prototype:

```
int diguy_hla_get_send_custom_pdus();
```

5.107 DI-Guy API, VR-Link

diguy_module_net.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

- ◆ [Coordinate System Functions](#)
- ◆ [Ground Clamp Functions](#)
- ◆ [Network Time Functions](#)
- ◆ [Sync Mode Functions](#)
- ◆ [Experimental Functions](#)

Link against: libdiguy_module_dis or libdiguy_module_hla

5.107.1 enumeration diguyCoordinateSystemType

```
typedef enum
{
    DIGUY_COORDINATE_SYSTEM_NONE          = 0,
    DIGUY_COORDINATE_SYSTEM_UTM           = 1,
    DIGUY_COORDINATE_SYSTEM_FLAT_EARTH    = 2,
    DIGUY_COORDINATE_SYSTEM_GEOCENTRIC    = 3
} diguyCoordinateSystemType;
```

5.107.2 class diguyNetInterface

Description:

This class provides accessors to the interface parameters, as well as functions for going online and offline.

```
class BDI_DECLSPEC diguy diguyNetInterface
{
public:
    virtual int go_online(diguyScenario* scenario, void* exconn = NULL) = 0;
    virtual int go_offline() = 0;
    virtual int get_online() = 0;
    virtual int set_cfg_file(const char* cfg_filename) = 0;
    virtual int update() = 0;
    virtual int set_exercise(int exercise) = 0;
    virtual int get_exercise() = 0;
    virtual int set_host(int host) = 0;
    virtual int get_host() = 0;
    virtual int set_site(int site) = 0;
    virtual int get_site() = 0;
    virtual int set_port(int port) = 0;
    virtual int get_port() = 0;
    virtual int set_ipaddr(const char* ipaddr) = 0;
    virtual int get_ipaddr(char** ipaddr) = 0;
    virtual int set_display_remote_chars(int display) = 0;
    virtual int get_display_remote_chars() = 0;
    virtual void set_send_custom_pdus(int) = 0;
    virtual int get_send_custom_pdus() = 0;
    virtual void enter_vpnet_mode() = 0;
    virtual diguyCharacter* get_character_from_id(char const* id) = 0;
    virtual char* get_name_from_entity(void* reflected_entity) = 0;
    virtual int set_publish_local_chars(int publish) = 0;
    virtual int get_publish_local_chars() = 0;
    virtual int set_position_offset(float z, float yaw, float roll, float pitch) = 0;
    virtual int get_position_offset(float* z,
        float* yaw,
        float* roll,
        float* pitch) = 0;
    virtual void* get_next_participant_name(char* buf, void* handle) = 0;
    virtual int get_model_from_septet(const char** chartype,
        const char** appearance,
        const int septet[7]) = 0;
    virtual int get_septet_from_model(int septet[7],
        const char* chartype,
        const char* appearance) = 0;
    virtual void set_timeout_interval(float interval) = 0;
    virtual float get_timeout_interval() = 0;
    virtual void set_time_to_live(int time_to_live) = 0;
    virtual int get_time_to_live() = 0;
    virtual int set_lat_lon(double lat, double lon) = 0;
    virtual int get_lat_lon(double* lat, double* lon) = 0;
    virtual void format_lat_or_lon(char* outbuf,
        double value,
        bool is_lat,
        bool is_decimal,
        bool value_is_degrees) = 0;
    virtual int set_utm_zone(int zone, int southzone) = 0;
    virtual int set_utm(int zone,
```



```

        int southzone,
        double easting,
        double northing) = 0;
virtual int get_utm(int* zone,
        int* southzone,
        double* easting,
        double* northing) = 0;
virtual int set_use_user_defined_coords(int use_user_defined) = 0;
virtual int get_use_user_defined_coords() = 0;
virtual int get_earth_coord(char* outbuf,
        const float x,
        const float y,
        const float z) = 0;
virtual void toggle_earth_coord_format() = 0;
virtual int set_use_database_coords(int use_dbc) = 0;
virtual int get_num_coordinate_systems();
virtual const char* get_coordinate_system_name(int index);
virtual int set_coordinate_system_type(const char* name);
virtual const char* get_coordinate_system_name();
virtual int set_coordinate_system_type(diguyCoordinateSystemType system_type);
virtual diguyCoordinateSystemType get_coordinate_system_type();
virtual void set_use_UTM_standard_offsets(int standard_offsets);
virtual int get_use_UTM_standard_offsets();
virtual double parse_dms_to_radians(const char* dms_string) = 0;
virtual int convert_mgrs_to_local(const char* MGRS_str,
        float* x,
        float* y,
        float* z) = 0;
virtual const char* convert_local_to_mgrs(float x, float y, float z) = 0;
virtual float convert_local_to_lat(float x, float y, float z) = 0;
virtual float convert_local_to_long(float x, float y, float z) = 0;
virtual const char* convert_local_to_lat_long_string(float x, float y, float z) = 0;
virtual int convert_lat_long_to_local(float lat, float lon, float* x, float* y, float* z) = 0;
virtual int set_ground_clamp_reflected_land_entities_enabled(int enabled);
virtual int get_ground_clamp_reflected_land_entities_enabled();
virtual int set_ground_clamp_reflected_land_entities_from_host(int from_host);
virtual int get_ground_clamp_reflected_land_entities_from_host();
virtual int set_ground_clamp_max_climb(float max_climb);
virtual float get_ground_clamp_max_climb();
virtual int set_ground_clamp_max_drop(float max_drop);
virtual float get_ground_clamp_max_drop();
virtual void set_network_datetime_mode(diguyDatetimeNetworkMode network_datetime_mode);
virtual diguyDatetimeNetworkMode get_network_datetime_mode();
virtual void set_network_datetime_broadcast_interval(float broadcast_interval);
virtual float get_network_datetime_broadcast_interval();
virtual int set_sync_mode_enabled(int enabled) = 0;
virtual int get_sync_mode_enabled() = 0;
virtual int set_sync_mode_master(int master) = 0;
virtual int get_sync_mode_master() = 0;
virtual int set_sync_mode_slave_cam(int sl_cam) = 0;
virtual int get_sync_mode_slave_cam() = 0;
virtual int set_sync_mode_camera_offset(float offset_r, float offset_y, float offset_p);
virtual int get_sync_mode_camera_offset(float* offset_r,
        float* offset_y,
        float* offset_p) = 0;
virtual int set_sync_mode_camera_slaved(int slaved) = 0;
virtual int get_sync_mode_camera_slaved() = 0;
virtual int set_camera_details_out_of_date() = 0;
virtual int set_use_adaptive_guide(int use_adaptive_guide) = 0;

virtual int get_use_adaptive_guide() = 0;

```

```

virtual int set_federation_name(const char* federation_name);
virtual const char* get_federation_name();
virtual int set_culling_enabled(int culling);
virtual int get_culling_enabled();
virtual void set_network_datetime_enable_diguy_custom_pdus(int enable_diguy_custom_pdus);
virtual int get_network_datetime_enable_diguy_custom_pdus();
virtual void set_network_datetime_enable_dis_start_resume_pdus(int enable_dis_start_resume_pdus);
virtual int get_network_datetime_enable_dis_start_resume_pdus();

```

5.107.3 function diguyNetInterface::go_online

Prototype:

```
virtual int go_online(diguyScenario* scenario, void* exconn = NULL) = 0;
```

Description:

This function tells DI-Guy that it should start talking over VR-Link.

It should be called when DI-Guy should join an exercise.

Arguments:

<code>scenario</code>	pointer to the <code>diguyScenario</code> which will be connected to the network
<code>exconn</code>	optional pointer to a VR-Link exercise connection to use

Returns:

0 on success, -1 on failure

5.107.4 function diguyNetInterface::go_offline

Prototype:

```
virtual int go_offline() = 0;
```

Description:

This function tells DI-Guy that it should stop talking over VR-Link.

Returns:

0 on success, -1 on failure

5.107.5 function diguyNetInterface::get_online

Prototype:

```
virtual int get_online() = 0;
```

Description:

This function asks if DI-Guy is connected to the network.

Returns:

1 if yes, 0 if no

5.107.6 function diguyNetInterface::set_cfg_file

Prototype:

```
virtual int set_cfg_file(const char* cfg_filename) = 0;
```

Description:

This function sets the filename of the VR-Link configuration file that will be read by the DI-Guy software. The default is "network.cfg". This function must be called before the network module is initialized.

C++ only

Arguments:

cfg_file filename of desired configuration file

Returns:

0 on success, -1 on failure

5.107.7 function diguyNetInterface::update

Prototype:

```
virtual int update() = 0;
```

Description:

This function tells DI-Guy that it should read PDUs from the network and process them. Call this function periodically.

Returns:

0 on success, -1 on failure

5.107.8 function diguyNetInterface::set_exercise**Prototype:**

```
virtual int set_exercise(int exercise) = 0;
```

Description:

Sets the exercise id to be used by DI-Guy networking.

The function will fail if DI-Guy is on-line (is participating in an exercise).

Arguments:

`exercise` the exercise id to use

Returns:

0 on success, -1 on failure

5.107.9 function diguyNetInterface::get_exercise**Prototype:**

```
virtual int get_exercise() = 0;
```

Description:

Gets the exercise id being used by DI-Guy networking.

Returns:

the exercise id

5.107.10 function diguyNetInterface::get_host**Prototype:**

```
virtual int get_host() = 0;
```

Description:

Gets the host id being used by DI-Guy networking.

Returns:

the host id

5.107.11 function diguyNetInterface::set_site**Prototype:**

```
virtual int set_site(int site) = 0;
```

Description:

Sets the site id to be used by DI-Guy networking.

The function will fail if DI-Guy is on-line (is participating in an exercise).

Arguments:

site	the site id to use; the site id identifies where the local machine is located

Returns:

0 on success, -1 on failure

5.107.12 function diguyNetInterface::get_site**Prototype:**

```
virtual int get_site() = 0;
```

Description:

Gets the site id being used by DI-Guy networking.

Returns:

the site id

5.107.13 function diguyNetInterface::set_port**Prototype:**

```
virtual int set_port(int port) = 0;
```

Description:

Sets the network port number to be used by DI-Guy networking.

The function will fail if DI-Guy is on-line (is participating in an exercise).

Arguments:

port the port number to use

Returns:

0 on success, -1 on failure

5.107.14 function diguyNetInterface::get_port**Prototype:**

```
virtual int get_port() = 0;
```

Description:

Gets the port number of the port being used for DI-Guy networking.

Returns:

the port address

5.107.15 function diguyNetInterface::set_ipaddr**Prototype:**

```
virtual int set_ipaddr(const char* ipaddr) = 0;
```

Description:

Sets the IP address to be used by DI-Guy networking. This address is the address to which messages are sent. Normally, it is a broadcast or multicast address.

The function will fail if DI-Guy is on-line (is participating in an exercise).

Arguments:

`ipaddr` the IP address

`ipaddr` is a string containing 4 integers separated by periods, or the null string, which means "let DI-Guy choose the address". An address of "0" or "0.0.0.0" means broadcast.

Returns:

0 on success, -1 on failure

5.107.16 function diguyNetInterface::get_ipaddr

Prototype:

```
virtual int get_ipaddr(char** ipaddr) = 0;
```

Description:

Gets the IP address currently being used by DI-Guy networking. This address is the address to which messages are sent. Normally this is a broadcast or multicast address.

Arguments:

ipaddr IP address, returned; see below

Returns:

0 on success, -1 on failure

ipaddr is a char**, into which a char* will be placed. This char* points to static data owned by DI-Guy. Do not store this pointer, and do not modify

what it points to. Copy the strings you need out of the static buffer.
(Assignment of a char* to a Microsoft CString will work very nicely, for example.)

ipaddr is a string containing 4 integers separated by periods.

5.107.17 function diguyNetInterface::set_display_remote_chars

Prototype:

```
virtual int set_display_remote_chars(int display) = 0;
```

Description:

Sets whether DI-Guy networking will display remote characters.

The function will fail if DI-Guy is on-line (is participating in an exercise).

Arguments:

display	pass 1 if DI-Guy networking should display remote characters, 0 if not

Returns:

0 on success, -1 on failure

5.107.18 function diguyNetInterface::get_display_remote_chars**Prototype:**

```
virtual int get_display_remote_chars() = 0;
```

Description:

Returns whether DI-Guy networking will display remote characters when online.

Returns:

1 if enabled, 0 if not

5.107.19 function diguyNetInterface::set_send_custom_pdus**Prototype:**

```
virtual void set_send_custom_pdus(int) = 0;
```

Description:

This function sets whether DI-Guy is sending custom pdus.

Arguments:

0 if custom pdus should not be sent, 1 if custom pdus should be sent

5.107.20 function diguyNetInterface::get_send_custom_pdus**Prototype:**

```
virtual int get_send_custom_pdus() = 0;
```

Description:

This function returns whether DI-Guy is sending custom pdus.

Returns:

0 if custom pdus are not being sent, 1 if custom pdus are being sent

5.107.21 function diguyNetInterface::enter_vpnet_mode**Prototype:**

```
virtual void enter_vpnet_mode() = 0;
```

Description:

Turns the 'never create character' bit on.

5.107.22 function diguyNetInterface::get_character_from_id**Prototype:**

```
virtual diguyCharacter* get_character_from_id(char const* id) = 0;
```

Description:

returns the [diguyCharacter](#) associated with the entity id specified.

Arguments:

id	Under HLA, two formats are permitted: the H L A object name, or the DIS entity identifier ("site:application:entity", where each is an integer)

Under DIS, only the DIS Entity Identifier is permitted.

5.107.23 function diguyNetInterface::get_name_from_entity**Prototype:**

```
virtual char* get_name_from_entity(void* reflected_entity) = 0;
```

Description:

Returns the name corresponding to the entity. The caller must delete the returned string.

Arguments:

reflected_entity	entity to be named

Returns:

name of reflected entity

5.107.24 function diguyNetInterface::set_publish_local_chars**Prototype:**

```
virtual int set_publish_local_chars(int publish) = 0;
```

Description:

Sets whether DI-Guy networking will publish local characters to the network.

This function WILL succeed even if DI-Guy is on-line (is participating in an exercise).

Arguments:

<code>publish</code>	pass 1 if DI-Guy networking should publish local characters to the network, 0 if not

Returns:

0 on success, -1 on failure

5.107.25 function diguyNetInterface::set_position_offset**Prototype:**

```
virtual int set_position_offset(float z, float yaw, float roll, float pitch) = 0;
```

Description:

This function sets the position offset which is currently added to each character when it is published, and subtracted from each character when it is reflected. This is useful to compensate for inter-application location error.

Arguments:

<code>z</code>	z offset in meters
<code>yaw</code>	orientation offset in degrees
<code>roll</code>	orientation offset in degrees
<code>pitch</code>	orientation offset in degrees

Returns:

0 on success, -1 on failure

5.107.26 function diguyNetInterface::get_position_offset

Prototype:

```
virtual int get_position_offset(float* z,  
    float* yaw,  
    float* roll,  
    float* pitch) = 0;
```

Description:

This function gets the position offset which is currently added to each character when it is published, and subtracted from each character when it is reflected. The offset is useful to compensate for inter-application database location and orientation error.

Arguments:

z	z offset in meters
yaw	orientation offset in degrees
roll	orientation offset in degrees
pitch	orientation offset in degrees

Returns:

0 on success, -1 on failure

5.107.27 function diguyNetInterface::get_next_participant_name

Prototype:

```
virtual void* get_next_participant_name(char* buf, void* handle) = 0;
```

Description:

This function allows iteration through the list of names remote VR-Link hosts that DI-Guy knows about.

Arguments:

<code>buf</code>	destination buffer for host name (64 bytes is plenty)
<code>handle</code>	return value from last call to the function; pass NULL to get the first host name; pass previous return value to get the next host name

Returns:

void* to be passed as handle argument on the next call; NULL when end of list is reached

5.107.28 function diguyNetInterface::get_septet_from_model**Prototype:**

```
virtual int get_septet_from_model(int septet[7],
    const char* chartype,
    const char* appearance) = 0;
```

Description:

Fetches what DIS entity type septet should be used for a given character type and appearance.

Arguments:

<code>chartype</code>	a string which names the DI-Guy character type
<code>appearance</code>	a string which names the DI-Guy character appearance
<code>septet</code>	array of seven integers, returned, indicating the kind, domain, country, category, subCategory, specific, and extra of the entity in question

Returns:

0 on success, -1 on failure

5.107.29 function diguyNetInterface::set_timeout_interval**Prototype:**

```
virtual void set_timeout_interval(float interval) = 0;
```

Description:

This functions sets the amount of time that can elapse since the last update before an entity is timed out. The default is 12 seconds.

Arguments:

<code>interval</code>	number of seconds until timeout

5.107.30 function diguyNetInterface::get_timeout_interval**Prototype:**

```
virtual float get_timeout_interval() = 0;
```

Returns:

timeout value in seconds; see [set_timeout_interval\(\)](#)

5.107.31 function diguyNetInterface::set_time_to_live**Prototype:**

```
virtual void set_time_to_live(int time_to_live) = 0;
```

Description:

This function sets the number of hops a packet can take before being killed. Setting this value to -1 means that the system default will be used.

This value must be set before the connection goes online.

Arguments:

`time_to_live` number of hops until packet is killed

5.107.32 function diguyNetInterface::get_time_to_live**Prototype:**

```
virtual int get_time_to_live() = 0;
```

Returns:

number of hops until packed is killed; see [set_time_to_live\(\)](#)

Coordinate System Functions

5.107.33 function diguyNetInterface::set_lat_lon**Prototype:**

```
virtual int set_lat_lon(double lat, double lon) = 0;
```

Description:

Sets the latitude and longitude to be used by DI-Guy networking. The latitude and longitude are taken as the location, on Earth, of the origin of the database.

The function will fail if DI-Guy is on-line (is participating in an exercise).

Arguments:

lat	the latitude in radians, positive north
lon	the longitude in radians, positive east

Returns:

0 on success, -1 on failure

5.107.34 function diguyNetInterface::get_lat_lon**Prototype:**

```
virtual int get_lat_lon(double* lat, double* lon) = 0;
```

Description:

Gets the latitude and longitude currently being used by DI-Guy networking for the placement of the database origin.

Arguments:

lat	latitude of origin, returned; see below
lon	longitude of origin, returned; see below

Returns:

0 on success, -1 on failure

lat and lon are char**'s, into which char*'s will be placed. These char*'s point to static data owned by DI-Guy. Do not store these pointers, and do not

modify what they point to. Copy the strings needed out of the static buffers. (Assignment of a char* to a Microsoft CString will work very nicely, for example.)

lat and lon are expressed as strings of the form:

◇ [-]d [N|W|S|E] or
 ◇ [-]d m [N|W|S|E] or
 ◇ [-]d m s [N|W|S|E]

N and E are taken as positive. S and W will negate the value otherwise computed. d, m, and s are positive, floating point values, taken as degrees, minutes, and seconds, respectively. A leading minus sign will negate the entire value otherwise computed (so, e.g., a '-' and a 'S' will result in a double- negation).

5.107.35 function diguyNetInterface::format_lat_or_lon

Prototype:

```
virtual void format_lat_or_lon(char* outbuf,
    double value,
    bool is_lat,
    bool is_decimal,
    bool value_is_degrees) = 0;
```

Description:

returns a string of the form

◇ d [N|W|S|E] or
 ◇ d m s [N|W|S|E]

N and E are used for positive angles. S and W are used for negative angles. In the d m s format, d and m are integers (degrees and minutes) and s (seconds) is a floating point value. In the d format, d (degrees) is a floating point value.

Arguments:

<code>outbuf</code>	a buffer into which to copy the resulting string
<code>value</code>	the lat or lon value to be converted into a string
<code>is_lat</code>	boolean indicating latitude (North/South) rather than longitude(East/West)
<code>is_decimal</code>	boolean indicating that decimal degrees are desired, rather than degrees, minutes, and seconds
<code>value_is_degrees</code>	boolean indicating whether the input value is in degrees (as opposed to radians)

5.107.36 function `diguyNetInterface::set_utm_zone`

Prototype:

```
virtual int set_utm_zone(int zone, int southzone) = 0;
```

Description:

Sets the UTM zone of the location currently being used by DI-Guy networking for the placement of the database origin. The easting and northing are assumed not to change.

Arguments:

<code>zone</code>	the utm zone
<code>southzone</code>	pass 1 if the origin is in the southern hemisphere

Returns:

0 on success, -1 on failure

5.107.37 function diguyNetInterface::set_utm**Prototype:**

```
virtual int set_utm(int zone,  
    int southzone,  
    double easting,  
    double northing) = 0;
```

Description:

Sets, in UTM coordinates, the location currently being used by DI-Guy networking for the placement of the database origin.

Arguments:

zone	the utm zone
southzone	pass 1 if the origin is in the southern hemisphere
easting	the normalized (positive only) easting of the database origin within the zone
northing	the normalized (positive only) northing of the database origin within the zone

Returns:

0 on success, -1 on failure

5.107.38 function diguyNetInterface::get_utm**Prototype:**

```
virtual int get_utm(int* zone,  
    int* southzone,  
    double* easting,  
    double* northing) = 0;
```

Description:

Gets, in UTM coordinates, the location currently being used by DI-Guy networking for the placement of the database origin.

Arguments:

zone	the utm zone, returned
southzone	1 if the origin is in the southern hemisphere, returned
easting	the normalized (positive only) easting of the database origin within the zone, returned
northing	the normalized (positive only) northing of the database origin within the zone, returned

Returns:

0 on success, -1 on failure

5.107.39 function `diguyNetInterface::set_use_user_defined_coords`

Prototype:

```
virtual int set_use_user_defined_coords(int use_user_defined) = 0;
```

Description:

This function sets whether user-defined latitude/longitude coordinates should be used for the origin of the database, or the coordinates that come from graphics files for scenario scene objects.

Arguments:

`use_user_defined` pass 1 if user-defined coordinates should be used, 0 if coordinates from graphics file should be used

Returns:

0 on success, -1 on failure

5.107.40 function `diguyNetInterface::get_use_user_defined_coords`

Prototype:

```
virtual int get_use_user_defined_coords() = 0;
```

Returns:

0 if graphics file coordinates are used, 1 if user-defined coordinates are used

5.107.41 function diguyNetInterface::get_earth_coord**Prototype:**

```
virtual int get_earth_coord(char* outbuf,  
    const float x,  
    const float y,  
    const float z) = 0;
```

Description:

Converts a 3d database-origin-relative location into a string describing it in earth coordinates. This string could be in a number of different formats including decimal lat/lon, dms lat/lon, and UTM.

Arguments:

outbuf	buffer into which to put the output string
x, y, z	3 floats specifying location

Returns:

0 on success, -1 on failure. Fails if DI-Guy Scenario is not currently on the network, meaning that a database-to-earth coordinate conversion cannot be performed.

5.107.42 function diguyNetInterface::toggle_earth_coord_format**Prototype:**

```
virtual void toggle_earth_coord_format() = 0;
```

Description:

Changes the format of the output of `get_earth_coord`. Toggles between UTM, dms lat/lon, decimal lat/lon, and "punting", meaning that the caller of `get_earth_coord( )` should fend for itself. This generally means that the caller will use database-origin-relative coordinates.

5.107.43 function diguyNetInterface::set_use_database_coords**Prototype:**

```
virtual int set_use_database_coords(int use_dbc) = 0;
```

Description:

Sets whether networking should use database coordinates or not. If database coordinates are used, the database origin is placed at the center of the earth. This makes coordinate conversion to and from UTM unnecessary. Using database coordinates renders ineffective any UTM-related settings which would position the database.

Arguments:

`use_database_coords` 1 if database coordinates should be used, 0 otherwise

Returns:

0 if successful, -1 on failure.

5.107.44 function `diguyNetInterface::get_num_coordinate_systems`

Prototype:

```
virtual int get_num_coordinate_systems();
```

Description:

This function returns the number of potential coordinate converters that the networking module supports.

Returns:

number of converters

5.107.45 function `diguyNetInterface::set_coordinate_system_type`

Prototype:

```
virtual int set_coordinate_system_type(const char* name);
```

Description:

Sets the coordinate system based on the name given. Currently supports: "No Conversion", "UTM", "Flat Earth", "Geocentric".

Returns:

0 on success, -1 on failure.

5.107.46 function `diguyNetInterface::set_coordinate_system_type`

Prototype:

```
virtual int set_coordinate_system_type(diguyCoordinateSystemType system_type);
```

Description:

Sets the coordinate system based on the system type.

Returns:

0 on success, -1 on failure.

5.107.47 function `diguyNetInterface::get_coordinate_system_type`

Prototype:

```
virtual diguyCoordinateSystemType get_coordinate_system_type();
```

Description:

Gets the current coordinate system in use.

Returns:

`diguyCoordinateSystemType` enumeration

5.107.48 function `diguyNetInterface::set_use_UTM_standard_offsets`

Prototype:

```
virtual void set_use_UTM_standard_offsets(int standard_offsets);
```

Description:

Documentation Pending

5.107.49 function `diguyNetInterface::get_use_UTM_standard_offsets`

Prototype:

```
virtual int get_use_UTM_standard_offsets();
```

Description:

Documentation Pending

5.107.50 function diguyNetInterface::parse_dms_to_radians**Prototype:**

```
virtual double parse_dms_to_radians(const char* dms_string) = 0;
```

Description:

Parses a string of the form

◇ [-]d [N|W|S|E] or
 ◇ [-]d m [N|W|S|E] or
 ◇ [-]d m s [N|W|S|E]

N and E are taken as positive. S and W will negate the value otherwise computed. d m and s are positive, floating point values, taken as degrees, minutes, and seconds, respectively. A leading minus sign will negate the entire value otherwise computed (so, e.g., a '-' and a 'S' will result in a double-negation).

Arguments:

`dms_string` a string of the format described above

Returns:

a double precision floating point value in radians.

5.107.51 function diguyNetInterface::convert_mgrs_to_local**Prototype:**

```
virtual int convert_mgrs_to_local(const char* MGRS_str,  
    float* x,  
    float* y,  
    float* z) = 0;
```

Description:

Documentation Pending

5.107.52 function diguyNetInterface::convert_local_to_lat**Prototype:**

```
virtual float convert_local_to_lat(float x, float y, float z) = 0;
```

Description:

Documentation Pending

5.107.53 function diguyNetInterface::convert_local_to_long**Prototype:**

```
virtual float convert_local_to_long(float x, float y, float z) = 0;
```

Description:

Documentation Pending

5.107.54 function diguyNetInterface::convert_lat_long_to_local**Prototype:**

```
virtual int convert_lat_long_to_local(float lat, float lon, float* x, float* y, float* z) = 0;
```

Description:

Documentation Pending

Ground Clamp Functions

5.107.55 function

diguyNetInterface::set_ground_clamp_reflected_land_entities_en

Prototype:

```
virtual int set_ground_clamp_reflected_land_entities_enabled(int enabled);
```

Description:

Sets whether land-based network entities will be ground clamped, changing their altitude to better correlate with the local terrain.

Arguments:

`enabled` pass 1 to enable automatic ground clamping; 0 to disable

Returns:

0 on success, -1 on failure

5.107.56 function**diguyNetInterface::get_ground_clamp_reflected_land_entities_en****Prototype:**

```
virtual int get_ground_clamp_reflected_land_entities_enabled();
```

Returns:

1 if land-based network entities will be ground clamped, 0 if not

5.107.57 function**diguyNetInterface::set_ground_clamp_reflected_land_entities_fro****Prototype:**

```
virtual int set_ground_clamp_reflected_land_entities_from_host(int from_host);
```

Description:

Similar to [set_ground_clamp_reflected_land_entities_enabled\(\)](#), but will clamp only entities from the specified host. If there is one host on the network that has particular trouble sending correlated altitudes this function can be used to clamp its entities without clamping all entities from all hosts.

This has been useful in some simulation exercises where participants from multiple vendors are in use. Entities from JCATS in particular have required this.

Arguments:

`from_host` host whose land entities will be clamped

Returns:

0 on success, -1 on failure

5.107.58 function

diguyNetInterface::get_ground_clamp_reflected_land_entities_from_host()

Prototype:

```
virtual int get_ground_clamp_reflected_land_entities_from_host();
```

Returns:

host whose land entities will be clamped; see
[set_ground_clamp_reflected_land_entities_from_host\(\)](#)

5.107.59 function

diguyNetInterface::set_ground_clamp_max_climb

Prototype:

```
virtual int set_ground_clamp_max_climb(float max_climb);
```

Description:

Sets the maximum upward change in altitude that ground clamping will have.

Arguments:

`max_climb` maximum upward change in altitude, in meters

Returns:

0 on success, -1 on failure

5.107.60 function

diguyNetInterface::get_ground_clamp_max_climb

Prototype:

```
virtual float get_ground_clamp_max_climb();
```

Returns:

maximum upward change in altitude from ground clamping, in meters

5.107.61 function

`diguyNetInterface::set_ground_clamp_max_drop`

Prototype:

```
virtual int set_ground_clamp_max_drop(float max_drop);
```

Description:

Similar to `set_ground_clamp_max_climb()`, but for downward change.

5.107.62 function

`diguyNetInterface::get_ground_clamp_max_drop`

Prototype:

```
virtual float get_ground_clamp_max_drop();
```

Returns:

maximum downward change in altitude from ground clamping, in meters

Network Time Functions

5.107.63 function `diguyNetInterface::set_network_datetime_mode`

Prototype:

```
virtual void set_network_datetime_mode(diguyDatetimeNetworkMode network_datetime_mode)
```

Description:

Sets whether network datetime data will be broadcast, received, or ignored. See documentation for [diguyDatetimeNetworkMode](#) for more information.

Arguments:

`network_datetime_mode` how to handle network datetime data

5.107.64 function diguyNetInterface::get_network_datetime_mode

Prototype:

```
virtual diguyDatetimeNetworkMode get_network_datetime_mode();
```

Returns:

how handle network datetime data will be handled; see
[set_network_datetime_mode\(\)](#).

5.107.65 function diguyNetInterface::set_network_datetime_broadcast_interval

Prototype:

```
virtual void set_network_datetime_broadcast_interval(float broadcast_interval);
```

Description:

Sets how often datetime data will be broadcast if this host is in mode
[DIGUY_DATETIME_NETWORK_MODE_BROADCAST](#).

Default is broadcast every 5 seconds.

Arguments:

`broadcast_interval` how often to broadcast datetime data, in seconds

5.107.66 function

`diguyNetInterface::get_network_datetime_broadcast_interval`

Prototype:

```
virtual float get_network_datetime_broadcast_interval();
```

Returns:

how often datetime data will broadcast, in seconds

Sync Mode Functions

These functions enable DI-Guy to do pseudo frame syncing across multiple machines.

5.107.67 function diguyNetInterface::set_sync_mode_enabled**Prototype:**

```
virtual int set_sync_mode_enabled(int enabled) = 0;
```

Description:

Sets whether sync mode will be enabled when online.

The function will fail if DI-Guy is already online, participating in an exercise.

Arguments:

enabled pass 1 if sync mode should be enabled when online, 0 if not

Returns:

0 on success, -1 on failure

5.107.68 function diguyNetInterface::get_sync_mode_enabled**Prototype:**

```
virtual int get_sync_mode_enabled() = 0;
```

Description:

Gets whether sync mode is enabled when online.

Returns:

1 if sync mode is enabled when online, 0 if not

5.107.69 function diguyNetInterface::set_sync_mode_master**Prototype:**

```
virtual int set_sync_mode_master(int master) = 0;
```

Description:

Sets whether this machine will serve as sync master when online and in sync mode.

Arguments:

master	pass 1 if this machine will serve as sync master when online and in sync mode, 0 if not

Returns:

0 on success, -1 on failure

5.107.70 function diguyNetInterface::get_sync_mode_master**Prototype:**

```
virtual int get_sync_mode_master() = 0;
```

Description:

Gets whether this machine will serve as sync master when online and in sync mode.

Returns:

1 if this machine will serve as sync master when online and in sync mode, 0 if not

5.107.71 function diguyNetInterface::set_sync_mode_slave_cam**Prototype:**

```
virtual int set_sync_mode_slave_cam(int sl_cam) = 0;
```

Description:

Sets whether this machine's camera will be slaved to a remote master when online and in sync mode and not sync master.

The function will fail if DI-Guy is already online, participating in an exercise.

Arguments:

sl_cam – pass 1 if this machine's camera will be slaved to a remote master when online and in sync mode and not sync master, 0 if not

Returns:

0 on success, -1 on failure

5.107.72 function diguyNetInterface::get_sync_mode_slave_cam**Prototype:**

```
virtual int get_sync_mode_slave_cam() = 0;
```

Description:

Gets whether this machine's camera will be slaved to a remote master when online, in sync mode, and not sync master.

Returns:

1 if this machine's camera will be slaved, 0 if not

5.107.73 function

diguyNetInterface::set_sync_mode_camera_offset

Prototype:

```
virtual int set_sync_mode_camera_offset(float offset_r, float offset_y, float offset_p)
```

Description:

This function sets the camera orientation offset to be used in sync_mode (when VR-Link is online).

Arguments:

offset_r	orientation offset
offset_y	orientation offset
offset_p	orientation offset

yaw and pitch represent PERCENTAGES of the camera's horizontal and vertical field of view, respectively. The ROLL (which is in degrees) is applied first, then the yaw, then the pitch.

A typical two screen arrangement would use yaws of 50 and -50 for the left screen, and right screen, respectively. For three screens, the yaws would be -100, 0, and 100.

Using percentages this way makes the screens contiguous even if the camera's field of view changes.

THIS FUNCTION DOES NOTHING (and returns -1) if called while VR-Link is online

Returns:

0 on success, -1 on failure (VR-Link is online).

5.107.74 function

diguyNetInterface::get_sync_mode_camera_offset

Prototype:

```
virtual int get_sync_mode_camera_offset(float* offset_r,
    float* offset_y,
    float* offset_p) = 0;
```

Description:

This function gets the camera orientation offset to be used in sync_mode (when VR-Link is online).

Arguments:

offset_r	orientation offset
offset_y	orientation offset
offset_p	orientation offset

yaw and pitch represent PERCENTAGES of the camera's horizontal and vertical field of view, respectively. The ROLL (which is in degrees) is applied first, then the yaw, then the pitch.

In the orientation offset the yaw and pitch represent PERCENTAGES of the camera's horizontal and vertical field of view, respectively. The ROLL (which is in degrees) is applied first, then the yaw, then the pitch.

A typical two screen arrangement would use yaws of 50 and -50 for the left screen, and right screen, respectively. For three screens, the yaws would be 100, 0, and -100 going from left to right.

Using percentages this way makes the screens contiguous even if the camera's field of view changes.

Returns:

0 on success, -1 on failure

5.107.75 function**diguyNetInterface::set_sync_mode_camera_slaved****Prototype:**

```
virtual int set_sync_mode_camera_slaved(int slaved) = 0;
```

Description:

This function sets whether the camera will be slaved when VR-Link is online, in sync mode, and not sync master.

It can be set even when VR-Link is online.

Arguments:

`slaved` whether the camera will be slaved when VR-Link is online, in sync mode, and not sync master.

Returns:

0 on success, -1 on failure

5.107.76 function**diguyNetInterface::get_sync_mode_camera_slaved****Prototype:**

```
virtual int get_sync_mode_camera_slaved() = 0;
```

Description:

This function gets whether the camera is currently slaved. It does NOT simply return what was passed to [set_sync_mode_camera_slaved\(\)](#).

Returns:

Whether the camera is currently slaved. This will be 1 only if VR-link is online, is in sync mode, and is not the sync master, and if `sync_mode_camera_slaved` has been set to 1.

5.107.77 function diguyNetInterface::set_camera_details_out_of_date

Prototype:

```
virtual int set_camera_details_out_of_date() = 0;
```

Description:

Lets VR-Link know that the user has moved the camera, so that if it is in sync mode, and is the sync master, it can let other hosts know the new camera position so they can move their cameras.

Returns:

0 on success, -1 on failure

5.107.78 function diguyNetInterface::set_use_adaptive_guide

Prototype:

```
virtual int set_use_adaptive_guide(int use_adaptive_guide) = 0;
```

Description:

Lets DI-Guy know that it should use an adaptive guide that attempts to fix reckoning errors by stepping. Defaults to on.

Returns:

0 on success, -1 on failure

5.107.79 function diguyNetInterface::get_use_adaptive_guide**Prototype:**

```
virtual int get_use_adaptive_guide() = 0;
```

Returns:

0 if adaptive guide is disabled 1 otherwise.

Experimental Functions

The following functions are experimental and may change or disappear with no warning.

5.107.80 function diguyNetInterface::set_federation_name**Prototype:**

```
virtual int set_federation_name(const char* federation_name);
```

5.107.81 function diguyNetInterface::set_culling_enabled**Prototype:**

```
virtual int set_culling_enabled(int culling);
```

5.107.82 function diguyNetInterface::get_culling_enabled**Prototype:**

```
virtual int get_culling_enabled();
```

5.107.83 function diguyNetInterface::set_network_datetime_enable_diguy_custom_**Prototype:**

```
virtual void set_network_datetime_enable_diguy_custom_pdus(int enable_diguy_custom_p
```

Description:

Sets whether network datetime data will be broadcast via custom PDUs.

NOTE: This is functionality that is under development and should currently not be used by non DI-Guy programmers.

Arguments:

enable_diguy_custom_pdus pass 1 to broadcast network datetime data via custom PDUs, 0 to not

5.107.84 function**diguyNetInterface::get_network_datetime_enable_diguy_custom_****Prototype:**

```
virtual int get_network_datetime_enable_diguy_custom_pdus();
```

Returns:

1 if network datetime data will be broadcast via custom PDUs, 0 if not

5.107.85 function**diguyNetInterface::set_network_datetime_enable_dis_start_resume****Prototype:**

```
virtual void set_network_datetime_enable_dis_start_resume_pdus(int enable_dis_start_resume_pdus);
```

Description:

Sets whether this host will send DIS standard start/resume PDUs.

NOTE: This is functionality that is under development and should currently not be used by non DI-Guy programmers.

Arguments:

enable_dis_start_resume_pdus	pass 1 to send DIS start/resume PDUs, 0 to not

5.107.86 function**diguyNetInterface::get_network_datetime_enable_dis_start_resume****Prototype:**

```
virtual int get_network_datetime_enable_dis_start_resume_pdus();
```

Returns:

1 if DIS start/resume should be send, 0 if not

5.108 DI-Guy API, Particles Module

diguy_module_particles.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

5.108.1 function diguy_particles_initialize

Prototype:

```
int diguy_particles_initialize();
```

5.108.2 function diguy_particles_deinitialize

Prototype:

```
int diguy_particles_deinitialize();
```

5.109 DI-Guy API, Lua Script Module

diguy_module_script_lua.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

◆ [Functions](#)

Link against: libdiguy_module_script_lua

Functions

5.109.1 function diguy_script_lua_initialize

Prototype:

```
int diguy_script_lua_initialize();
```

Description:

This function tells DI-Guy that it should use the Lua interpreter for interpreting scripts. It should be called along with the other `diguy_*_initialize( )` calls, after the primary graphics initialization is made.

Returns:

0 on success, -1 on failure

Example:

```
diguy_ogl_initialize(NULL);  
diguy_script_lua_initialize();
```

5.109.2 function diguy_script_lua_deinitialize

Prototype:

```
int diguy_script_lua_deinitialize();
```

Description:

This function tells DI-Guy that it should shut down the Lua interpreter for interpreting scripts. It should be called along with the other `diguy_*_deinitialize( )` calls, before the primary `diguy_deinitialize( )` call.

Returns:

0 on success, -1 on failure

Example:

```
diguy_script_lua_deinitialize();  
diguy_deinitialize();
```

5.110 DI-Guy API, PhysX Simulation Module

diguy_module_sim_physx.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

- ◆ [Init Structure](#)
- ◆ [Functions](#)

Init Structure

5.110.1 struct diguyModuleSimPhysXInit

```
typedef struct diguyModuleSimPhysXInit
{
    diguyModuleSimPhysXInit::diguyModuleSimPhysXInit()
    :
        physx_px_physics(NULL),
        physx_px_scene(NULL),
        ground_plane_z(0.0f),
        enable_vrd_connection(0)
    {}
    void* physx_px_physics;
    void* physx_px_scene;
    float ground_plane_z;
    int enable_vrd_connection;
} diguyModuleSimPhysXInit;
```

5.110.2 variable diguyModuleSimPhysXInit::physx_px_physics

```
void* physx_px_physics;
```

Description:

An optional pointer to an `physx::PxPhysics` object, if the module should use an already existing PhysX session. If NULL DI-Guy will create a new session.

5.110.3 variable diguyModuleSimPhysXInit::physx_px_scene

```
void* physx_px_scene;
```

Description:

An optional pointer to an `physx::PxScene` object, if the module should use an already existing PhysX scene. If NULL DI-Guy will create a new scene.

5.110.4 variable diguyModuleSimPhysXInit::ground_plane_z

```
float ground_plane_z;
```

Description:

The initial altitude of a ground plane. If this value is set to -10000.0 or lower no ground plane will be enabled.

5.110.5 variable diguyModuleSimPhysXInit::enable_vrd_connection

```
int enable_vrd_connection;
```

Description:

Set to 1 to enable a connection to the PhysX VRD debugger.

Functions

5.110.6 function `diguy_sim_physx_initialize`

Prototype:

```
int diguy_sim_physx_initialize(diguyModuleSimPhysXInit* init);
```

Description:

This function tells DI-Guy that it should use the PhysX simulation library for doing physics simulations on selected characters.

This function must be called after the main DI-Guy initialize call (e.g., `diguy_ogl_initialize()`), and before any scenario is created.

The PhysX system software must be installed on any computer that will use this module. The directory `3rdparty/physx` under the DI-Guy install directory has system software installed. Be sure to read the `README.TXT` file!

See the "Physics Module Functions" section in the [diguyApp](#) documentation for functions that affect the simulation system.

Returns:

0 on success, -1 on failure

Example:

```
diguy_ogl_initialize(NULL);  
  
diguyModuleSimPhysXInit init;  
init.ground_plane_z = 2.0f;  
diguy_sim_physx_initialize(  
diguyScenario* scenario = diguy_create_scenario());
```

5.110.7 function diguy_sim_physx_deinitialize

Prototype:

```
int diguy_sim_physx_deinitialize();
```

Description:

This function tells DI-Guy to shut down the PhysX simulation library module.

Returns:

0 on success, -1 on failure

5.111 DI-Guy API, OpenAL Sound Module

diguy_module_sound_openal.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

◆ [Functions](#)

Link against: libdiguy_module_sound_openal

Regarding Ogg Vorbis software used in this library:

Copyright (c) 2002, Xiph.org Foundation

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the Xiph.org Foundation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Functions

5.111.1 function diguy_sound_openal_initialize

Prototype:

```
int diguy_sound_openal_initialize(bdiSoundInitOpenAL* init);
```

Description:

This function tells DI-Guy that it should use the OpenAL API for playing back sounds.

This function must be called before a scenario is created.

This function must be called after the main DI-Guy initialize call (e.g., `diguy_ogl_initialize()`).

Returns:

0 on success, -1 on failure

Example:

```
bdiSoundInitOpenAL init;
```



```

diguy_ogl_initialize(NULL);
diguy_script_perl_initialize();
diguy_sound_openal_get_defaults(
diguy_sound_openal_initialize(
diguyScenario* scenario = diguy_create_scenario();

```

5.111.2 function diguy_sound_openal_get_defaults

Prototype:

```
int diguy_sound_openal_get_defaults(bdiSoundInitOpenAL* init);
```

Description:

Fills out the init structure with default values. Selected entries can then be changed before calling `diguy_ogl_initialize()`.

Arguments:

`init` pointer to type `bdiGraphicsInitOgl`

Returns:

0 on success, -1 on failure

5.111.3 function diguy_sound_openal_deinitialize

Prototype:

```
int diguy_sound_openal_deinitialize();
```

Description:

This function tells DI-Guy to shut down the OpenAL DirectSound API for playing back sounds.

Returns:

0 on success, -1 on failure

5.112 DI-Guy Release Notes

diguy_release_notes.h

DI-Guy SDK Version 12.5.2

Contents:

- ◆ DI-Guy Version 9 Release Notes
 - ◇ DI-Guy SDK
 - Version 10.0.0
 - Version 9.0.0
 - ◇ DI-Guy Scenario
 - Version 10.0.0
 - Version 9.0.0
 - ◇ DI-Guy AI
 - Version 9.0.0
 - ◇ DI-Guy Networking
 - Version 9.0.0
- ◆ DI-Guy Version 8 Release Notes
 - Version 8.5.0
 - Version 8.0.2
- ◆ DI-Guy Version 7 Release Notes
 - Version 7.0.11
 - Version 7.0.10
 - Version 7.0.9
 - Version 7.0.8
 - Version 7.0.7
 - Version 7.0.6
 - Version 7.0.5
 - Version 7.0.3
 - Version 7.0.2
 - Version 7.0.1
 - Version 7.0.0
- ◆ DI-Guy Version 6 Applications Notes
 - ◇ DI-Guy API
 - Version 6.1.6
 - Version 6.1.4
 - Version 6.0.8
 - Version 6.0.6
 - Version 6.0.1
 - Version 6.0.0
 - ◇ DI-Guy Scenario
 - Version 6.0.8
 - Version 6.0.0
 - ◇ DI-Guy Motion Editor
 - Version 6.0.6
 - ◇ DI-Guy for Vega Prime
 - Version 6.0.8
 - Version 6.0.7

- [Version 6.0.5](#)
- ◆ [DI-Guy Version 6 Data Notes](#)
- [Version 6.1.2](#)
- [Version 6.1.1](#)

Refer to the DI-Guy Changes and Version History document for in-depth change information.

Also refer to the DI-Guy 9.0 New Features document.

DI-Guy is subject to the terms detailed in the license agreement accompanying it.

DI-Guy Version 9 Release Notes

DI-Guy SDK

Version 10.0.0

- ◆ Release notes pending.

Version 9.0.0

- ◆ Added support for the Visual Studio 8 compiler (aka VC++ 2005).
- ◆ Added support for Vega Prime 2.2.

DI-Guy Scenario

Version 10.0.0

- ◆ Release notes pending.

Version 9.0.0

- ◆ Scenarios with long paths will use considerably less memory.
- ◆ DI-Guy now supports multiple active formations.

DI-Guy SDK Version 12.5.2 Index

- ◆ Better support for changing character type and appearance without losing authored content.
- ◆ Added new movie maker; DI-Guy Scenario can now directly output AVIs. (Tiff output is still available; see preferences window.)
- ◆ New munition system. It is now possible to specify different munition types, with an associated DIS identifier. Munition detonations can have radius effects, and will report impact callbacks.
- ◆ Added ability to cull characters that are far away from the camera.
- ◆ Scene culling in worlds with xrefs and rotated objects is handled more efficiently.

DI-Guy AI

Version 9.0.0

- ◆ Improved user interface, stability, added a number of query functions for developing more elaborate responses with decision beads.
- ◆ Improvements to scene object grid framework to allow region painting and better collision detection.
- ◆ DI-Guy SAF has been renamed to DI-Guy AI.

DI-Guy Networking

Version 9.0.0

- ◆ Added support for Flat Earth and Geocentric coordinates.
- ◆ Munition system improvements, see above.
- ◆ Added diguy_module_net.h, replaces the deprecated diguy_module_hla.h, and diguy_module_dis.h header files.

DI-Guy Version 8 Release Notes

Version 8.5.0

- ◆ Initial DI-Guy SAF release.
- ◆ IMPORTANT CHANGES FROM EARLIER VERSIONS: It is intended that DI-Guy is always fully backwards compatible. However, the following two items are areas where backwards compatibility has been sacrificed for improved DI-Guy functionality:
 - ◇ The default aim algorithm for aiming characters is now algorithm 8 instead of algorithm 7. In almost all cases version 8 exhibits better behavior, but if it breaks some aspect of your scenario you can set the aim algorithm back to version 7 by calling `diguyCharacter::set_aim_algorithm()`.

Version 8.0.2

- ◆ Initial DI-Guy 8.0 release. See DI-Guy 8.0 New Features document.
- ◆ IMPORTANT CHANGES FROM EARLIER VERSIONS: It is intended that DI-Guy is always fully backwards compatible. However, the following two items are areas where backwards compatibility has been sacrificed for improved DI-Guy functionality:
 - ◇ DI-Guy characters now fire live rounds by default when functions such as `diguyCharacter::fire_weapon()` are called. The old behavior can be restore in two ways:
 - In DI-Guy Scenario, the Weapon Params tab on the Character Elements palette has a checkbox called "Weapon Fires Live Rounds". The checkbox can be unchecked, followed by pressing the "Copy to All Characters" button.
 - In a program using the DI-Guy API, the function `diguyCharacter::set_weapon_fires_live_rounds()` can be called.
 - ◇ The concept of a non-zero Tin ("T In") value for scenarios is no longer supported. There is no work-around for this, except to subtract out the old Tin value from the scenario Tout value. Scenarios that have used a non-zero Tin value will have to be changed.
 - ◇ Removed function `diguy_dis_get_action_from_entity_state()` from `diguy_module_dis.h`. `diguyCharacter::get_action_from_entity_state()` should be called instead to map DIS values to an action.
- ◆ DI-Guy 8 for Vega Prime is intended to be compatible with Vega Prime 2.1 and vpNet 2.1.1.
- ◆ We have experienced issues with 7900 GTX graphics cards when running DI-Guy Scenario. These include GL_SELECT being performed in software instead of hardware, and issues with DI-Guy Scenario being recognized as a graphics program.

These issues were resolved by rolling back drivers from the latest update from the chipset manufacturer to the driver version provided by the computer manufacturer. If you encounter unusual performance problems with DI-Guy when migrating to such a machine, please let us know so that we can improve our support and understanding of this issue.

DI-Guy Version 7 Release Notes

Version 7.0.11

- ◆ Enabled DI-Guy for Vega Prime to interoperate successfully with all the different earth models provided in vpNet, including Flat Earth.

Version 7.0.10

- ◆ Upgraded VR-Link version to 3.9.6. This also means that DI-Guy for Vega Prime customers should use vpNet 2.1 with Vega Prime 2.0.

Version 7.0.9

- ◆ Enabled DI-Guy Scenario to broadcast in 1278.1 format by setting "protocol_version" in the network.cfg file to "5".
- ◆ Improved behavior of DI-Guy characters when networking and varying character speed.
- ◆ Enhanced DI-Guy characters' abilities to find the best action when driven from the network using standard PDUs.
- ◆ Fixed problem with hands on some flight deck characters; increased available appearances for flight deck characters.

Version 7.0.8

- ◆ Fixed "ignore script events" checkbox in Camera Control of DI-Guy Scenario. Improved handling of bogus numeric input.

DI-Guy SDK Version 12.5.2 Index

- ◆ Removed VR-Link license dependency from DI-Guy for Vega Prime.
- ◆ Added documentation for the `vpDiguy` classes.
- ◆ Added static libraries to distribution for DI-Guy for Vega Prime.
- ◆ Added example for creating a `vpDiguyCharacter` at runtime.
- ◆ Fixed glitches when creating and deleting particle descriptions.
- ◆ Fixed minor bug with opening textures in DI-Guy Scenario.
- ◆ Improved network settings page responsiveness in DI-Guy Scenario.

Version 7.0.7

- ◆ Formalized support for just-released `vpNet 2.0` in DI-Guy for Vega Prime. NOTE: In order to increase reliability of `vpNet` interaction, the reading of UTM coordinates out of scenario objects (terrains) has been disabled in `vpDiguy`. It is recommended that the UTM coordinates for a simulation be explicitly defined in the `vpNet` Connection `gcf` panel.

Version 7.0.6

- ◆ Fixed problem where character's initial path has incorrect name.
- ◆ Added fixes for destruction and reference counting in DI-Guy for Vega Prime.

Version 7.0.5

- ◆ Fixed a crash that sometimes happened while reading comments out of texture `.attr` files.

Version 7.0.3

- ◆ Fixed a problem with the `diguyGraphicsShape::update()` function not always being called.
- ◆ Fixed various bugs with VR-Link and DI-Guy time sync code.
- ◆ Added warning if `diguyCharacter::set_desired_action()` is called with `0.0f` as the speed.

Version 7.0.2

- ◆ Fixed a problem with gazing and aiming not zeroing in on the correct positions when using DI-Guy for Vega Prime.

Version 7.0.1

- ◆ Deprecated "named_callback" functions in `diguyApp`. Event handler functions should be used instead.
- ◆ Renamed some `diguyViewCamera` functions:
 - ◇ `diguyViewCamera::get_world_offset_position()` has become `diguyViewCamera::get_segment_origin`
 - ◇ `diguyViewCamera::get_local_position()` has become `diguyViewCamera::get_position_segment_offset()`
 - ◇ `diguyViewCamera::get_local_fix()` has become `diguyViewCamera::get_fix_segment_offset()`
- ◆ Added function `diguyCharacterGroup::update_all_members()`, and deprecated function `diguyCharacterGroup::set_all_members_t()`
- ◆ Fixed problem with deleting one scenario and creating another causing a crash.
- ◆ Fixed crash problem with function `diguy_set_cfg_file()`.

- ◆ Changed the default settings of automatic switching of graphics LODs and motion LODs, as set by `diguyScenario::set_automatic_graphics_lod_switching()` and `diguyScenario::set_automatic_motion_lod_switching()`. Previously automatic LOD switching was always enabled unless explicitly disabled by calling the above functions. Now automatic LOD switching will be disabled by default in DI-Guy API applications, but remain enabled by default in DI-Guy Scenario.
- ◆ Added the following functions for adding and removing appearances:
 - ◇ `diguyScenario::add_appearance()`
 - ◇ `diguyScenario::add_appearance_based_on_existing_appearance()`
 - ◇ `diguyScenario::remove_appearance()`
- ◆ `diguyScenario::add_custom_appearance()` deprecated; use `diguyScenario::add_appearance()`.
- ◆ `diguyScenario::add_custom_appearance_based_on_existing_appearance()` deprecated; use `diguyScenario::add_appearance_based_on_existing_appearance()`.
- ◆ Promoted the following functions from experimental to official status:
 - ◇ `diguyCharacter::get_label_text()`
 - ◇ `diguyCharacter::set_label_text()`
 - ◇ `diguyCharacter::get_label_is_visible()`
 - ◇ `diguyCharacter::set_label_is_visible()`
 - ◇ `diguyCharacter::set_label_text_color()`
 - ◇ `diguyCharacter::get_label_text_color()`
 - ◇ `diguyCharacter::set_label_background_color()`
 - ◇ `diguyCharacter::get_label_background_color()`
- ◆ Removed `diguyLoadManager` camera functions (load manager camera now always tracks the scenario's primary view camera):
 - ◇ `diguyLoadManager::add_camera()`
 - ◇ `diguyLoadManager::add_view_camera()`
 - ◇ `diguyLoadManager::move_camera()`
 - ◇ `diguyLoadManager::change_view_camera_settings()`
 - ◇ `diguyLoadManager::remove_camera()`
 - ◇ `diguyLoadManager::remove_all_cameras()`

Version 7.0.0

- ◇ Fixed problem with `diguyCharacter::set_speed()`. The set speed was being reset to the action's default speed after a short time, which is not correct behavior.
- ◇ Changed values returned by `diguyCharacter::get_aim_azimuth()` and `diguyCharacter::get_aim_elevation()` to be character-relative rather than world-relative.
- ◇ New character types:
 - `chem_bio_pedestrian`

DI-Guy SDK Version 12.5.2 Index

- `fdc_chock_and_chains`
- `fdc_fuelman`
- `fdc_grounding`
- `fdc_pallet`
- `fdc_pedestrian`
- `firefighter`
- `fireman_silver`
- `vehicle_wheels`

DI-Guy Version 6 Applications Notes

DI-Guy API

Note that most changes to the DI-Guy API also apply to DI-Guy Scenario.

Version 6.1.6

- ◇ Removed unused variable "working_directory" from `bdiExFaceInit` structure. Replaced with variable "placeholder", leaving the option open for future init variables. Calls passing NULL for the `bdiExFaceInit` structure to `diguy_exface_initialize( )` will not need to change.

Version 6.1.4

- ◇ Removed unused functions from `diguySensorRegion` class:
 - `diguySensorRegion::get_characters_in_region( )`
 - `diguySensorRegion::update_characters_in_region( )`
 - `diguySensorRegion::update_candidate_characters_in_region( )`

Version 6.0.8

DI-Guy SDK Version 12.5.2 Index

- For Win32, DI-Guy DLLs now include version information in their filenames, to make it easier for multiple versions of DI-Guy to exist side-by-side.

Version 6.0.6

- ◇ Made all characters default to not using stepping gaze. This fixes a backwards incompatibility problem where some version 5 characters that were gazing would leave their path on version 6 and thus not complete the actions specified by their path.

Version 6.0.1

- ◇ Removed deprecated **diguyCharacter** functions:
 - diguyCharacter::force_action_by_name()
 - diguyCharacter::get_desired_action_name()
 - diguyCharacter::set_desired_action_by_name()
 - diguyCharacter::force_partial_path_by_name()
 - diguyCharacter::force_path_by_name()
 - diguyCharacter::push_path_by_name()
 - diguyCharacter::set_parent_by_name()
- ◇ Removed deprecated **diguySensorRegion** function
diguySensorRegion::contains_character_by_name().
- ◇ Removed deprecated **diguySignal** function
diguySignal::play_sound().
- ◇ Removed deprecated **diguyScenario** function
diguyScenario::find_or_create_face_expression().

Version 6.0.0

- ◇ Users who are upgrading their earlier versions of DI-Guy should uninstall their existing DI-Guy Applications, DI-Guy Data, (and DI-Guy for Vega Prime for Vega Prime users), and then install the 6.0 versions.
- ◇ Uninstalling will not delete your existing customizations located in the "custom" folder. You can copy that folder to the new installation.
- ◇ Users with FLOATING LICENSES **must** upgrade to the latest version of the license manager. The necessary files and documentation are in the folder
%DIGUY%\3rdparty\flexlm\diguy_flexlm_10.1_win32. Existing DI-Guy 5 license files **will** work with the new license manager. To upgrade your license server:
 - Shutdown the old license server:
 - Start the OLD lmttools.
 - On the Server/License file tab, select the OLD DI-Guy license server
 - On the StartStopReread tab, click "Stop Server"

DI-Guy SDK Version 12.5.2 Index

- On the Config Services tab, click "Remove Service"
 - Exit lmttools
-
- To be really certain the old server is gone, you can check the services control panel applet and verify the DI-Guy license server is no longer there. To be compulsively certain, reboot the server and make sure DI-Guy license server doesn't come back.
 - Remove the old license server files: lmttools.exe, lmgrd.exe, bdilm.exe. DO NOT Delete the license file.
 - Copy
%DIGUY%\3rdparty\flexlm\diguy_flexlm_10.1_win32
to the license server computer
 - Copy the license file to the new folder.
 - Start the NEW license service with the new license manager daemon and the copied license file. Follow the instructions in
%DIGUY%\3rdparty\flexlm\diguy_flexlm_10.1_win32\diguy-win32-server.txt.
 - Test by running a DI-Guy app on a client.
-
- ◇ Removed support for character history types
DIGUY_HISTORY_TYPE_FIRST and
DIGUY_HISTORY_TYPE_LAST. Characters saved using these history types will have their history type set to
DIGUY_HISTORY_TYPE_COMPLETE.

DI-Guy Scenario

Note that most changes to the DI-Guy API also apply to DI-Guy Scenario.

Version 6.0.8

- ◇ In DI-Guy Scenario, incoming networked characters will now automatically be assigned to groups "net_opposing", "net_friendly", or "net_neutral" based on their network force id. Outgoing networked characters will check if a character belongs to such a group when setting its network force id.

Version 6.0.0

- ◇ DI-Guy Scenario may play sounds too fast on some PCs equipped with hardware sound accelerators. (This may also be true for DI-Guy SDK programs using diguy_module_sound_gap). We are working on the problem. A work-around is to reduce the level of hardware acceleration for audio playback. To change the acceleration:
- Open Control Panel.
 - Open Sounds and Multimedia.
 - Open the "Audio" tab.
 - Under "Sound Playback" click "Advanced".

DI-Guy SDK Version 12.5.2 Index

- Open the "Performance" tab.
- Move the "Hardware acceleration" slider to the left.
- Click Okay to close "Advanced Audio Properties".
- Click Okay to close "Sounds and Multimedia Properties".
- Test to see if sounds play correctly.

DI-Guy Motion Editor

Version 6.0.6

- ◇ Fixed DI-Guy Motion Editor bug where some repeated actions caused performance degradation.
- ◇ Enhanced DI-Guy Motion Editor importing to support non-base 6-DOF joints.
- ◇ Made all characters default to not using stepping gaze. This fixes a backwards incompatibility problem where some version 5 characters that were gazing would leave their path on version 6 and thus not complete their performance.

DI-Guy for Vega Prime

Version 6.0.8

- ◇ Fixed problems to improve interaction with HLA – update recommended for HLA Vega Prime users.

Version 6.0.7

- ◇ Fixed problems with dumping .cpp from LynxPrime using DI-Guy for Vega Prime.
- ◇ Fixed bug where isectors did not work on some DI-Guy characters in Vega Prime.
- ◇ Fixed bug where including DI-Guy in a Vega Prime application would block keystrokes being sent through the display window.
- ◇ Two new examples, "view_settings" and "simple" have been added to coding examples for DI-Guy for Vega Prime.

BACKWARDS INCOMPATIBILITY NOTICE FOR DI-GUY FOR VEGA PRIME:

ACF files made using early versions of DI-Guy 6 will require users to find all strings in the file that use "DI-Guy" and replace them with "Diguy". The vpDiguy.h and vpDiguy classes have been replaced with vpDiguySettings.h and the vpDiguySettings class.

Version 6.0.5

DI-Guy for Vega Prime now works with DIS/HLA for Vega Prime. As of version 6.0.5, DI-Guy for Vega Prime works with Vega Prime 1.2.2 from MultiGen Paradigm and vpNet 1.2.2 from MAK Technologies.

DI-Guy Version 6 Data Notes

Version 6.1.2

- ◇ Fixed problem with config file appearance syntax. The item "shaperset:shape" format caused a crash; it now works again.
- ◇ Replaced default scene object (*setsdefault_stagedefault_stage.flt*) with new default (*setsdefault_stage7flt/default_stage7.flt*). The new *default_stage7.flt* scene object has a number of different areas in which example scenarios can be placed.

Version 6.1.1

- ◇ Added new appearance "invisible_human"; Using this appearance for a human character will remove all geometry from the character, effectively making it invisible.

5.113 diguy_typedefs.h

DI-Guy SDK Version 12.5.2

Contents:

- ◆ [Character Typedefs.](#)
- ◆ [Scenario Typedefs.](#)
- ◆ [Other DI-Guy Object Typedefs](#)
- ◆ [Module Library Function Typedefs.](#)
- ◆ [Graphics API Typedefs.](#)
- ◆ [Deprecated Typedefs.](#)

Character Typedefs.

Scenario Typedefs.

Other DI-Guy Object Typedefs

Module Library Function Typedefs.

5.113.1 function int

Prototype:

```
typedef int (*diguyModuleFunctionStandard());
```

5.113.2 function void

Prototype:

```
typedef void (*diguyModuleFunctionReturnsVoid)();
```

5.113.3 function int

Prototype:

```
typedef int (*diguyModuleFunctionArgumentConstChar)(const char*);
```

5.113.4 function int

Prototype:

```
typedef int (*diguyModuleFunctionArgumentCharStarStar)(char**);
```

5.113.5 function int

Prototype:

```
typedef int (*diguyModuleFunctionArgument2Ints)(int, int);
```

5.113.6 function int

Prototype:

```
typedef int (*diguyModuleFunctionArgument2Doubles)(double, double);
```

5.113.7 function int

Prototype:

```
typedef int (*diguyModuleFunctionArgument2DoublePtrs)(double*, double*);
```

5.113.8 function int

Prototype:

```
typedef int (*diguyModuleFunctionArgument3Floats)(float, float, float);
```

5.113.9 function int

Prototype:

```
typedef int (*diguyModuleFunctionArgument3FloatPtrs)(float*, float*, float*);
```

5.113.10 function int

Prototype:

```
typedef int (*diguyModuleFunctionArgument4Floats)(float, float, float, float);
```

5.113.11 function int

Prototype:

```
typedef int (*diguyModuleFunctionArgument4FloatPtrs)(float*, float*, float*, float*);
```

Graphics API Typedefs.

Deprecated Typedefs.

5.113.12 function int**Prototype:**

```
typedef int (*diguyUserQueryYesNoFunction)(diguyVariable* result_variable,
    int pause_scenario,
    const char* caption,
    const char* text,
    const char* button0,
    const char* button1);
```

5.113.13 function int**Prototype:**

```
typedef int (*diguyUserQueryItemFunction)(diguyVariable* result_variable,
    int pause_scenario,
    const char* caption,
    const char* question,
    const char* arg1,
    const char* arg2, // need at least 2 arguments
    const char* arg3,
    const char* arg4,
    const char* arg5,
    const char* arg6,
    const char* arg7,
    const char* arg8,
    const char* arg9,
    const char* arg10);
```


5.113.14 function int**Prototype:**

```
typedef int (*diguyUserInformFunction)(int pause_scenario,  
    const char* caption,  
    const char* text);
```

5.113.15 function int**Prototype:**

```
typedef int (*diguyStudentLogFunction)(int notify_level, const char* text);
```

5.114 DI-Guy Vector Classes

diguy_vector_classes.h

DI-Guy SDK Version 12.5.2

Includes: declspec_diguy.h

Contents:

Link against: libdiguy

5.114.1 class diguyVec2f

```
class BDI_DECLSPEC_diguy diguyVec2f
{
public:
    inline diguyVec2f() {x = y = 0.0f;}
    inline diguyVec2f(float new_x) {x = y = new_x;}
    inline diguyVec2f(float new_x, float new_y) {x = new_x; y = new_y;}
    inline float _x() {return x;}
    inline float _y() {return y;}
    inline float _azimuth() {return azimuth;}
    inline float _elevation() {return elevation;}
#ifdef CPLUSPLUS_ONLY
    union {
        float x;
        float azimuth;
    };
    union {
        float y;
        float elevation;
    };
#endif
};
```

5.114.2 function diguyVec2f::diguyVec2f

Prototype:

```
inline diguyVec2f();
```

5.114.3 function diguyVec2f::diguyVec2f

Prototype:

```
inline diguyVec2f(float new_x);
```

5.114.4 function diguyVec2f::diguyVec2f

Prototype:

```
inline diguyVec2f(float new_x, float new_y);
```

5.114.5 variable diguyVec2f::x

```
float x;
```

5.114.6 variable diguyVec2f::azimuth

```
float azimuth;
```

5.114.7 variable diguyVec2f::y

```
float y;
```

5.114.8 variable diguyVec2f::elevation

```
float elevation;
```

5.114.9 class diguyVec3f

```
class BDI_DECLSPEC diguy diguyVec3f
{
public:
    inline diguyVec3f() {x = y = z = 0.0f;}
    inline diguyVec3f(float new_x) {x = y = z = new_x;}
    inline diguyVec3f(float new_x, float new_y, float new_z) {x = new_x; y = new_y; z = new_z;}
    diguyVec3f diguy_to_y_forward_z_up_coords();
    diguyVec3f y_forward_z_up_to_diguy_coords();
    diguyVec3f diguy_to_z_forward_y_up_coords(); // diguy coords to unity coords
    diguyVec3f z_forward_y_up_to_diguy_coords(); // unity coords to diguy coords
    diguyVec4f diguy_euler_angles_to_quaternion() const;
    inline float _x() {return x;}
    inline float _y() {return y;}
    inline float _z() {return z;}
    inline float _rz() {return rz;}
    inline float _rx() {return rx;}
    inline float _ry() {return ry;}
    inline float _r() {return r;}
    inline float _g() {return g;}
    inline float _b() {return b;}
    inline float _azimuth() {return azimuth;}
    inline float _elevation() {return elevation;}
    inline float _radius() {return radius;}
#ifdef CPLUSPLUS_ONLY
    union {
```

```
    float x;  
    float rz;  
    float r;  
    float azimuth;  
};  
union {  
    float y;  
    float rx;  
    float g;  
    float elevation;  
};  
union {  
    float z;  
    float ry;  
    float b;  
    float radius;  
};  
#endif  
};
```

5.114.10 function diguyVec3f::diguyVec3f

Prototype:

```
inline diguyVec3f();
```

5.114.11 function diguyVec3f::diguyVec3f**Prototype:**

```
inline diguyVec3f(float new_x);
```

5.114.12 function diguyVec3f::diguyVec3f**Prototype:**

```
inline diguyVec3f(float new_x, float new_y, float new_z);
```


5.114.13 function diguyVec3f::diguy_to_y_forward_z_up_coords**Prototype:**

```
diguyVec3f diguy_to_y_forward_z_up_coords();
```

5.114.14 function diguyVec3f::y_forward_z_up_to_diguy_coords**Prototype:**

```
diguyVec3f y_forward_z_up_to_diguy_coords();
```

5.114.15 function diguyVec3f::diguy_to_z_forward_y_up_coords**Prototype:**

```
diguyVec3f diguy_to_z_forward_y_up_coords(); // diguy coords to unity coords
```

5.114.16 function diguyVec3f::z_forward_y_up_to_diguy_coords**Prototype:**

```
diguyVec3f z_forward_y_up_to_diguy_coords(); // unity coords to diguy coords
```

5.114.17 function diguyVec3f::diguy_euler_angles_to_quaternion**Prototype:**

```
diguyVec4f diguy_euler_angles_to_quaternion() const;
```

5.114.18 variable diguyVec3f::x

```
float x;
```

5.114.19 variable diguyVec3f::rz

```
float rz;
```

5.114.20 variable diguyVec3f::r

```
float r;
```

5.114.21 variable diguyVec3f::azimuth

```
float azimuth;
```

5.114.22 variable diguyVec3f::y

```
float y;
```

5.114.23 variable diguyVec3f::rx

```
float rx;
```

5.114.24 variable diguyVec3f::g

```
float g;
```

5.114.25 variable diguyVec3f::elevation

```
float elevation;
```

5.114.26 variable diguyVec3f::z

```
float z;
```

5.114.27 variable diguyVec3f::ry

```
float ry;
```

5.114.28 variable diguyVec3f::b

```
float b;
```


5.114.29 variable `diguyVec3f::radius`

```
float radius;
```

5.114.30 class `diguyVec4f`

```
class BDI_DECLSPEC_diguy diguyVec4f
{
public:
    inline diguyVec4f() {x = y = z = w = 0.0f;}
    inline diguyVec4f(float new_x) {x = y = z = w = new_x;}
    inline diguyVec4f(float new_x, float new_y, float new_z, float new_w) {x = new_x; y =
    inline float _x() {return x;}
    inline float _y() {return y;}
    inline float _z() {return z;}
    inline float _w() {return w;}
    inline float _rz() {return rz;}
    inline float _rx() {return rx;}
    inline float _ry() {return ry;}
    inline float _rw() {return rw;}
    inline float _r() {return r;}
    inline float _g() {return g;}
    inline float _b() {return b;}
    inline float _a() {return a;}
    inline float _qx() {return qx;}
    inline float _qy() {return qy;}
    inline float _qz() {return qz;}
    inline float _qw() {return qw;}
#ifdef CPLUSPLUS_ONLY
    union {
        float x;
```

```

        float rz;
        float r;
        float qx;
    };
    union {
        float y;
        float rx;
        float g;
        float qy;
    };
    union {
        float z;
        float ry;
        float b;
        float qz;
    };
    union {
        float w;
        float rw;
        float a;
        float qw;
    };
#endif
};

```

5.114.31 function diguyVec4f::diguyVec4f

Prototype:

```
inline diguyVec4f();
```

5.114.32 function diguyVec4f::diguyVec4f**Prototype:**

```
inline diguyVec4f(float new_x);
```

5.114.33 function diguyVec4f::diguyVec4f**Prototype:**

```
inline diguyVec4f(float new_x, float new_y, float new_z, float new_w);
```

5.114.34 variable diguyVec4f::x

```
float x;
```

5.114.35 variable diguyVec4f::rz

```
float rz;
```

5.114.36 variable diguyVec4f::r

```
float r;
```

5.114.37 variable diguyVec4f::qx

```
float qx;
```

5.114.38 variable `diguyVec4f::y`

```
float y;
```

5.114.39 variable `diguyVec4f::rx`

```
float rx;
```

5.114.40 variable diguyVec4f::g

```
float g;
```

5.114.41 variable diguyVec4f::qy

```
float qy;
```

5.114.42 variable diguyVec4f::z

```
float z;
```

5.114.43 variable diguyVec4f::ry

```
float ry;
```


5.114.44 variable diguyVec4f::b

```
float b;
```

5.114.45 variable diguyVec4f::qz

```
float qz;
```

5.114.46 variable diguyVec4f::w

```
float w;
```

5.114.47 variable diguyVec4f::rw

```
float rw;
```

5.114.48 variable diguyVec4f::a

```
float a;
```

5.114.49 variable diguyVec4f::qw

```
float qw;
```

5.114.50 class diguyVec3d

```

class BDI_DECLSPEC_diguy diguyVec3d
{
public:
    inline diguyVec3d() {x = y = z = 0.0;}
    inline diguyVec3d(double new_x) {x = y = z = new_x;}
    inline diguyVec3d(double new_x, double new_y, double new_z) {x = new_x; y = new_y; z = new_z;}
    inline double _x() {return x;}
    inline double _y() {return y;}
    inline double _z() {return z;}
    inline double _rz() {return rz;}
    inline double _rx() {return rx;}
    inline double _ry() {return ry;}
    inline double _r() {return r;}
    inline double _g() {return g;}
    inline double _b() {return b;}
    inline double _azimuth() {return azimuth;}
    inline double _elevation() {return elevation;}
    inline double _radius() {return radius;}
#ifdef CPLUSPLUS_ONLY
    union {
        double x;
        double rz;
        double r;
        double azimuth;
    };
    union {
        double y;
        double rx;
        double g;
        double elevation;
    };
    union {
        double z;
        double ry;
        double b;
        double radius;
    };
#endif
};

```

5.114.51 function diguyVec3d::diguyVec3d**Prototype:**

```
inline diguyVec3d();
```

5.114.52 function diguyVec3d::diguyVec3d**Prototype:**

```
inline diguyVec3d(double new_x);
```

5.114.53 function diguyVec3d::diguyVec3d**Prototype:**

```
inline diguyVec3d(double new_x, double new_y, double new_z);
```

5.114.54 variable diguyVec3d::x

```
double x;
```

5.114.55 variable diguyVec3d::rz

```
double rz;
```

5.114.56 variable diguyVec3d::r

```
double r;
```

5.114.57 variable diguyVec3d::azimuth

```
double azimuth;
```

5.114.58 variable diguyVec3d::y

```
double y;
```


5.114.59 variable diguyVec3d::rx

```
double rx;
```

5.114.60 variable diguyVec3d::g

```
double g;
```

5.114.61 variable diguyVec3d::elevation

```
double elevation;
```

5.114.62 variable diguyVec3d::z

```
double z;
```

5.114.63 variable diguyVec3d::ry

```
double ry;
```

5.114.64 variable diguyVec3d::b

```
double b;
```

5.114.65 variable diguyVec3d::radius

```
double radius;
```

5.114.66 class diguyVec4d

```
class BDI_DECLSPEC diguy diguyVec4d
{
public:
    inline diguyVec4d() {x = y = z = w = 0.0;}
    inline diguyVec4d(double new_x) {x = y = z = w = new_x;}
    inline diguyVec4d(double new_x, double new_y, double new_z, double new_w) {x = new_x;
    inline double _x() {return x;}
    inline double _y() {return y;}
    inline double _z() {return z;}
    inline double _w() {return w;}
    inline double _rz() {return rz;}
    inline double _rx() {return rx;}
    inline double _ry() {return ry;}
    inline double _rw() {return rw;}
    inline double _r() {return r;}
    inline double _g() {return g;}
    inline double _b() {return b;}
    inline double _a() {return a;}
    inline double _qx() {return qx;}
    inline double _qy() {return qy;}
    inline double _qz() {return qz;}
    inline double _qw() {return qw;}
#ifdef CPLUSPLUS_ONLY
    union {
        double x;
```

```
        double rz;  
        double r;  
        double qx;  
    };  
    union {  
        double y;  
        double rx;  
        double g;  
        double qy;  
    };  
    union {  
        double z;  
        double ry;  
        double b;  
        double qz;  
    };  
    union {  
        double w;  
        double rw;  
        double a;  
        double qw;  
    };  
};  
#endif  
};
```

5.114.67 function diguyVec4d::diguyVec4d

Prototype:

```
inline diguyVec4d();
```

5.114.68 function diguyVec4d::diguyVec4d**Prototype:**

```
inline diguyVec4d(double new_x);
```

5.114.69 function diguyVec4d::diguyVec4d**Prototype:**

```
inline diguyVec4d(double new_x, double new_y, double new_z, double new_w);
```

5.114.70 variable diguyVec4d::x

```
double x;
```

5.114.71 variable diguyVec4d::rz

```
double rz;
```

5.114.72 variable diguyVec4d::r

```
double r;
```

5.114.73 variable diguyVec4d::qx

```
double qx;
```


5.114.74 variable diguyVec4d::y

```
double y;
```

5.114.75 variable diguyVec4d::rx

```
double rx;
```

5.114.76 variable diguyVec4d::g

```
double g;
```

5.114.77 variable diguyVec4d::qy

```
double qy;
```

5.114.78 variable diguyVec4d::z

```
double z;
```

5.114.79 variable diguyVec4d::ry

```
double ry;
```

5.114.80 variable diguyVec4d::b

```
double b;
```

5.114.81 variable diguyVec4d::qz

```
double qz;
```

5.114.82 variable diguyVec4d::w

```
double w;
```

5.114.83 variable diguyVec4d::rw

```
double rw;
```

5.114.84 variable diguyVec4d::a

```
double a;
```

5.114.85 variable diguyVec4d::qw

```
double qw;
```

5.114.86 class diguyVec3i

```

class BDI_DECLSPEC_diguy diguyVec3i
{
public:
    inline diguyVec3i() {x = y = z = 0;}
    inline diguyVec3i(int new_x) {x = y = z = new_x;}
    inline diguyVec3i(int new_x, int new_y, int new_z) {x = new_x; y = new_y; z = new_z;}
    inline int _x() {return x;}
    inline int _y() {return y;}
    inline int _z() {return z;}
    inline int _r() {return r;}
    inline int _g() {return g;}
    inline int _b() {return b;}
#ifdef CPLUSPLUS_ONLY
    union {
        int x;
        int r;
    };
    union {
        int y;
        int g;
    };
    union {
        int z;
        int b;
    };
#endif
};

```

5.114.87 function diguyVec3i::diguyVec3i**Prototype:**

```
inline diguyVec3i();
```

5.114.88 function diguyVec3i::diguyVec3i**Prototype:**

```
inline diguyVec3i(int new_x);
```

5.114.89 function diguyVec3i::diguyVec3i**Prototype:**

```
inline diguyVec3i(int new_x, int new_y, int new_z);
```


5.114.90 variable diguyVec3i::x

```
int x;
```

5.114.91 variable diguyVec3i::r

```
int r;
```

5.114.92 variable diguyVec3i::y

```
int y;
```

5.114.93 variable diguyVec3i::g

```
int g;
```

5.114.94 variable diguyVec3i::z

```
int z;
```

5.114.95 variable diguyVec3i::b

```
int b;
```

5.114.96 class diguyVec4i

```

class BDI_DECLSPEC_diguy diguyVec4i
{
public:
    inline diguyVec4i() {x = y = z = w = 0;}
    inline diguyVec4i(int new_x) {x = y = z = w = new_x;}
    inline diguyVec4i(int new_x, int new_y, int new_z, int new_w) {x = new_x; y = new_y; z = new_z; w = new_w;}
    inline int _x() {return x;}
    inline int _y() {return y;}
    inline int _z() {return z;}
    inline int _w() {return w;}
    inline int _r() {return r;}
    inline int _g() {return g;}
    inline int _b() {return b;}
    inline int _a() {return a;}
#ifdef CPLUSPLUS_ONLY
    union {
        int x;
        int r;
    };
    union {
        int y;
        int g;
    };
    union {
        int z;
        int b;
    };
    union {
        int w;
        int a;
    };
#endif
};

```

5.114.97 function diguyVec4i::diguyVec4i**Prototype:**

```
inline diguyVec4i();
```

5.114.98 function diguyVec4i::diguyVec4i**Prototype:**

```
inline diguyVec4i(int new_x);
```

5.114.99 function diguyVec4i::diguyVec4i**Prototype:**

```
inline diguyVec4i(int new_x, int new_y, int new_z, int new_w);
```

5.114.100 variable diguyVec4i::x

```
int x;
```

5.114.101 variable diguyVec4i::r

```
int r;
```

5.114.102 variable diguyVec4i::y

```
int y;
```

5.114.103 variable diguyVec4i::g

```
int g;
```

5.114.104 variable diguyVec4i::z

```
int z;
```


5.114.105 variable diguyVec4i::b

```
int b;
```

5.114.106 variable diguyVec4i::w

```
int w;
```

5.114.107 variable diguyVec4i::a

```
int a;
```

5.115 Boston Dynamics Expressive Face Init Structure

bdiFaceFxInit.h

DI-Guy SDK Version 12.5.2

Contents:

5.115.1 struct bdiFaceFxInit

```
typedef struct bdiFaceFxInit
{
    int placeholder;
} bdiFaceFxInit;
```

5.115.2 variable bdiFaceFxInit::placeholder

```
int placeholder;
```

5.116 Boston Dynamics Sound Init Structure, OpenAL

bdiSoundInitOpenAL.h

DI-Guy SDK Version 12.5.2

Contents:

5.116.1 struct bdiSoundInitOpenAL

```
typedef struct bdiSoundInitOpenAL
{
    int call_alutInit_and_alutExit_flag;
    int enable_3d_audio;
} bdiSoundInitOpenAL;
```

5.116.2 variable bdiSoundInitOpenAL::call_alutInit_and_alutExit_flag

```
int call_alutInit_and_alutExit_flag;
```

Description:

Whether the module should call alutInit() and alutExit(); Set to 1 (the default) unless this is handled explicitly by the application.

5.116.3 variable bdiSoundInitOpenAL::enable_3d_audio

```
int enable_3d_audio;
```

Description:

Flag determining whether 3D positional audio is enabled. Set to 1 (the default) to enable 3D positional audio; set to 0 to disable it.

5.117 DI-Guy Graphics Init Structure, Graphics API

bdiGraphicsInitGraphicsAPI.h

DI-Guy SDK Version 12.5.2

Includes: [diguyGraphicsVertexFormats.h](#)

Contents:

5.117.1 struct bdiGraphicsInitGraphicsAPI

```
typedef struct bdiGraphicsInitGraphicsAPI
{
    int use_array_format_for_vertex_buffer_data;
    diguyGraphicsIndexType default_vertex_type;
    int use_shaders;
    diguyGraphicsShaderLanguage shader_language;
    int transpose_shader_matrices;
    bool use_4x3_shader_matrices;
    int use_cached_optimized_geometry;
    int use_cached_compressed_textures;

    int use_sRGB_corrected_textures;

    int use_linear_traversal_draw;
    int characters_have_world_space_triangle_query;
    int scene_objects_have_world_space_triangle_query;
} bdiGraphicsInitGraphicsAPI;
```

5.117.2 variable**bdiGraphicsInitGraphicsAPI::use_array_format_for_vertex_buffer**

```
int use_array_format_for_vertex_buffer_data;
```

Description:

If vertex buffer data is not in array form, data will be packed in interleaved form like so:

```
vnt vnt vnt vnt vnt vnt
```

Where 'v' represents three floats specifying vertex position, 'n' represents three floats specifying normal direction, and 't' represents two or more floats specifying texture indices.

If vertex buffer data is in array form, data will be packed like:

```
vvvvvv nnnnnn ttttt
```

Where 'v' , 'n', and 't' have the same meanings as above.

The default is 0, do not use array format.

See the documentation for [diguyGraphicsVertexFormat](#) for more information.

5.117.3 variable `bdiGraphicsInitGraphicsAPI::default_vertex_type`

```
diguyGraphicsIndexType default_vertex_type;
```

Description:

This specifies the data type that indices in index buffer data should use.

The default is `DIGUY_GRAPHICS_INDEX_TYPE_USHORT`.

5.117.4 variable `bdiGraphicsInitGraphicsAPI::use_shaders`

```
int use_shaders;
```

Description:

Determines whether shaders should be loaded and compiled.

Note the DI-Guy skinned appearances depend on shaders being available and enabled.

Possible values are:

0	don't use shaders
1	do use shaders

Default is 1.

5.117.5 variable `bdiGraphicsInitGraphicsAPI::shader_language`

```
diguyGraphicsShaderLanguage shader_language;
```

Description:

Sets the shader language for loading shaders.

```
DIGUY_GRAPHICS_SHADER_LANGUAGE_GLSL
DIGUY_GRAPHICS_SHADER_LANGUAGE_HLSL
```

Default is `DIGUY_GRAPHICS_SHADER_LANGUAGE_GLSL`

5.117.6 variable `bdiGraphicsInitGraphicsAPI::transpose_shader_matrices`

```
int transpose_shader_matrices;
```

Description:

Whether matrices that will be used as transforms in shaders should be transposed from the DI-Guy default order.

This affects the data passed to/retrieved from the following DI-Guy Graphics API functions:

```
◇ diguyGraphicsShaderProgram::bind_link_matrices()
◇ diguyGraphicsShape::get_shader_matrix_data()
```

Set to 1 (the default) to transpose, 0 to not.

Note: Change in 12.5

Open Scene Graph uses the transposed form. OpenGL the transpose is done by the video card.

5.117.7 variable**bdiGraphicsInitGraphicsAPI::use_4x3_shader_matrices**

```
bool use_4x3_shader_matrices;
```

Description:

Whether matrices that will be used as transforms in shaders should be sent as 4x3 matrix instead of 4x4.

This affects the data passed to/retrieved from the following DI-Guy Graphics API functions:

- ◇ `diguyGraphicsShaderProgram::bind_link_matrices()`
- ◇ `diguyGraphicsShape::get_shader_matrix_data()`

Set to 1 (the default) to send the more efficient structures.

- ◇ OpenGL uses 4x3 matrices.
- ◇ Open Scene Graph and DirectX use 4x4.

5.117.8 variable**bdiGraphicsInitGraphicsAPI::use_cached_optimized_geometry**

```
int use_cached_optimized_geometry;
```

Description:

Sets whether .flt and .dae files will be loaded as geometry files optimized for DI-Guy use.

Generated files will be put into a subdirectory of the DI-Guy geometry directory, and will be automatically updated as needed (doesn't exist, or original file is newer).

The .bdg geometry format is a smaller, more efficient version of the geometry data encoded in .flt and .dae files. It is an internal DI-Guy format that is not directly supported by geometry editing programs.

There is no quality trade-off in using optimized geometry files, so it is recommended that they always be enabled.

Default is 1.

5.117.9 variable**bdiGraphicsInitGraphicsAPI::use_cached_compressed_textures**

```
int use_cached_compressed_textures;
```

Description:

Sets whether compressed texture files should be generated from source uncompressed texture files in order to reduce texture memory usage.

There is a quality trade-off when using compressed textures. If there are too many undesired artifacts in the generated textures, they should not be used.

Generated files will be put into a subdirectory of the DI-Guy geometry directory, and will be automatically updated as needed (doesn't exist, or original file is newer).

The compressed texture files will be in the .dds (DirectDraw Surface) format.

Default is 0.

5.117.10 variable**bdiGraphicsInitGraphicsAPI::use_sRGB_corrected_textures**

```
int use_sRGB_corrected_textures;
```

Description:

Sets whether sRGB textures should be used, these allow a relatively free lookup into texture maps that have been transformed into sRGB space allowing a linear lighting pipeline to be setup. OSG 3.0 does not support sRGB textures properly.

Default is 1.

5.117.11 variable**bdiGraphicsInitGraphicsAPI::use_linear_traversal_draw**

```
int use_linear_traversal_draw;
```

Description:

Sets whether DI-Guy draws characters by traversing links in a recursive tree traversal, or in a flat linear traversal. Linear drawing allows for easier culling and some additional performance on simpler characters. Linear drawing is required for bump map support.

Versions of DI-Guy prior to version 11 always used a recursive traversal.

Default is 1.

5.117.12 variable**bdiGraphicsInitGraphicsAPI::characters_have_world_space_trian**

```
int characters_have_world_space_triangle_query;
```

Description:

Experimental bits that allow the octtree to be built from internally loaded scene objects and vehicles to calculate their bounding box.

Default is 0 (off); set to 1 to turn on.

Note: this functionality requires that [diguyGraphicsMesh::free_loader_memory\(\)](#) is not called during the build process.

5.117.13 variable

bdiGraphicsInitGraphicsAPI::scene_objects_have_world_space_t

```
int scene_objects_have_world_space_triangle_query;
```

5.118 DI-Guy Graphics Init Structure, OpenGL

graphics_init_ogl.h

DI-Guy SDK Version 12.5.2

Includes: windows.h | gl.h | gl.h

Contents:

5.118.1 struct bdiGraphicsInitOgl

```
typedef struct bdiGraphicsInitOgl
{
    int use_textures;
    int use_materials;
    int use_name_stack;
    GLenum texture_mag_filter;
    GLenum texture_min_filter;
    int call_mgInit_and_mgExit;
    int use_VBOs;
    int use_FBOs;
    int use_shaders;
    int use_multitexture_extension;
    int texture_unit;
    float polygon_offset_factor;
    float polygon_offset_units;
    int use_default_fire_weapon_intersection_function;
    int use_cached_optimized_geometry;
    int use_cached_compressed_textures;
    int use_sRGB_corrected_textures;
    int use_linear_traversal_draw;
    int use_4x3_shader_matrices;
} bdiGraphicsInitOgl;
```

5.118.2 variable bdiGraphicsInitOgl::use_textures

```
int use_textures;
```

Description:

Determines whether textures will be applied. Default is 1.

5.118.3 variable bdiGraphicsInitOgl::use_materials

```
int use_materials;
```

Description:

Determines whether materials will be applied. Default is 1.

5.118.4 variable bdiGraphicsInitOgl::use_name_stack

```
int use_name_stack;
```

Description:

Determines whether character, link, and shape indices will be pushed onto the OpenGL name stack when drawing.

Possible values are:

1	pairs will be pushed onto the OpenGL name stack when drawing
0	the OpenGL name stack will be unmodified

Default is 0.

5.118.5 variable bdiGraphicsInitOgl::texture_mag_filter

```
GLenum texture_mag_filter;
```

Description:

Determines the OpenGL magnification filtering method for textures. Should be one of:

GL_NEAREST
GL_LINEAR

Default is GL_LINEAR

5.118.6 variable bdiGraphicsInitOgl::texture_min_filter

```
GLenum texture_min_filter;
```

Description:

Determines the OpenGL minification filtering method for textures. Should be one of:

```
GL_NEAREST
GL_LINEAR
GL_NEAREST_MIPMAP_NEAREST
GL_NEAREST_MIPMAP_LINEAR
GL_LINEAR_MIPMAP_NEAREST
GL_LINEAR_MIPMAP_LINEAR
```

Default is GL_LINEAR_MIPMAP_LINEAR

5.118.7 variable bdiGraphicsInitOgl::call_mgInit_and_mgExit

```
int call_mgInit_and_mgExit;
```

Description:

Deprecated; *Boston Dynamics* no longer uses the MG-API library, so this setting is not used. It is left here in order to not break existing code.

5.118.8 variable `bdiGraphicsInitOgl::use_VBOs`

```
int use_VBOs;
```

Description:

Determines whether vertex buffer object extensions are used. In some cases using vertex buffer objects rather than display lists can improve performance.

Possible values are:

0	don't use vertex buffer object extensions
1	do use vertex buffer object extensions

Default is 1.

5.118.9 variable `bdiGraphicsInitOgl::use_FBOs`

```
int use_FBOs;
```

Description:

Determines whether frame buffer object extensions are used. In some cases using frame buffer objects can improve performance of intersection testing.

This currently only has an effect on Win32 platforms.

There is a known problem with some graphics cards that using frame buffer objects leads to a program crash.

Possible values are:

0	don't use frame buffer object extensions
1	do use frame buffer object extensions

Default is 0.

5.118.10 variable `bdiGraphicsInitOgl::use_shaders`

```
int use_shaders;
```

Description:

Determines whether shaders should be loaded and compiled.

Note the DI-Guy skinned appearances depend on shaders being available and enabled.

This currently only has an effect on Win32 platforms.

There is a known problem with some graphics cards using GLSL possibly causing a crash.

Possible values are:

0	don't use shaders
1	do use shaders

Default is 1.

5.118.11 variable `bdiGraphicsInitOgl::use_multitexture_extension`

```
int use_multitexture_extension;
```

Description:

Determines whether the multitexture extension is used.

Possible values are:

0	don't use multitexture extensions
1	do use multitexture extensions

Default is 0.

5.118.12 variable bdiGraphicsInitOgl::texture_unit

```
int texture_unit;
```

Description:

Sets which texture unit is used for applying textures. Should be set to an integer, **not** to a GL enumeration.

Possible values are:

0	use GL_TEXTURE0
1	use GL_TEXTURE1
2	use GL_TEXTURE2
3	use GL_TEXTURE3

Default is 0, or texture unit GL_TEXTURE0.

5.118.13 variable bdiGraphicsInitOgl::polygon_offset_factor

```
float polygon_offset_factor;
```

Description:

Sets the value passed to the factor argument of the glPolygonOffset() function. This function is used when rendering polygon subfaces, which are commonly used for edge highlighting and decals on signs.

Default is -1.0.

5.118.14 variable `bdiGraphicsInitOgl::polygon_offset_units`

```
float polygon_offset_units;
```

Description:

Sets the value passed to the units argument of the `glPolygonOffset()` function.

Default is `-1.0`.

5.118.15 variable `bdiGraphicsInitOgl::use_default_fire_weapon_intersection_function`

```
int use_default_fire_weapon_intersection_function;
```

Description:

Sets if a default shooting intersection function should be automatically set up.

See

[`diguyScenario::set\_default\_fire\_weapon\_intersection\_function\(\)`](#) for more details.

Default is `1`.

5.118.16 variable**bdiGraphicsInitOgl::use_cached_optimized_geometry**

```
int use_cached_optimized_geometry;
```

Description:

Sets whether .flt and .dae files will be loaded as geometry files optimized for DI-Guy use.

Generated files will be put into a subdirectory of the DI-Guy geometry directory, and will be automatically updated as needed (doesn't exist, or original file is newer).

The .bdg geometry format is a smaller, more efficient version of the geometry data encoded in .flt and .dae files. It is an internal DI-Guy format that is not directly supported by geometry editing programs.

There is no quality trade-off in using optimized geometry files, so it is recommended that they always be enabled.

Default is 1.

5.118.17 variable**bdiGraphicsInitOgl::use_cached_compressed_textures**

```
int use_cached_compressed_textures;
```

Description:

Sets whether compressed texture files should be generated from source uncompressed texture files in order to reduce texture memory usage.

There is a quality trade-off when using compressed textures. If there are too many undesired artifacts in the generated textures, they should not be used.

Generated files will be put into a subdirectory of the DI-Guy geometry directory, and will be automatically updated as needed (doesn't exist, or original file is newer).

The compressed texture files will be in the .dds (DirectDraw Surface) format.

Default is 1.

5.118.18 variable**bdiGraphicsInitOgl::use_sRGB_corrected_textures**

```
int use_sRGB_corrected_textures;
```

Description:

Sets whether SRGB textures should be used, these allow a relatively free lookup into texture maps that have been transformed into sRGB space allowing a linear lighting pipeline to be setup.

Default is 1.

5.118.19 variable bdiGraphicsInitOgl::use_linear_traversal_draw

```
int use_linear_traversal_draw;
```

Description:

Sets whether DI-Guy draws characters by traversing links in a recursive tree traversal, or in a flat linear traversal. Linear drawing allows for easier culling and some additional performance on simpler characters. Linear traversal is required for some advanced lighting shaders.

Versions of DI-Guy prior to version 11 always used a recursive traversal.

Default is 1.

5.118.20 variable bdiGraphicsInitOgl::use_4x3_shader_matrices

```
int use_4x3_shader_matrices;
```

Description:

Whether matrices that will be used as transforms in shaders should be sent as a 4x3 matrix instead of 4x4. This should only be changed if your video card is unable to use 4x4 matrices.

Default is 1.

5.119 Boston Dynamics Log Library

libbdilog.h

DI-Guy SDK Version 12.5.2

Includes: declspec_bdiutil.h

Contents:

- ◆ [Constants and Types](#)
- ◆ [Standard stderr/stdout Callbacks](#)
- ◆ [File Callbacks](#)
- ◆ [Print Functions](#)
- ◆ [Registration Functions](#)

The *Boston Dynamics* log library is used by most of *Boston Dynamics'* software. It allows programs to write errors, debugging information, etc., to zero, one, or more outputs including stderr, a file, or a user-supplied output.

5.119.1 constant CHECK_PRINTF_FORMAT

```
#define CHECK_PRINTF_FORMAT __attribute__((format (printf, 2, 3)))
```

Constants and Types

5.119.2 typedef bdiCallbackHandle

```
typedef int bdiCallbackHandle;
```

5.119.3 constants

```
#define BDI_LOG_MAX_CALLBACKS      (8)
#define BDI_LOG_BAD_CALLBACK_HANDLE (-1)
```

5.119.4 constants

```
#define BDI_LOG_NOTHING      (-1)
#define BDI_LOG_ALWAYS      (0)
#define BDI_LOG_FATAL       (1)
#define BDI_LOG_ERROR       (2)
#define BDI_LOG_WARN        (3)
#define BDI_LOG_INFO        (4)
```

DI-Guy SDK Version 12.5.2 Index

```
#define BDI_LOG_DEBUG      (5)
#define BDI_LOG_DEBUG_LV1 (6)
#define BDI_LOG_DEBUG_LV2 (7)
#define BDI_LOG_DEBUG_LV3 (8)
#define BDI_LOG_DEBUG_LV4 (9)
#define BDI_LOG_DEBUG_LV5 (10)
#define BDI_LOG_DEBUG_LV6 (11)
#define BDI_LOG_DEBUG_LV7 (12)
#define BDI_LOG_DEBUG_LV8 (13)
#define BDI_LOG_DEBUG_LV9 (14)
#define BDI_LOG_EVERYTHING (15)
```

Log notify levels. A lower number indicates higher importance.

Standard stderr/stdout Callbacks

5.119.5 function `bdi_log_stderr_enable`

Prototype:

```
int bdi_log_stderr_enable(int notify_level);
```

Description:

This function will route a copy of appropriate log messages to stdio's stderr.

Arguments and Return Value:

return value	0 on success, -1 on failure
notify_level	only messages whose notify level is at least as important as this will appear in this log

5.119.6 function `bdi_log_stderr_disable`

Prototype:

```
int bdi_log_stderr_disable(void);
```

Description:

This function shuts down the log opened by `bdi_log_stderr_enable()`.

5.119.7 function `bdi_log_stdout_enable`

Prototype:

```
int bdi_log_stdout_enable(int notify_level);
```

Description:

This function will route a copy of appropriate log messages to stdio's stdout.

Arguments and Return Value:

return value	0 on success, -1 on failure
notify_level	only messages whose notify level is at least as important as this will appear in this log

5.119.8 function `bdi_log_stdout_disable`

Prototype:

```
int bdi_log_stdout_disable(void);
```

Description:

This function shuts down the log opened by [bdi_log_stdout_enable\(\)](#).

Arguments and Return Value:

return value	0 on success, -1 on failure

File Callbacks

5.119.9 function bdi_log_file_enable

Prototype:

```
int bdi_log_file_enable(int notify_level, const char* filename, int clear_file);
```

Description:

This function will route a copy of appropriate log messages to the named file.

Arguments and Return Value:

return value	0 on success, -1 on failure
notify_level	only messages whose notify level is at least as important as this will appear in this log
filename	name of the file in which messages will appear
clear_file	0 or 1; 1 means clear out prior contents of file, 0 means append to the file

5.119.10 function `bdi_log_file_disable`

Prototype:

```
int bdi_log_file_disable(void);
```

Description:

This function shuts down the log opened by `bdi_log_file_enable()`.

Arguments and Return Value:

return	0 on success, -1 on failure
value	

Print Functions

5.119.11 function bdi_log_print**Prototype:**

```
int bdi_log_print(int notify_level, const char* string);
```

Description:

This function sends the specified string to all open logs. Whether or not the string is actually printed depends the specified `notify_level` in this call and the `notify_level` with which each log was opened.

Arguments and Return Value:

return value	0 on success, -1 on failure
string	message to be (potentially) printed
notify_level	"importance" of message; lower is more important

5.119.12 function bdi_log_set_global_notify_level**Prototype:**

```
int bdi_log_set_global_notify_level(int notify_level);
```

Description:

This function sets the global notify level. Messages whose `notify_level` argument is less important than this will not be printed, regardless of the `notify_level` used to open logs.

The starting global notify level is `BDI_LOG_EVERYTHING`.

Arguments and Return Value:

return value	0 on success, -1 on failure
notify_level	global level of importance

Example:

```
// open a file log with notify level BDI_LOG_WARN
bdi_log_file_enable(BDI_LOG_WARN, "errors.txt", 1);

// this message WILL appear in the log
bdi_log_print(BDI_LOG_WARN, "Message 1.\n");

// set the global notify level to be more strict
bdi_log_set_global_notify_level(BDI_LOG_ERROR);

// this message WILL NOT appear in the log
// (log messages of importance less than BDI_LOG_ERROR are ignored.)
bdi_log_print(BDI_LOG_WARN, "Message 2.\n");
```

5.119.13 function `bdi_log_get_global_notify_level`

Prototype:

```
int bdi_log_get_global_notify_level(void);
```

Description:

Returns the most recent setting of
`bdi_log_set_global_notify_level()`.

Arguments and Return Value:

return	most recent setting
value	

Registration Functions

5.119.14 function `bdi_log_register_callback`

Prototype:

```
bdiCallbackHandle bdi_log_register_callback(int notify_level,
      bdiLogCallbackFunction* callback,
      void* user_data);
```

Description:

This function registers a callback for the specified `notify_level`.

Arguments and Return Value:

return value	a handle
callback	pointer to the callback function
notify_level	level of "importance" the callback will be called for
user_data	pointer that will be supplied to the callback each time it is called.

5.119.15 function `bdi_log_unregister_callback`

Prototype:

```
int bdi_log_unregister_callback(bdiCallbackHandle handle);
```

Description:

This function unregisters a previously registered callback.

Arguments and Return Value:

return value	0 on success, -1 on failure
handle	the handle returned by <code>bdi_log_register_callback()</code>

5.119.16 function `bdi_log_get_num_logs_open`

Prototype:

```
int bdi_log_get_num_logs_open();
```

Description:

This function returns the number of logs currently open.

Arguments and Return Value:

return value	number of logs open

5.119.17 function bdi_log_set_notify_level**Prototype:**

```
int bdi_log_set_notify_level(bdiCallbackHandle handle, int notify_level);
```

Description:

This function changes the notify level of a registered callback.

Arguments and Return Value:

return value	0 on success, -1 on failure
handle	the handle returned by bdi_log_register_callback()
notify_level	the new notify level

5.119.18 function bdi_log_get_notify_level**Prototype:**

```
int bdi_log_get_notify_level(bdiCallbackHandle handle);
```

Description:

This function returns the notify level of a registered callback.

Arguments and Return Value:

return value	most recent setting or BDI_LOG_NOTHING if handle is invalid
handle	the handle returned by bdi_log_register_callback()

5.119.19 function `bdi_log_get_string_from_notify_level`**Prototype:**

```
void bdi_log_get_string_from_notify_level(int notify_level,
    char* result,
    int result_buf_size);
```

Description:

Derives a string from the notify level and returns it in the passed result buffer.

Arguments:

<code>notify_level</code>	notify level to parse
<code>result</code>	character buffer for result
<code>result_max_len</code>	result buffer size

5.119.20 function `bdi_log_register_fatal_exit_function`**Prototype:**

```
void bdi_log_register_fatal_exit_function(bdiLogFatalExitFunction* fatal_exit_func),
```

Description:

This function registers a function that will be called if a log message is sent with level `BDI_LOG_FATAL`. The function should do whatever it takes to shut down gracefully.

If no fatal exit function is registered, `exit(EXIT_FAILURE)` will be called by default.

5.119.21 function bdi_log_print_to_output_window**Prototype:**

```
int bdi_log_print_to_output_window(char* format, ...);
```

Description:

This function prints using OutputDebugString on Windows, which will show up in the Output window in Microsoft Developer Studio.

5.119.22 function bdi_log_lock**Prototype:**

```
void bdi_log_lock();
```

Description:

This function grabs the lock of the logging system. This is pretty hacky. If you do not release this lock, you're in trouble. This can be useful for certain specialized interactions with log callback objects that rely on the libbdilog lock for their synchronization.

5.119.23 function `bdi_log_unlock`

Prototype:

```
void bdi_log_unlock();
```

Description:

This function releases the lock of the logging system. This is pretty hacky. If you do not release this lock, you're in trouble. This can be useful for certain specialized interactions with log callback objects that rely on the libbdilog lock for their synchronization.