DI-Guy 13.2 Release Notes

This document provides the following release-specific information:

Document ID: DIG-13.2-8-161214

Systems Supported

Table 1 lists the platforms currently supported by MÄK DI-Guy 13.2. Application code must be built with the indicated compilers in order to link to DI-Guy libraries.

Table 1: Platforms supported

Platform	Compiler
Red Hat Enterprise Linux Workstation 6.0	Default compiler (64-bit)
PC with Vista/Windows 7/Windows 8/Windows 10	Microsoft Visual C++ 10.0 (64 bit) Microsoft Visual C++ 12.0 (64 bit)

You must be an administrator to install MÄK products on Windows Vista.

Renderer Support

DI-Guy 13.2 supports the following renderers:

- Windows 64 bit: OpenGL, OpenGL Graphics API, OSG Graphics API.
- Reference implementations: DX9, DX11, Unity.
- Linux: OpenGL, OpenGL Graphics API, OSG.

License Manager

To run the licensed version of DI-Guy, you must install license management software. DI-Guy 13.2 uses FLEXlm 11.13.0.2. If you are upgrading from a version of the DI-Guy that used an older version of FLEXlm, you must upgrade your license management files. You do not need a new license. Licenses are forward compatible.

The License Manager files are not part of the DI-Guy installer. You can download them at:

- Windows: ftp://ftp.mak.com/out/MAKLicenseManager-win64-setup.exe
- Linux: ftp://ftp.mak.com/out/MAKLicenseManager-linux64-setup.tar.gz

A license manager support is available at:

http://www.mak.com/support/license-support.html

Network Compatibility

HLA only

DI-Guy 13.2 was built against VR-Link 5.3 and is compliant with:

- RPR-FOM 1.0 and 2.0.
- MÄK RTI 4.1 or later.

DIS only

DI-Guy 13.2 supports DIS 4, 5, 6, and 7.

RTI Support

We recommend using the latest version of the MÄK RTI. However, DI-Guy should work with other RTIs that conform to the HLA 1.3 specification, the IEEE 1516 SISO DLC API or HLA Evolved and are built with the same compiler as DI-Guy.

When using the MÄK RTI, remember to make sure that DI-Guy can find your FED file, and optional *rid.mtl,* file by either putting them in the directory from which you are running, or by setting the environment variable to the directory that contains them.

FOM Support

DI-Guy 13.2 has built-in support for versions 0.5, 0.7, 0.8, 1.0, and 2.0. It also supports VR-Link's ability to support alternative FOMs through the FOM Mapper. By default, DI-Guy 13.2 uses RPR FOM 1.0. It is recommended that you use the DI-Guy FOM Extensions. DI-Guy includes two custom FOMs, one for HLA 1.3 and HLA 1516, and one for HLA Evolved.

New Features and Updates

DI-Guy 13.2 has the following updated features:

- The OpenGL renderer has been rewritten to support OpenGL 3.2 and remove deprecated OpenGL functionality. This does not change the DI-Guy API, but you may need to modify legacy code due to changes to OpenGL.
- Graphics in rgba format have been replaced with pngs and tiffs.
- Many new heads. For examples, please see Figure 1.

API Changes

Critical API Changes

- diguyGraphicsMesh::get_current_graphics_state() has been removed.
- The function prototype for diguyGraphicsMesh::draw() has changed to diguyGraphicsMesh::draw(diguyDrawCallData * data). This allows both our instanced and non-instanced code to share a mesh drawing path, and allows us to make further refinements without breaking API compatibility.

Geometry Pipeline

- Fixed mesh count calculation for flight file external references.
- Delayed mesh count calculation since it's rarely needed by immediate mode APIs.
- Improved calculation of bump and transparency bits so they are available to the graphics API.
- Fixed support for alpha to coverage clip maps.
- Fixed issues with draw pass debugger in Character Viewer.
- Fixed a hard crash when optimizing mesh with null textures.
- Fixed material optimization bug that lead to shinyness values of 0 on many FLT files.
- New global ambient material override system, unifies all ambient responses regardless of source per model ambient customization really should be at the per pixel level via ao maps. This gives you more high level control of the look of your DI-Guy content.

Gesture System

- New option for gestures to be done additively based on 1st pose of gesture frame. Useful for recoil animations.
- Gestures with no weighted vars now play animation on all bones.
- diguyCharacter::execute_gesture() now supports playing arbitrary bdm files off disk. This can be useful for end users who aren't concerned with setting up action/transition tables.

Performance

- Streamlined memory usage of links. This lowers the size of every simulated joint DI-Guy manages. Removed underlying static offset matrix, unified data into dynamic matrix and simplified matrix math operations.
- Streamlined memory usage of pose override system. This affects gazing, pointing, aiming.

Joint Control API

Adding diguyLinkController, a new API for doing link control at the link level. Much simpler then pose override class and able to be scripted with features to let it automatically animate.

Particles

- Fixed default bounding radius initialization in particle system constructor, could cause effect characters to not update properly due to being mistakenly culled out. (Seen in OSG).
- Improvements to multi-threading of particle systems.
- Added support for transient fire effects with link offsets.
- Improvements to support transient particle systems driving external particle systems, and properly timing out.
- Transient particles that are flagged as mergeable will now attempt to move existing emitter and restart the system. Most weapon effects are now flagged as mergeable, allowing many characters to create muzzle smoke, and shell eject casings efficiently.
- Fixes to collision with world/animated spinning.

Munition System

- Removed:
  - diguyCharacter::set_weapon_shell_eject_enabled()
  - get_weapon_shell_eject_enabled()
  - set_weapon_smoke_enabled()
  - get_weapon_smoke_enabled().

  These now all key off of diguyScenario::get_weapon_fire_effects_enabled().
- Munitions type effects can now inherit other munitions, makes it easier to author one set of effects and share it with all the various tanks or other weapons with just a different septet.
- Tanks now have a smoke explosion when firing and a recoil for their barrel.
- People have an additive kickback gesture that works regardless of the characters aiming direction. Puff of smoke at muzzle point, and shell eject on the right side of most weapons. Tracer fire has been improved as well.

Miscellaneous

- Adding particle movement animation module, object will move at the exact speed requested via animation_velocity API.
- Improvements to LOD calculation for large objects. Fixed bug where motion LOD would make a ground clamping car spin through the ground.
- Improved gaze at angle to work better in arbitrary coordinate systems, regardless of movement of character.
- Adding multi-threaded read/write lock mutex to protect octtree from editing while testing, fixes crashes with moving props.
- Improved movement visuals to not appear inside of vehicles.
- Added visualization of angular velocity.
- Improved calculation of angular velocity from vehicles.
- Fixed ground bounds calculation to work with turreted vehicles again.
- Changing the character appearance should attempt to retain more of the animation system state.
- Stationary motions will no longer get scaled by character height.
- Switched sound module to openAL-soft.
- Added support for scale to physics simulation.
- Added caching for local space bounding box retrieval.
- Added warning for bad rigging on vehicles.

New API Classes:

- diguyLinkController - a simple class that allows end users to override a character's animation on a joint level without the complexity of the diguyPoseOverride class.
- diguyUniformBufferUpdater - a virtual interface class that wraps all uniform buffer management.
- diguyGraphicsFrameBuffer - a virtual interface class that allows DI-Guy to create frame buffers for shadows and intersection testing.

New API functions:

- diguyCharacter::get_supports_texture_variations()
- diguyCharacter::get_supports_weight_variations()
- diguyCharacter::set_weight_scale()
- diguyCharacter::get_weight_scale()
- diguyCharacter::set_allow_instancing()
- diguyCharacter::get_allow_instancing()
- diguyCharacter::get_is_instanced()
- diguyCharacter::set_final_tbo_position_matrix()
- diguyCharacter::get_shader_technique();

- diguyCharacter::set_aim_variable_interpolation_time()
- diguyCharacter::get_aim_variable_interpolation_time()
- diguyCharacter::get_link_controller()
- diguyCharacter::set_link_translation_override()
- diguyCharacter::set_link_translation_override()
- diguyCharacter::end_link_translation_override()
- diguyCharacter::set_link_rotation_override()
- diguyCharacter::set_link_rotation_override()
- diguyCharacter::end_link_rotation_override()
- diguyCharacter::get_num_articulated_parts()
- diguyCharacter::get_articulated_part_link_name()
- diguyCharacter::get_articulated_part_id()
- diguyCharacter::map_articulated_part_id_to_link()
- diguyCharacterPoseOverride::set_weight_array_modifier()
- diguyCharacterPoseOverride::get_weight_array_modifier()
- diguyGraphicsInstanceGroup::get_show_debug_colors()
- diguyGraphicsLink::set_shape_switch_override()
- diguyGraphicsLink::get_initial_translation()
- diguyScenario::set_visualize_instance_groups()
- diguyScenario::get_visualize_instance_groups()
- diguyScenario::set_instancing_tbo_patching_enabled()
- diguyScenario::get_instancing_tbo_patching_enabled()
- diguyScenario::set_instancing_position_callback_enabled()
- diguyScenario::get_instancing_position_callback_enabled()
- diguyScenario::set_num_extra_per_instance_data_floats()
- diguyScenario::get_num_extra_per_instance_data_floats()
- diguyScenario::build_instance_groups()
- diguyScenario::update_instancing_data()
- diguyScenario::draw_particles()
- diguyScenario::get_num_appearances_of_appearance_type()
- diguyScenario::get_appearance_name_at_index()
- diguyScenario::set_use_override_ambient_material()
- diguyScenario::get_use_override_ambient_material()
- diguyScenario::set_global_ambient_material()
- diguyScenario::get_global_ambient_material()
- diguyGraphicsMesh::get_is_blend_shape()
- diguyGraphicsMesh::get_blend_shape()
- diguyGraphicsMesh::get_num_blend_shapes()

- diguyViewPainter::draw_opengl_primitive()

DI-Guy Scenario

- Added scene object instancing system that speeds up rendering of external references.
- Large scenes can render up to two times faster.
- Action bead can now calculate velocity for locomotion actions when deriving duration.
- Added scene rendering debugging tools.
- Added UI for controlling global ambient material.
- Added geometry viewing tools for characters and scene objects.
- Added ability to change weight of characters from thin to overweight.

Networking

- Upgraded to VR-Link 5.3.
- Adding support for multi-threaded publishing.
- Fixed bug where reflected characters had variation textures on by default.
- New Networking debugging panel.
- Incoming and outgoing packet info visualization aids.
- Packet indicators pop little spheres over the heads of characters when they either send or receive a new movement packet. Outgoing limited to DIS currently.
- Support for enabling log messages from VR-Link.
- New network adapter chooser that should help set the transmit to IP UI.
- Fixed bug with sending rotational velocity, was reversed compared with DIS specification.
- Fixed bug with dead reckoning individual that was moving backwards, led to very incorrect behavior.
- Better distribute PDU heartbeat events in time.
- Fixed bug preventing send_xc_ commands from rewinding/resetting remote scenario.
- Added acceleration publishing.
- Vehicles without custom PDUs will have rotating wheels.
- DIGUY_CUSTOM_PDU_ID_CHARACTER_SCALE has been modified to include an additional 4th value for weight scale.

Camera System Changes

- Camera zipping is once again triggered by shift-click.
- A tracked character can only be watched from a stationary position, unless the user puts the camera into orbit mode. If not, the tracking will stop as soon as the camera is moved.
- Double-clicking on a character will put the camera into orbit mode around it.

DI-Guy Motion Editor

- New Outliner UI, nicer way to see and edit all the joints in a skeleton. The Joint editor UI buttons now trigger the new Outliner UI.
- Fixed crashes on processing degenerate animation files.
- dae animations with static parts are now properly imported into Motion Editor.
- Fixed bugs creating Collada animation files from DI-Guy motions.
- Unified advanced and regular export motion dialog box. Made it possible to export blended motion without idealizing.
- Adding support for a scripting interface to motion editor export process via diguyMotionExporter class.
- Improved Gesture UI to work with more gesture types.
- Gesture weights UI now just allow editing of a unified rotation weight, more accurately reflect the behavior of the underlying code.

Graphics API and Rendering Pipeline Changes

- The core DI-Guy rendering pipeline has been upgraded to use OpenGL 3.2. We have removed deprecated fixed function code from our 2 OpenGL Renderers, all of our programming examples, DI-Guy Scenario, Character Viewer, and Motion Editor. You should be able to debug all of our code with Nvidia's NSight as well.
  - Legacy OpenGL 2.1 shaders are now located in *./geometry/shaders/opengl_21*. These are still actively used by OSG reference implementation. Modern Shaders are located in the opengl_32 subdirectory.
  - diguyGraphicsShaderProgram::derive_diguy_shader_source_filename() now takes a subdirectory argument to allow us to have the same shader description with different implementations.
  - All OpenGL 3.2 shaders have been rewritten to use a shared library (*diguy_shared_library.glsl*). This centralizes the use and implementation of uniform buffers, standard uniforms, and attributes. Shared functions that are used by many shaders have also moved there.
  - The built-in fixed function uniforms have been removed from all modern shaders.
  - The various LOD shaders and instancing shaders have been cleaned up to remove slight lighting discrepancies between the shader LODs.
  - Bump mapping has been streamlined to remove the need for a number of per object uniforms. This has simplified bump map uniform set up in diguyGraphicsLink::begin_character_draw() (that is, it's gone). It has also simplified the whole lighting pipeline.
  - Bump mapped shaders have changed quite a bit. They do more work in the pixel shader, but that was the cleanest way to support multiple bump mapped lights as well as improve uniformity between shader stages.
  - DI-Guy shaders now incorporate Fresnel lighting and basic support for physically-based shading (PBR). A number of cars have been upgraded. We hope to update more of our content to use PBR in the future.
- Long term, we intend to remove the libdiguy_graphics_ogl implementation and move all of our Applications (DI-Guy Scenario) and sample code to the diguy_graphics_api, the diguyOglParticleSystemRenderer and the diguyOglUniformBufferUpdater for example are now used by both the diguy_ogl examples and the diguyGraphicsAPI OpenGL examples, and DI-Guy Scenario. The justification for this is many faceted:
  - Eliminate the need to maintain two OpenGL code bases, one closed source and one public source.
  - Guarantee that any thing that can be done with the closed source code is doable with the public API and implementation.
  - Recognize the reality that with the removal of the fixed function pipeline and fixed function OpenGL state the closed source libdiguy_graphics_ogl renderer is not going useful for many people.

- glew and glm are now required by all of our OpenGL programming examples.
- The function prototype for diguyGraphicsMesh::draw() has changed to diguyGraphicsMesh::draw(diguyDrawCallData * data). This allows both our instance and non-instanced code to share a mesh drawing path, and allows us to make further refinements without breaking API compatibility. Your diguyGraphicsMesh implementation is now responsible for binding geometry material and texture state and managing early outing for transparency, diguyOglGraphicsMesh now demonstrates all of that.
- The instancing system now supports our variation system. The sample code for the variation now combines all variation textures into one large 4096 texture. This avoids creating hundreds of tiny textures. This also allows the instancing system to use it. A uniform for specifying the row of the shared variation texture has been added. And it is now encoded as part of the TBO data that is sent to the video card.
- The instancing system now has a shadow pass shader.
- Unlit shader now included variation texture support. (Avoids popping.)
- Instancing now defaults to LOD 4, kicks on much earlier but new variation support makes it less noticeable.
- Fixed bug where num rendering passes was defaulting to 1 not 2. Caused transparency to look bad. Change requires that diguyScenario::draw_pass1() and then diguyScenario::draw_pass2() are called with different blend settings.

DirectX 11 Sample

We've added a very bare bones sample of using DI-Guy with DirectX 11. This is a very rough proof of concept, but all the major pieces to get a character on the screen are there.

OSG Improvements

- OSG now demonstrates using the DI-Guy ogl immediate mode particle renderer as an add-on drawable.
- Fixed scale bug where character scaling was applied 2x.
- Streamlined mesh loading, added support for blend shapes applied to meshes. Currently limited by OSG's infrastructure.

Cleanups

- Removed DIGUY_GRAPHICS_INDEX_TYPE_D3D_WORD, and some unused vertex array formats.
- Removed bdiGraphicsInitGraphicsAPI.use_array_format_for_vertex_buffer_data.
- Removed bdiGraphicsInitGraphicsAPI.use_linear_traversal_draw.
- Removed bdiGraphicsInitGraphicsAPI.use_shaders.

- Removed bdiGraphicsInitGraphicsAPI.default_vertex_type. This really shouldn't have ever been a preference index data is either a ushort or uint depending on the size of the mesh.
- bdiGraphicsInitGraphicsAPI.characters_have_world_space_triangle_query now defaults to 1 in gfx API. This allows bounding box calculations to work. This really is only there for historical reasons when we didn't always control our geometry pipeline.
- Cleaned up bdiGraphicsInitOgl, removing options that no longer made sense:
- Removed bdiGraphicsInitOgl.texture_unit.
- Removed bdiGraphicsInitOgl.use_multitexture_extension.
- Removed bdiGraphicsInitOgl.use_shaders.
- Removed bdiGraphicsInitOgl.use_VBOs.
- Removed diguyGraphicsShaderProgram::reload(). It is simpler to just require that load() properly handle being called multiple times.
- Removed diguy_ogl_render_shadow_map() replaced with diguyOglUtils::render_-shadow_map().

New Content

This section illustrates some of the new content in DI-Guy.

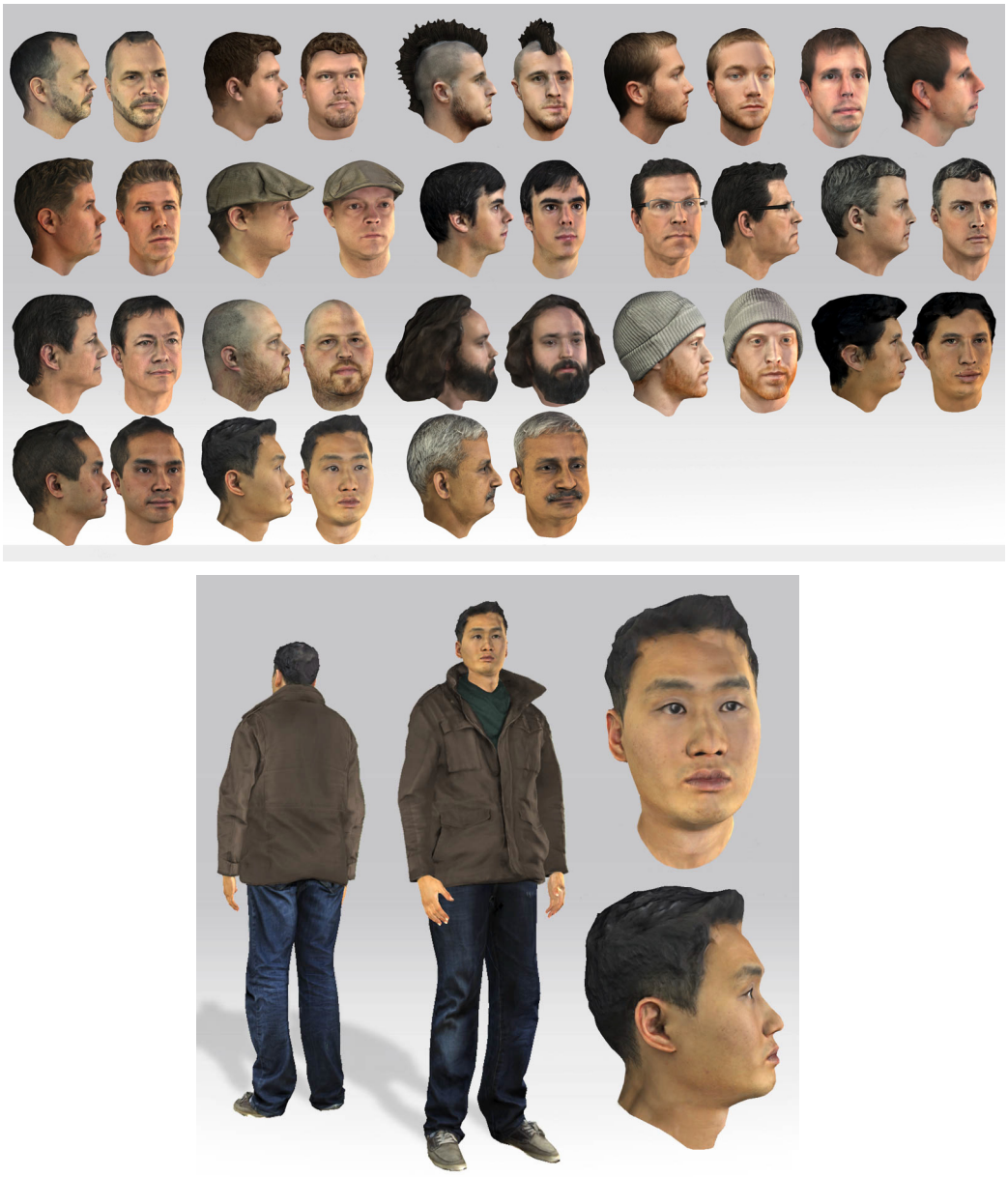

Figure 1. New heads

Figure 2. New civilians

Figure 3. Weight variation

Figure 4. Camels

Known Problems

DI-Guy 13.2 has the following known problems:

- There are Z buffering issues when running the OGL in Linux.