DI-Guy Scenario

Users Guide

VT MÄK
A company of VT Systems

DI-Guy Scenario

Users Guide

VT MÄK
A company of VT Systems

VT MÄK
150 Cambridge Park Drive, 3rd Floor
Cambridge, MA 02140 USA

Voice: 617-876-8085
Fax: 617-876-9208

info@mak.com

www.mak.com

Revision DIG-13.0-6-140523

Contents

Preface

How This Manual is Organized xi
Documentation xii
Derivative Work xiii
MÄK Products xiv
How to Contact Us xvii
Document Conventions xviii
- DI-Guy Conventions xix
- Mouse Button Naming Conventions xix

Third Party Licenses xix
- Boost License xix
- libXML and libICONV xx
- Lua xx
- pThreads Library xxi
- LizardTech xxi
- Freefont OpenType Font Set xxi
- Kynapse License xxi
- Third-Party Licenses for VR-Vantage Applications xxiii

Chapter 1. Introduction to DI-Guy Scenario

1.1. Introduction 1-2
- 1.1.1. What is DI-Guy Scenario? 1-2
- 1.1.2. DI-Guy Scenario is Real Time 1-3
- 1.1.3. DI-Guy Scenario Comes with People 1-3
- 1.1.4. DI-Guy AI 1-4
- 1.1.5. DI-Guy Scenario is Easy to Use 1-4
- 1.1.6. Supported Platforms 1-4
- 1.1.7. Optional Components for DI-Guy Scenario 1-5

1.2. Vehicles 1-6

1.3. Expressive Faces 1-7
1.4. Customizing DI-Guy Content 1-7

Chapter 2. Starting DI-Guy Scenario

2.1. Installing DI-Guy Scenario 2-3
2.2. Starting DI-Guy Scenario 2-3
2.2.1. Starting DI-Guy Scenario from a Command Prompt 2-4
2.2.2. Command-line Arguments 2-4
2.2.3. Managing Plug-ins 2-7
2.2.4. Specifying a Startup Script 2-8
2.3. Introduction to Scenarios 2-9
2.3.1. Creating a Scenario 2-10
2.3.2. Saving Scenarios 2-10
2.3.3. Opening a Scenario 2-10
2.3.4. Running Scenarios 2-11
2.3.5. Specifying a Scenario As Read-Only 2-13
2.3.6. Writing a Scenario Description 2-13
2.3.7. Merging Scenarios 2-14
2.3.8. Recording a Scenario to a Movie 2-14
2.4. The DI-Guy Scenario Log 2-16
2.5. Setting DI-Guy Scenario Preferences 2-17
2.6. Using Multiple 3D Windows 2-20
2.7. Common GUI Controls 2-20
2.7.1. Adding a New Object 2-21
2.7.2. Deleting Objects 2-22
2.7.3. Copying Objects 2-22
2.7.4. Teleporting the Camera to a Specific Object 2-22
2.7.5. Changing the Index of an Object 2-22
2.7.6. Sharing Libraries Among Scenarios 2-23
2.8. Hiding and Displaying Objects 2-28

Chapter 3. DI-Guy Scenario Tutorials

3.1. A Simple Scenario 3-3
3.1.1. Start DI-Guy Scenario 3-3
3.1.2. Move the Camera 3-5
3.1.3. Add People to the Scenario 3-6
3.1.4. Play the Scenario 3-8
3.1.5. Save the Scenario 3-9
3.2. Selecting Characters and Changing their Appearance 3-9
3.2.1. Choose a Character Type 3-10
3.2.2. Choose an Appearance 3-10
3.2.3. Choose a Hand Item 3-11
3.2.4. Choose an Action 3-11
3.2.5. Add the Character 3-11
3.2.6. Other Types of Characters 3-12
3.2.7. Change a Character's Appearance 3-12

3.2.8. Selecting a Character 3-13
3.3. Creating Paths for Characters 3-14
3.3.1. Selecting a Path 3-14
3.3.2. Adjust a Path 3-15
3.3.3. Extend a Path 3-16
3.3.4. Insert Waypoints 3-17
3.4. Actions and Behaviors 3-18
3.4.1. Action Beads 3-18
3.4.2. Adding Actions 3-19
3.4.3. Dragging Actions 3-20
3.4.4. Adding Stationary Actions 3-20
3.4.5. Change an Action 3-21
3.5. Decision Logic 3-21
3.5.1. Sensor Regions 3-21
3.5.2. Creating Decision Logic 3-22
3.6. Cameras 3-28
3.6.1. Saved Camera Settings 3-29
3.6.2. Teleport the Camera 3-30
3.6.3. Tether the Camera to a Character 3-30
3.6.4. Tracking a Character 3-31
3.6.5. Camera Point-of-View (POV) 3-32
3.7. Load Your Own Terrain 3-32
3.8. Change the Environment 3-34

Chapter 4. Creating and Editing Characters

4.1. Creating a Character 4-3
4.1.1. Creating a Character and a Path 4-5
4.1.2. Creating a Character on the Character Page 4-6
4.1.3. Using the Appearance Wizard to Choose a Character 4-6
4.2. Selecting a Character 4-8
4.3. Deleting a Character 4-9
4.4. Changing a Character's Name 4-9
4.5. Enabling Characters 4-9
4.6. Hiding Characters 4-10
4.7. Editing Characters 4-10
4.7.1. Changing a Character's Character Type 4-10
4.7.2. Changing a Character's Appearance 4-11
4.7.3. Setting the Initial Path 4-11
4.7.4. Initial Position and Orientation 4-12
4.7.5. Setting a Character's Scale 4-12
4.7.6. Setting Time In and Time Out 4-13
4.7.7. Specifying that a Character is a Scene Object 4-13
4.7.8. Apply Actor Scale to Action Bead Travel 4-13
4.7.9. Advanced Character Editing 4-13
4.8. Cutting, Copying, and Pasting Characters 4-19
4.9. Undoing Changes 4-20

4.10. Creating Character Variables 4-20
4.11. Parenting One Character to Another 4-21
4.12. Specifying Weapon Parameters 4-21
4.13. Using a Joystick or Keyboard to Control a Character 4-22
4.13.1. Using a Joystick 4-23
4.13.2. Keyboard and Mouse Mode 4-26
4.14. Creating Formations 4-27
4.14.1. Formation Subgroups 4-30
4.14.2. Formation Roles 4-31
4.15. Creating Groups 4-32
4.16. Expressive Faces 4-33
4.17. Character Labels and 3D View Labels 4-34
4.17.1. Enabling and Disabling Default Character Labels 4-35
4.17.2. Editing the Style of Character Labels 4-36
4.17.3. Labels and the DI-Guy API 4-38
4.17.4. Creating Screen and Button Labels 4-39
4.18. Configuring Chains 4-44

Chapter 5. Paths and Waypoints

5.1. Creating and Managing Paths 5-2
5.1.1. Creating a Path 5-2
5.1.2. Selecting a Path 5-4
5.1.3. Hiding Path Elements 5-4
5.1.4. Path Draw Style 5-4
5.1.5. Dead Space 5-4
5.2. Creating and Editing Waypoints 5-5
5.2.1. Creating Waypoints in the 3D View Window 5-6
5.2.2. Creating Waypoints Using the Waypoint Page 5-6
5.2.3. Selecting Waypoints 5-7
5.2.4. Editing Waypoints in the 3D View 5-8
5.2.5. Editing Waypoints on the Waypoint Page 5-8
5.2.6. Moving a Group of Waypoints 5-9
5.2.7. Ground Clamping Waypoints 5-10
5.3. Using Branched Paths 5-10

Chapter 6. Character-Level Actions

6.1. Creating and Managing Action Beads 6-2
6.1.1. Creating Action Beads 6-2
6.1.2. Editing Action Beads 6-5
6.1.3. Moving Action Beads 6-6
6.1.4. Pointing the Camera at an Action 6-7
6.1.5. Non-Forward Actions and Companion Waypoints 6-7
6.2. The Action Spreadsheet 6-9
6.2.1. Inserting an Action Using the Action Spreadsheet 6-10
6.2.2. Editing an Action in the Action Spreadsheet 6-10
6.2.3. Deleting an Action in the Action Spreadsheet 6-11

6.2.4. Stretch and Shrink 6-11
6.3. Decision Beads 6-11
6.3.1. Creating a Decision Bead 6-12
6.3.2. Creating Character-Level Decision Logic 6-13
6.3.3. Converting Decision Beads to Script Beads 6-17
6.4. Writing Script Beads 6-18
6.5. Creating Aim Beads 6-19
6.6. Creating Gaze Beads 6-21
6.7. Gestures 6-23
6.7.1. Head Nods and Shakes 6-23
6.7.2. Generic Gestures 6-23
6.7.3. Previewing Gestures 6-24

Chapter 7. Scenario Objects

7.1. Scenario-Level Objects 7-2
7.2. Guides 7-3
7.3. Adding Special Effects (Particle Systems) 7-5
7.3.1. Adding a New Particle System 7-7
7.3.2. Particle System Parameters 7-8
7.4. Creating Path Shapes 7-11
7.5. Sensor Regions 7-12
7.5.1. Creating Sensor Regions 7-13
7.5.2. Editing Sensor Regions 7-15
7.6. Creating Signals 7-17
7.6.1. Tracking Signal Use and Triggering Signals 7-18
7.7. Adding Sound Effects 7-19
7.7.1. Setting the Sound Volume 7-20
7.8. Creating Scenario Variables 7-21

Chapter 8. Using the Camera

8.1. Moving the Camera 8-2
8.1.1. Setting the Direction of Camera Motion 8-2
8.1.2. Changing the Camera's Speed in Mouse Move Mode 8-2
8.2. Teleporting the Camera 8-3
8.3. Attaching the Camera to a Character 8-4
8.4. Tracking a Character 8-5
8.5. Setting Up Point of View (POV) Mode 8-5
8.6. Saving and Loading Camera Settings 8-5
8.6.1. Saving Camera Settings 8-6
8.6.2. Loading Camera Settings 8-6
8.6.3. Changing the Order of Camera Settings 8-7
8.7. Advanced Camera Settings 8-7
8.7.1. Changing the Projection Mode 8-7
8.7.2. Changing the Camera's Speed for Projection Modes 8-7
8.7.3. Setting the Clipping Planes and Field of View 8-8
8.8. Specifying Camera Override Preferences 8-8

8.9. Viewing a Scenario in Symbolic View 8-9

Chapter 9. Events and Event Handlers

9.1. Introduction to Events and Event Handlers 9-2
9.1.1. Event Types 9-3
9.2. Creating Scenario-Level Decision Logic 9-3
9.2.1. Converting Decisions to Scripts 9-6
9.3. Writing a Scenario-Level Script 9-7
9.4. Adding a Library Function 9-9
9.5. Mapping Events to Callbacks and Logic 9-11
9.6. Creating Information Popups 9-13
9.6.1. Opening an Info Popup from a Script 9-14
9.7. Creating an Interaction Machine 9-14
9.7.1. Example Interaction Machine 9-15
9.7.2. Hot Objects 9-17

Chapter 10. Configuring Performance

10.1. Performance Tuning 10-2
10.2. Setting the Tick Rate 10-3
10.2.1. Specifying the Default Number of Ticks between Checks .. 10-4
10.2.2. Skipping Ticks 10-4
10.3. LOD Tuning 10-4
10.3.1. LOD Tuning for Terrain 10-5
10.3.2. Graphics Levels of Detail 10-6
10.4. Demand Loading for Terrains 10-6
10.5. Culling and Draw Management 10-8
10.5.1. Camera Frustum Culling 10-9
10.5.2. Distance Culling 10-10
10.6. Load Manager 10-10
10.7. Specifying the Drawing Preferences 10-12
10.8. Specifying Geometry Loading Preferences 10-13

Chapter 11. Terrains and Environment

11.1. Terrain Models 11-2
11.1.1. Loading a Terrain 11-3
11.2. Positioning the Terrain Database on the Earth's Surface 11-5
11.2.1. UTM and the Exercise Location 11-6
11.3. Paging Terrains 11-7
11.3.1. LOD Tuning 11-9
11.4. Displaying Terrain Coordinates for an Object 11-9
11.5. Configuring Fog 11-11
11.5.1. Adding a Fog Setting 11-12
11.5.2. Editing Fog Settings 11-13
11.6. Configuring Lighting Settings 11-13
11.6.1. Adding a Light Setting 11-14
11.6.2. Editing Lighting Settings 11-15

Chapter 12. Networking

12.1. Networking in DI-Guy Scenario 12-2
12.2. Joining a DIS Exercise 12-2
12.2.1. DI-Guy Custom PDUs 12-4
12.3. DI-Guy Scenario Entity Mapping 12-4
12.3.1. Adding a Model Mapping 12-5
12.3.2. Deleting a Model Mapping 12-6
12.3.3. Reloading Model Mappings 12-6
12.4. HLA Support 12-7
12.4.1. RPR-FOM Elements and DI-Guy Scenario FOM Extensions 12-7

Appendix A. Troubleshooting

A.1. Troubleshooting FAQ A-2

Appendix B. Keyboard Shortcuts

B.1. Keyboard Shortcuts and Hot Keys B-2
B.1.1. Hot Keys B-2

Appendix C. Callback IDs

C.1. Callback IDs C-2
C.1.1. Character Callbacks C-2
C.1.2. Scenario Callbacks C-4
C.1.3. Sensor Region Callbacks C-4
C.1.4. Signal Callbacks C-5
C.1.5. Crowd Callbacks (Requires DI-Guy AI) C-5

Appendix D. Using the Lua Editor

D.1. The DI-Guy Scenario Lua Code Editor D-2
D.1.1. Simple Auto Complete Based on Document Inspection D-2
D.1.2. Autocomplete for DI-Guy Constants D-2
D.1.3. Autocomplete for DI-Guy Object Functions D-2
D.1.4. Calltips D-3
D.1.5. Full Documentation in the API Viewer D-3
D.1.6. Querying Lua for Related Objects and Fields D-4

Appendix E. Switches and DOF in Scene Objects

E.1. Activating Switches and DOF in Scene Objects E-2
E.1.1. Scene Object Behavior File Keywords E-2
E.1.2. Specifying Switches E-2
E.1.3. Specifying DOF E-3
E.1.4. Example Scene Object Behavior File E-4

DI-Guy Glossary

Index

Preface

This manual is written for persons who will use DI-Guy Scenario. The manual assumes that you are familiar with your operating system and that you know how to work in your computer's graphical window environment.

Please see release documentation for information about changes and updates to DI-Guy since this manual went to press.

How This Manual is Organized

The chapters are organized as follows:

Chapter 1, *Introduction to DI-Guy Scenario* briefly describes DI-Guy Scenario and its features.

Chapter 2, *Starting DI-Guy Scenario* briefly describes how to start DI-Guy Scenario and configure preferences..

Chapter 3, *DI-Guy Scenario Tutorials* guides you through the process of creating scenarios, adding characters and actions, and some more advanced DI-Guy Scenario features.

Chapter 4, *Creating and Editing Characters* how to create characters and manipulate them.

Chapter 5, *Paths and Waypoints* explains how to create paths and add waypoints.

Chapter 6, *Character-Level Actions* explains how to assign actions to characters using action beads and scripts.

Chapter 7, *Scenario Objects* describes scenario-level objects such as guides, sensor regions, and sound effects.

Chapter 8, *Using the Camera* explains how to move the camera through the scene and how to save and recall camera settings.

Chapter 9, *Events and Event Handlers* explains how to create scenario-level decision logic, scripts, and function calls.

Chapter 10, *Configuring Performance* explains how to optimize DI-Guy Scenario performance.

Chapter 11, *Terrains and Environment* describes DI-Guy Scenario terrain support, how to load terrain, and how to change environmental settings such as fog and lighting.

Chapter 12, *Networking* explains how to connect DI-Guy Scenario to DIS and HLA networks using the optional networking module.

Appendix A, *Troubleshooting* provides solutions to common problems.

Appendix B, *Keyboard Shortcuts* lists the keyboard shortcuts that you can use to quickly access DI-Guy Scenario functions.

Appendix C, *Callback IDs* lists some of the commonly used callback IDs for events.

Appendix D, *Using the Lua Editor* briefly describes the Lua editor that is embedded in the script pages.

Appendix E, *Switches and DOF in Scene Objects* briefly describes how to use switches and DOF nodes in OpenFlight models.

DI-Guy Glossary defines terms that have special meaning in DI-Guy Scenario.

Documentation

DI-Guy comes with the following documentation:

- *DI-Guy SDK Developers Guide*. Explains how to program using the DI-Guy SDK.
- *DI-Guy Scenario Users Guide*. Explains how to use DI-Guy Scenario to create scenarios with realistic human activity.
- *DI-Guy Content Reference Manual*. Explains how to customize DI-Guy characters and add new character models.
- *DI-Guy AI Users Guide*. Explains how to use DI-Guy AI to add intelligent behavior to DI-Guy characters.
- *DI-Guy Motion Editor Users Guide*. Explains how to import, edit, and customize DI-Guy motions.
- *DI-Guy Expressive Faces Users Guide*. Provides a brief introduction to using the Expressive Faces module.
- *DI-Guy Master Index*. A PDF master index to the PDF manuals.
- DI-Guy Master Index. An HTML page that links to all documentation.
- Class documentation. HTML documentation describing all classes and functions.

These documents are available from the start menu under DI-Guy, or by browsing to *./doc*. Of particular interest is the DI-Guy Documentation Master Index, which lets you quickly access all the documents above along with additional technical information about DI-Guy.

Derivative Work

Any DI-Guy content (model, texture, motion, and so on) that you modify or copy is a derivative work that is protected under copyright law by the same rules that apply to content delivered with DI-Guy. This derivative content may be used only on computers that have a valid DI-Guy license. Please contact VT MÄK to discuss licensing for other uses of derivative models.

MÄK Products

DI-Guy Scenario is a member of the VT MÄK line of software products designed to streamline the process of developing and using networked simulated environments. The VT MÄK product line includes the following:

- VR-Link® Network Toolkit. VR-Link is an object-oriented library of C++ functions and definitions that implement the High Level Architecture (HLA) and the Distributed Interactive Simulation (DIS) protocol. VR-Link has built-in support for the RPR FOM and allows you to map to other FOMs. This library minimizes the time and effort required to build and maintain new HLA or DIS-compliant applications, and to integrate such compliance into existing applications.

  VR-Link includes a set of sample debugging applications and their source code. The source code serves as an example of how to use the VR-Link Toolkit to write applications. The executables provide valuable debugging services such as generating a predictable stream of HLA or DIS messages, and displaying the contents of messages transmitted on the network.

- MÄK RTI. An RTI (Run-Time Infrastructure) is required to run applications using the High Level Architecture (HLA). The MÄK RTI is optimized for high performance. It has an API, RTIspy®, that allows you to extend the RTI using plug-in modules. It also has a graphical user interface (the RTI Assistant) that helps users with configuration tasks and managing federates and federations.
- VR-Forces®. VR-Forces is a computer generated forces application and toolkit. It provides an application with a GUI, that gives you a 2D and 3D views of a simulated environment.

  You can create and view local entities, aggregate them into hierarchical units, assign tasks, set state parameters, and create plans that have tasks, set statements, and conditional statements. VR-Forces also functions as a plan view display for viewing remote entities taking part in an exercise. Using the toolkit, you can extend the VR-Forces application or create your own application for use with another user interface.

- VR-Vantage™. VR-Vantage is a line of products designed to meet your simulation visualization needs. It includes three end-user applications (VR-Vantage Stealth, VR-Vantage PVD, and VR-Vantage IG) and the VR-Vantage Toolkit.
  - VR-Vantage Stealth displays a realistic, 3D view of your virtual world, a 2D plan view, and an exaggerated reality (XR) view. Together these views provide both situational awareness and the big picture of the simulated world. You can move your viewpoint to any location in the 3D world and can attach it to entities so that it moves as they do.
  - VR-Vantage IG is a configurable desktop image generator (IG) for out the window (OTW) scenes and remote camera views. It has most of the features of the Stealth, but is optimized for its IG function.
  - VR-Vantage PVD provides a 2D plan view display. It gives you the big picture of the simulated world.

  - The VR-Vantage Toolkit is a 3D visual application development toolkit. Use it to customize or extend MÄK's VR-Vantage applications, or to integrate VR-Vantage capabilities into your custom applications. VR-Vantage is built on top of OpenSceneGraph (OSG). The toolkit includes the OSG version used to build VR-Vantage.

- MÄK Data Logger. The Data Logger, also called the Logger, can record HLA and DIS exercises and play them back for after-action review. You can play a recorded file at speeds above or below normal and can quickly jump to areas of interest. The Logger has a GUI and a text interface. The Logger API allows you to extend the Logger using plug-in modules or embed the Logger into your own application. The Logger editing features let you merge, trim, and offset Logger recordings.
- VR-Exchange™. VR-Exchange allows simulations that use incompatible communications protocols to interoperate. For example, within the HLA world, using VR-Exchange, federations using the HLA RPR FOM 1.0 can interoperate with simulations using RPR FOM 2.0, or federations using different RTIs can interoperate. VR-Exchange supports HLA, TENA, and DIS translation.
- VR-TheWorld™ Server. VR-TheWorld Server is a simple, yet powerful, web-based streaming terrain server, developed in conjunction with Pelican Mapping. Delivered with a global base map, you can also easily populate it with your own custom source data through a web-based interface. The server can be deployed on private, classified networks to provide streaming terrain data to a variety of simulation and visualization applications behind your firewall.
- VR-inTerra. VR-inTerra is a C++ API for adding terrain agility to applications. It can load, page, or stream terrain from a wide variety of formats or sources into a single, consistent run-time representation, consisting of a collision graph and vector network.
- DI-Guy. The DI-Guy product line is a set of software tools for real-time human visualization, simulation, and artificial intelligence. Every DI-Guy software offering comes with thousands of ready-to-use characters, appearances, and motions. DI-Guy enables the easy creation of crowds and individuals who are terrain aware, autonomous, and react intelligently to ongoing events. Save time, money and create outstanding simulations with DI-Guy. The DI-Guy product line includes the following products:
  - The DI-Guy SDK. Embed the DI-Guy library in your real-time application and populate your world with lifelike human characters.
  - DI-Guy Scenario. Author and visualize human performances in a rich, user-friendly graphical environment. Use DI-Guy Scenario as an end visualization application or save scenarios and load them into your DI-Guy SDK enabled application.
  - DI-Guy AI. Generate crowds of autonomous characters to quickly populate your worlds with hundreds and thousands of terrain-aware, collision avoiding DI-Guys. Used as a module on top of DI-Guy Scenario and DI-Guy SDK.
  - Expressive Faces Module. Enable DI-Guy characters to have faces that display emotion, eyes that look in directions and blink, and lips that sync to sound files.

– DI-Guy Motion Editor. Create or customize motions to your particular needs in an easy-to-use graphical application.

How to Contact Us

For DI-Guy technical support, information about upgrades, and information about other MÄK products, you can contact us in the following ways:

Telephone

Call or fax us at:	Voice:	617-876-8085 (extension 3 for support)
	Fax:	617-876-9208

E-mail

Sales and upgrade information:	info@mak.com
Technical support:	support@mak.com
VR-Vantage support:	vrv-support@mak.com

Internet

MÄK web site home page:	www.mak.com
License key requests:	www.mak.com/support/get-licenses.html
Product version and platform information:	www.mak.com/support/product-versions.html
For the free, unlicensed MÄK RTI:	www.mak.com/resources/bonus-material/cat_view/16-bonus-materials/24-mak-high-performance-rti.html
MÄK Community Forum:	www.mak.com/community-forum/1-forum.html

Post

Send postal correspondence to:	VT MÄK 150 Cambridge Park Drive, 3rd Floor Cambridge, MA, USA 02140

When requesting support, please tell us the product you are using, the version, and the platform on which you are running.

Document Conventions

This manual uses the following typographic conventions:

`Monospaced`	Indicates commands or values you enter.
`Monospaced Bold`	Indicates a key on the keyboard.
`Monospaced Italic`	Indicates command variables that you replace with appropriate values.
Blue text	A hypertext link to another location in this manual or another manual in the documentation set.
`{ }`	Indicates required arguments.
`[ ]`	Indicates optional arguments.
|	Separates options in a command where only one option may be chosen at a time.
`(	)`	In command syntax, indicates equivalent alternatives for a command-line option, for example, `(-h | --help)`.
/	Indicates a directory. Since MÄK products run on both Linux and Windows PC platforms, we use the / (slash) for generic discussions of pathnames. If you are running on a PC, substitute a \ (backslash) when you type pathnames.
Italic	Indicates a file name, pathname, or a class name.
sans Serif	Indicates a parameter or argument.
➤	Indicates a one-step procedure.
Menu → Option	Indicates a menu choice. For example, an instruction to select the Save option from the File menu appears as: Choose **File → Save**.
	Click the icon to run a tutorial video in the default browser.
	Indicates supplemental or clarifying information.
	Indicates additional information that you must observe to ensure the success of a procedure or other task.

Directory names are preceded with dot and slash characters that show their position with respect to the DI-Guy home directory. For example, the directory *vrforces13/doc* appears in the text as *./doc.*

DI-Guy Conventions

Table 2-1 lists the conventions used for entity coordinates in DI-Guy documentation.

Table 2-1: Coordinate system conventions

Convention	Indicates
XYZ	X = forward, Y = to the left Z = up
Yaw, Roll, Pitch	Orientation is applied in the order YRP. Yaw = rz – orientation about the vertical axis Roll = rx – orientation about the fore-aft axis Pitch = ry – orientation nose down, nose up

Mouse Button Naming Conventions

An instruction to click the mouse button, refers to clicking the primary mouse button, usually the left button for right-handed mice and the right button for left-handed mice. The context-sensitive menu, also called a popup menu or right-click menu, refers to the menu displayed when you click the secondary mouse button, usually the right button on right-handed mice and the left button on left-handed mice.

Third Party Licenses

MÄK software products may use code from third parties. This section contains the license documentation required by these third parties.

Boost License

VR-Link, and all MÄK software that uses VR-Link uses some code that is distributed under the Boost License. All header files that contain Boost code are properly attributed. The Boost web site is: www.boost.org.

Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

libXML and libICONV

VR-Link and all MÄK software that uses VR-Link, links in libXML and libICONV. On some platforms the compiled libraries and header files are distributed with MÄK Products. MÄK has made no modifications to these libraries. For more information about these libraries please see the following web sites:

- The LGPL license is available at: http://www.gnu.org/licenses/lgpl.html.
- Information about IconV is at: http://www.gnu.org/software/libiconv/.
- Information about LibXML is at: http://xmlsoft.org/.

Lua

VR-Forces and DI-Guy use the Lua programming language (www.lua.org). Its license is as follows:

Copyright © 1994–2012 Lua.org, PUC-Rio.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1. Introduction to DI-Guy Scenario

This chapter introduces DI-Guy Scenario.

Introduction 1-2
- What is DI-Guy Scenario? 1-2
- DI-Guy Scenario is Real Time 1-3
- DI-Guy Scenario Comes with People 1-3
- DI-Guy AI 1-4
- DI-Guy Scenario is Easy to Use 1-4
- Supported Platforms 1-4
- Optional Components for DI-Guy Scenario 1-5

Vehicles 1-6

Expressive Faces 1-7

Customizing DI-Guy Content 1-7

1.1. Introduction

Suppose you have a terrain model for a battlefield, facility, ship, village, building complex, factory, or city. Your job is to populate it with life-like people and vehicles. You will need people in different sizes and shapes, who move realistically in real time performing tasks central to your simulation or serving as background traffic. How long would it take to populate the scenario using the tools you currently have available? With DI-Guy Scenario you can do it in about a day.

DI-Guy Scenario only runs on Microsoft Windows.

1.1.1. What is DI-Guy Scenario?

DI-Guy Scenario is a tool for adding life-like human characters to real-time simulations. It lets you rapidly populate scenarios with people who move realistically, go in and out of buildings, and travel throughout the terrain. You can use DI-Guy Scenario for ground and urban combat, mission planning, flight deck training, law enforcement and first responder training, site security planning, architectural visualization, driving simulators, marketing, driving simulators, and many other purposes.

Figure 1-1. DI-Guy Scenario

1.1.2. DI-Guy Scenario is Real Time

Everything about DI-Guy Scenario is real-time. DI-Guy Scenario lets you edit activity in the scenario and see the people act out their roles immediately, without a compilation delay. In fact, you can edit a scenario while the scenario plays. DI-Guy Scenario also supports embedded run-times, which allows you to export the scenarios you create to your own real-time software or third-party systems.

1.1.3. DI-Guy Scenario Comes with People

DI-Guy Scenario comes with a full cast of life-like people. The characters move realistically, make natural transitions from one activity to the next, respond to high-level direction, and are generally intelligent about how they move. The DI-Guy Scenario people are completely operational and ready to go when you install the product. They go where you tell them to go, performing various tasks and actions. Each person has its own skeleton, texture-mapped geometry, behavior library, and real-time motion engine. DI-Guy Scenario seamlessly integrates the people, their behavior, and your environment.

Figure 1-2. DI-Guy characters

1.1.4. DI-Guy AI

DI-Guy AI, software for creating autonomous, terrain-aware characters and crowds, is an extra cost option to DI-Guy Scenario. Information about DI-Guy AI is in *DI-Guy AI Users Guide*. This manual focuses on the aspects of DI-Guy Scenario that do not use DI-Guy AI, but for a full experience you are encouraged to evaluate DI-Guy AI. There is a very strong coupling between the two products.

1.1.5. DI-Guy Scenario is Easy to Use

DI-Guy Scenario's graphical user interface makes it easy to use. Once you have loaded your terrain model into DI-Guy Scenario and you have entered DI-Guy Scenario's innovative PeopleBlitzer mode, you can add a person to the scene with a single click-and-drag of the mouse. In a few minutes you can add a variety of people to your scenario, all performing seemingly complex tasks. In an hour you can populate a whole region.

You can access a lot of DI-Guy Scenario functionality by pointing and clicking directly in the 3D scene where you place people, give them paths and assign tasks. Dedicated palettes let you to specify decision logic, control where characters look, create formations, and edit many other parameters. It is easy to go back and refine scenarios you or others have built.

To create a scenario using DI-Guy Scenario you perform four general steps:

1. Load a terrain model (DI-Guy Scenario supports Open Flight models).
2. Choose a character type and appearance.
3. Place the character on the terrain model and tell it where to go and what to do.
4. View the scenario.

1.1.6. Supported Platforms

DI-Guy Scenario runs on PC computers running Windows. Once scenarios are generated, you can load them on a wide variety of systems using the DI-Guy SDK. The DI-Guy SDK runs on both Windows and Linux computers using graphics libraries for OpenGL, OpenSceneGraph, DirectX, and Vega Prime. DI-Guy Scenario can also be customized to a specific graphics library using the DI-Guy Graphics API. Easy-to-use DI-Guy API functions are available for importing DI-Guy Scenario scenarios into your real time application.

1.1.7. Optional Components for DI-Guy Scenario

DI-Guy Scenario comes complete with terrains, people, and motions needed to create compelling human simulations. In addition, there are optional components for DIS/HLA networking, special effects, expressive faces, and plug-in capability. Contact your DI-Guy sales representative for information about ordering these options.

- DIS/HLA Networking. This option allows DI-Guy Scenario to support DIS and HLA networking. With networking enabled, DI-Guy Scenario can interoperate with SAFs, stealths, IGs, other DI-Guy Scenario installations, and other systems that use DIS or HLA.
- Expressive Faces. This module provides characters with faces that can show facial expressions, move their eyes, and synchronize their lips with speech.
- Plug-in support. DI-Guy Scenario offers a plug-in architecture that allows you to create your own dynamic libraries (DLLs) that link and load at runtime with DI-Guy Scenario. Typical applications include adding peripherals, AI, and custom C++ code.
- DI-Guy Motion Editor. DI-Guy Scenario comes with over 2,000 motions and transitions in its library. However, because human behavior is so diverse, you may want to modify some of these motions or add entirely new ones. That is now possible using the DI-Guy Motion Editor, an advanced graphical tool for adding new behavior to DI-Guy Scenario and DI-Guy Scenario. The DI-Guy Motion Editor has been engineered for seamless operation with DI-Guy Scenario.
- DI-Guy AI. Quickly create crowds composed of autonomous, terrain-aware DI-Guy characters who perform collision avoidance with each other, external entities, vehicles, and buildings and objects in the terrain. Please see *DI-Guy AI Users Guide* for more details.

1.2. Vehicles

DI-Guy Scenario includes generic vehicles with limited visual capabilities. It also has vehicles called "complete vehicles," that have the following capabilities:

- Rolling and turning wheels. Wheels on complete vehicles roll and turn appropriately as the vehicle moves through a scenario, whether the vehicle is on a path or in free action mode.

  One important exception is that for vehicles moving in reverse along a path, the wheels will turn backwards, but they will not turn left or right. This is due to the fact that complete vehicles are "attached" to the path at the middle of their front axle, and it is not possible to keep the rear of a vehicle in the proper position on the path without taking that attachment point off of the path.

- Smooth acceleration and deceleration. Complete vehicle characters have the change_speed action, which denotes the section of a path in which the vehicle should speed up or slow down to meet the speed of the next action. Having change_speed actions close to other actions results in faster accelerations, while having them further away results in more gradual accelerations.

  If change_speed actions are not inserted between moving and still actions, the vehicle will do abrupt speed changes.

- Doors that open and close. Doors move independently of the vehicle's current action (moving forward, sitting still, and so on). Each door can be independently controlled.

  Doors are controlled through the use of gestures. Gestures in DI-Guy Scenario take over part of the articulations of a character while leaving many of them alone. Human characters, for example, have gestures that allow them to wave, point, and shrug, and other motions that can be triggered independently of their base action.

  In the case of complete vehicles, gestures affect door hinges and vehicle suspension instead of hands and arms. Gestures are easily begun using the API, scripting, and decision beads.

- Body that can rock. Similar to the opening of its doors, the vehicle can rock on its chassis through the use of a vehicle gesture. This can be used, for example, to show a crowd attacking a vehicle.
- Aimable turret. For vehicle appearances that have turrets (technicals, turreted humvees, and so on), the turret can be aimed at specific points or characters. The target of the turret is commonly set through the use of aim beads, though it can also be controlled using the API and scripting.

The example scenario, *Vehicle_Drop_Off.dss*, demonstrates complete vehicles.

1.3. Expressive Faces

Expressive Faces is an optional module available for DI-Guy Scenario. Expressive Faces are animated head graphics that can portray different emotional states, blink, and do lip-sync to speech. To activate the expressive faces module, you need to initialize and deinitialize the expressive face module and have a valid expressive faces license.

DI-Guy Expressive Faces uses FaceFX technology.

1.4. Customizing DI-Guy Content

DI-Guy comes with a wide variety of motions, textures, human models, vehicle models, and prop models. In addition to the motions, models, and textures provided with the system, you can add your own motions, models, and textures. For information about adding content, please see in *DI-Guy Motion Editor Users Guide* and in *DI-Guy Content Reference Manual*.

2. Starting DI-Guy Scenario

This chapter how to start DI-Guy Scenario and introduces scenarios.

Installing DI-Guy Scenario 2-3
Starting DI-Guy Scenario 2-3
- Starting DI-Guy Scenario from a Command Prompt 2-4
- Command-line Arguments 2-5
- Managing Plug-ins 2-7
- Specifying a Startup Script 2-8

Introduction to Scenarios 2-9
- Creating a Scenario 2-9
- Saving Scenarios 2-9
- Opening a Scenario 2-9
- Running Scenarios 2-10
- Specifying a Scenario As Read-Only 2-12
- Writing a Scenario Description 2-12
- Merging Scenarios 2-13
- Recording a Scenario to a Movie 2-13

The DI-Guy Scenario Log 2-15
Setting DI-Guy Scenario Preferences 2-16
Using Multiple 3D Windows 2-18
Common GUI Controls 2-19
- Adding a New Object 2-20
- Deleting Objects 2-20
- Copying Objects 2-20
- Teleporting the Camera to a Specific Object 2-20
- Changing the Index of an Object 2-21

Sharing Libraries Among Scenarios 2-21
Hiding and Displaying Objects..... 2-27

2.1. Installing DI-Guy Scenario

DI-Guy Scenario is installed as part of the DI-Guy installation process. For details, please see Chapter 2, *Installing DI-Guy*, in *DI-Guy Getting Started Users Guide*.

2.2. Starting DI-Guy Scenario

You can start DI-Guy Scenario in any of the following ways:

- Double-click the DI-Guy Scenario desktop icon.
- On the Start menu, choose **All Programs** → **DI-Guy Applications 13** → **DI-Guy Scenario**.
- In Windows Explorer, double-click a scenario file with the *.dss* extension.
- From a console window.

DI-Guy Scenario opens up with a default database in the 3D View (Figure 2-1). Several control windows also open. By default, the windows are bound to the main DI-Guy Scenario window. However, you can change that in the Preferences dialog box. For details, please see "Setting DI-Guy Scenario Preferences," on page 2-16.

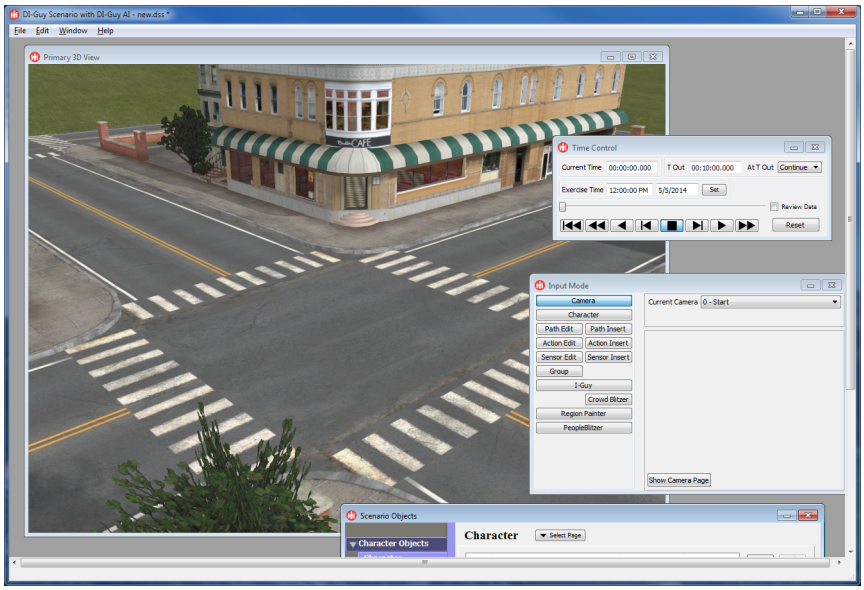


Figure 2-1. DI-Guy Scenario startup windows

2.2.1. Starting DI-Guy Scenario from a Command Prompt

To start DI-Guy Scenario from the command line:

1. Change to the drive in which DI-Guy Scenario was installed.

   ```
   cd c:\DI-Guy
   ```

2. Change to the *./scenarios* directory.

   ```
   cd scenarios
   ```

3. Run the demo scenario.

   ```
   diguyscenario scenario.dss
   ```

This manual assumes that DI-Guy Scenario has been installed in *C:\DI-Guy*. Paths start at the DI-Guy root and are shown as *./subdirectory*.

Dual Screen Mode

The recommended way to author in DI-Guy Scenario is to start in dual screen mode. This means that the graphics window is one screen and the authoring windows are on the second. Here are some typical startup commands for commanding this:

```
diguyscenario –fullscreen2 –hide_menu –hide_status
     –no_digs_overlay
diguyscenario –fullscreen
```

2.2.2. Command-line Arguments

DI-Guy Scenario has several command-line arguments. You can specify them by changing the target line of a DI-Guy Scenario shortcut, or by starting DI-Guy Scenario from a command line. The basic syntax is:

```
diguyscenario [options] [files]
```

Table 2-1 lists command-line options.:

Table 2-1: DI-Guy startup options

Option	Description
`-D`	Start DI-Guy Scenario in demo mode.
`-p play_mode`	Specifies the initial play mode when DI-Guy Scenario starts up. *play_mode* is a string: ◆ `-p play` means time will begin to advance as soon as the program is started. ◆ `-p stop` (the default) means time will not advance until you click the play button.
`-fullscreen`	DI-Guy Scenario starts up with the 3D View filling the entire screen of the display, without a border.
`-fullscreen2`	DI-Guy Scenario starts up with the 3D View filling the entire screen of the display, without a border, on the second screen.
`-hide_menu`	Hide the menu bar.
`-hide_status`	Hide the status bar.
`-net`	If the network module is enabled, connect to the network immediately.
`-N notify_level`	Specifies verbosity of log file output: ◆ 1 = nothing ◆ 15 = everything.
`-L notify_level`	Specifies verbosity of the Log window: ◆ 1 = nothing ◆ 15 = everything.
`-f log_filename`	Send log information to the specified file. The default is not to save log information to a file.
`-C cfg_filename`	Start DI-Guy Scenario with the specified configuration file. Default: *diguy.cfg*. The configuration file is typically placed relative to *./config/*.
`-V cfg_filename`	Use the specified configuration file as the network configuration file. Default: *./scenario/network.cfg*.

Table 2-1: DI-Guy startup options

Option	Description
file1 ... filen	Scenario files (*.dss* extension) to be opened when DI-Guy Scenario starts. The first file is opened as if by **File** → **Open**. The rest are opened as if by **File** → **Merge**. For details about merging scenarios, please see "Merging Scenarios," on page 2-13.
`+plugin` *plugin*	Add specified plug-in (DLL).
`-plugin` *plugin*	Remove specified plug-in (DLL).
`+module` *module*	Add specified module. Options are: ◆ script_lua (enabled by default). ◆ exface (enabled by default). ◆ sound_openal (enabled by default). ◆ particles (enabled by default). ◆ net_dis (enabled by default). ◆ net_hla.
`-module` *module*	Remove specified module (as listed for `+module`).
`-pmu`	Have mouse_func() called in plug-in when in plug-in mode.

Examples:

Start the *demo.dss* scenario and begin immediate playback:

```
diguyscenario -p play -N 0 demo.dss
```

Open the *demo.dss* scenario with verbose information sent to the *log.txt* file:

```
diguyscenario -N 6 -f log.txt demo.dss
```

2.2.3. Managing Plug-ins

Some scenarios require software modules called plug-ins. Plug-ins are a simple way for developers to add functionality to DI-Guy Scenario without rebuilding the entire application. You can specify which plug-ins a scenario requires and can manage their use. If you specify that a scenario requires a plug-in and you try to run it without the plug-in, DI-Guy Scenario issues a warning.

You can load plug-ins using the `+plugin` command-line argument, from an initialization script, or as described in this section. For details about using an initialization script, please see "Specifying a Startup Script," on page 2-8.

Plug-ins require a separate license. For details about writing plug-ins, please see Chapter 6, *Writing Plug-ins for DI-Guy Scenario*, in *DI-Guy SDK Developers Guide*.

To add a plug-in to a scenario:

1. Choose Window → Scenario Objects. The Scenario Objects window opens.
2. Expand the Specialized Objects tab.
3. Select the Plugins page (Figure 2-2).

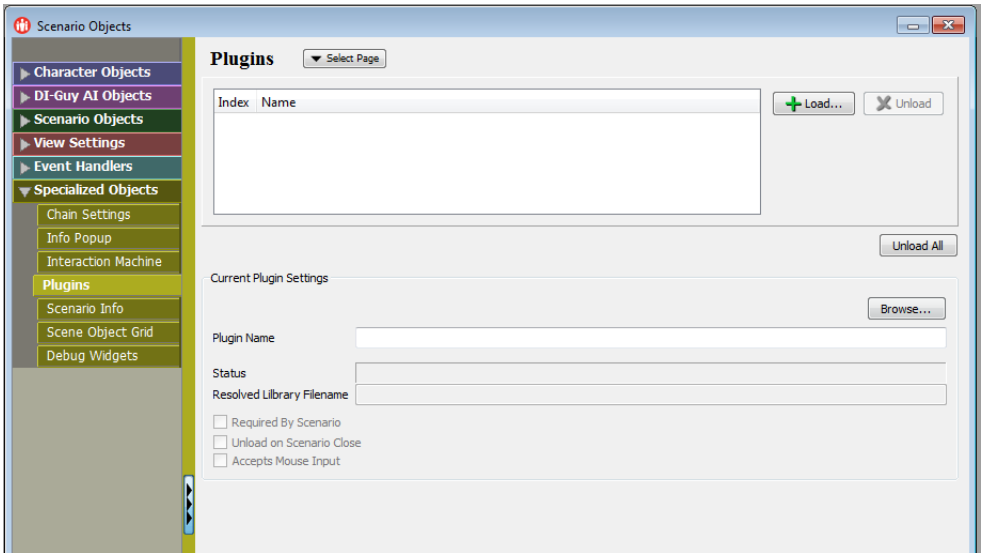


Figure 2-2. Plugins page

In the rest of this manual, we refer to pages in the Scenario Objects window by their tab and page, for example, Scenario Objects:Plugins.

4. Click Load. A file browser opens.
5. Select the DLL that you want to load.
6. Click Open. The plug-in gets added to the list.

7. Optionally, to specify that the plug-in is required, select the Required By Scenario check box.

2.2.4. Specifying a Startup Script

You can specify that DI-Guy Scenario run a script when it initializes. The following is an example Lua script:

```
if this app:is_debug() then
  print("Installing debug plugin DLL.");
  this_scenario:load_plugin("plugin_debug");
else
  print("Installing release plugin DLL.");
  this_scenario:load_plugin("plugin_release");
end
```

- You do not need to include the *.dll* suffix to load the DLL.
- The release/debug status of the DLL must match that of DI-Guy Scenario.
- Only the DLL-enabled version of DI-Guy Scenario works with plug-ins.

To specify a startup script:

1. Select the Event Handlers:Script page (Figure 2-3).

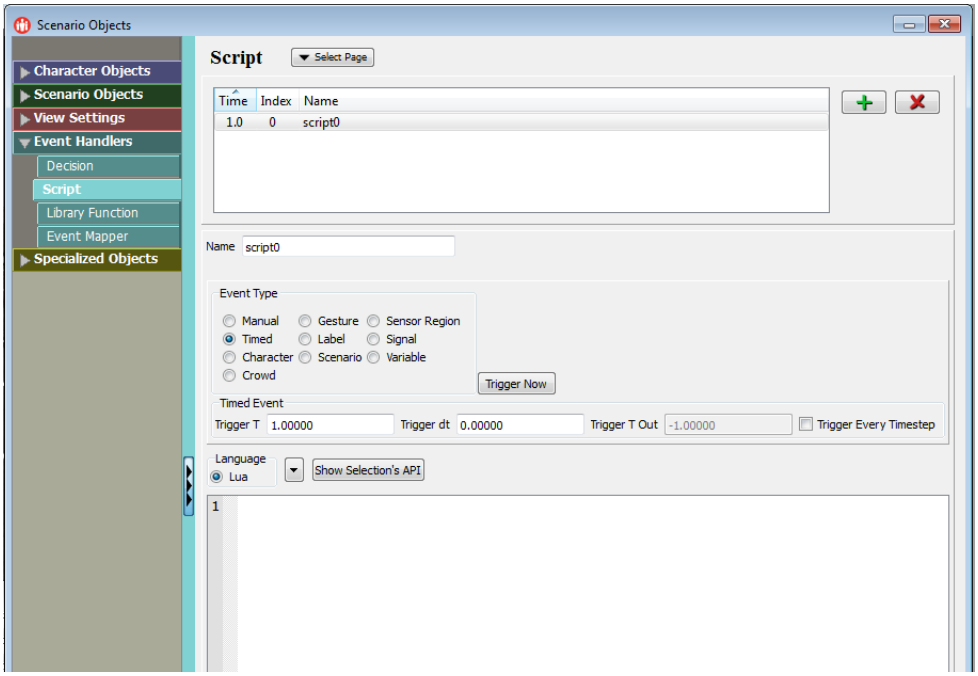


Figure 2-3. Script page

2. In the Init. Script Filename box, type the path for the script that you want to run.
3. Save the scenario.

2.3. Introduction to Scenarios

All the activity in a DI-Guy Scenario simulation takes place in the context of a scenario. This includes all the characters, terrain, simulation objects, scripts, and so on needed to simulate human activity in the world. Each scenario you create is distinct. They do not share specific characters, scripts, and so on. However, more than one scenario can use the same terrain data and they all draw on the same libraries of characters and other objects.

2.3.1. Creating a Scenario

To create a new scenario:

1. Choose **File** → **New**. A new scenario opens using the default terrain.
2. Optionally, open a different terrain, as described in "Loading a Terrain," on page 11-4.
3. Add characters, actions, scripts, and other scenario elements as described in this manual.

2.3.2. Saving Scenarios

After you create a scenario, you must save it or you will lose all your work. If you are running a previously saved scenario, the scenario filename is listed in the title bar. An asterisk (*) next to the filename means the scenario has not been saved since last modified.

To save a scenario:

1. Choose **File** → **Save**. If the scenario has been saved previously, it is saved. If it has not been saved previously, the Save As dialog box opens.
2. Type a name for the scenario.
3. Click Save.

2.3.3. Opening a Scenario

To open a scenario:

1. Choose **File** → **Open**. The Open dialog box opens.
2. Select the scenario you want to open.
3. Click Open.

2.3.4. Running Scenarios

To see characters move and interact with each other, you must run the scenario. Scenarios are managed in the Time Control window (Figure 2-4).

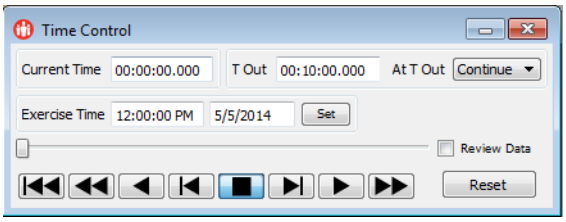


Figure 2-4. Time Control window

Table 2-2 describes the controls on the Time Control window.

Table 2-2: Time Control window controls

Control	Description
Current Time	Current elapsed scenario time. Can be displayed as HH:MM::SS or in seconds. For details, please see "Setting DI-Guy Scenario Preferences," on page 2-16.
T Out	The scenario time at which to stop playing or loop, if At T Out is set to one of these options. Can be displayed as HH:MM::SS or in seconds. For details, please see "Setting DI-Guy Scenario Preferences," on page 2-16.
At T Out	Specifies what DI-Guy Scenario should do when the scenario reaches its end time. Options are: ◆ Continue. Keep playing the scenario. ◆ Loop. Reset to the beginning and start playing again. ◆ Stop. Stop running the scenario.
Exercise Time	Real-world date and time. You can hide the exercise time display. For details, please see "Setting DI-Guy Scenario Preferences," on page 2-16.
Time slider	Lets you move to a particular time in the time range.
Review Data	Record scenario data for after-action-review and reverse playback.
Control buttons	Run the scenario forward and backward: ◆ |◀◀. Rewind to start time. ◆ ◀◀. Fast reverse. Default: 4X. You can configure this on the Preferences dialog box. ◆ ◀. Reverse playback. ◆ |◀. Reverse playback frame-by-frame (.033 seconds). ◆ ■. Stop playback. ◆ ▶|. Advance frame-by-frame (.033 seconds). ◆ ▶. Play. ◆ ▶▶. Fast forward. Default: 4X. You can configure this on the Preferences dialog box.
Reset	Stop playback and return to the starting state of the scenario.

➤ To play a scenario, in the Time Control window, click the Play button (▶). Use the other buttons to manipulate playback.

2.3.5. Specifying a Scenario As Read-Only

You can specify that a scenario be read-only so that it cannot be changed without undoing the read only status.

To specify that a scenario be read only:

1. Select the Specialized Objects:Scenario Info page (Figure 2-5).
2. Select the Scenario Is Read Only check box.

2.3.6. Writing a Scenario Description

You can write a description of a scenario and save it with the scenario. This can benefit future users or even yourself if you do not recall the details of the scenario.

To write a scenario description:

1. Select the Specialized Objects:Scenario Info page (Figure 2-5).

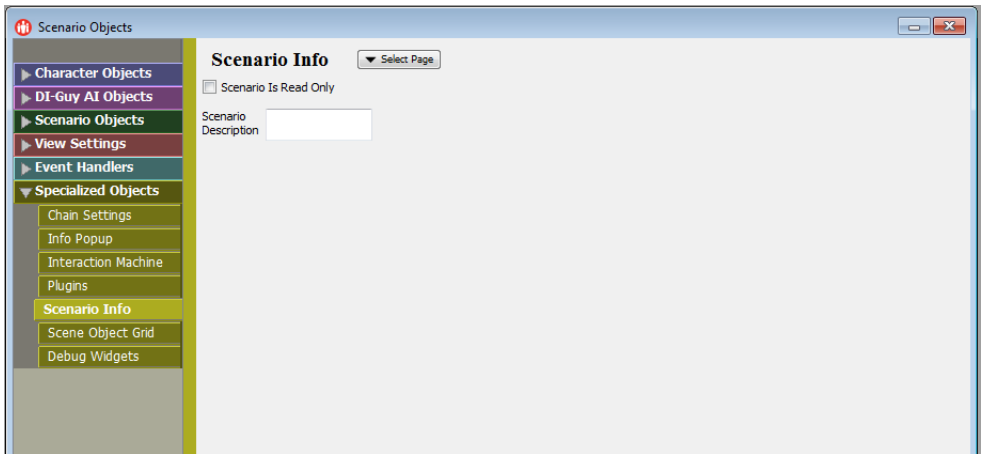


Figure 2-5. Scenario Info page

2. In the Scenario Description box, type a description of the scenario.
3. Optionally, specify that the scenario be read only.

2.3.7. Merging Scenarios

You can merge scenarios into one larger scenario. This allows you to easily build scenarios incrementally or to collaborate with other scenario builders to develop a large scenario.

When you merge scenarios, if any character or scene object names are duplicated, the object in the scenario being merged in is renamed.

To merge scenarios:

1. Open one of the scenarios to be merged.
2. Choose **File** → **Merge**. A file browser window opens.
3. Select the scenario you want to merge into the first scenario.
4. Click Open.
5. Repeat the procedure as needed to merge in additional scenarios.

2.3.8. Recording a Scenario to a Movie

DI-Guy Scenario lets you create high-quality digital videos of your scenarios. You can choose from several common formats. DI-Guy Scenario movies do not include audio. If you want to add audio, you must import DI-Guy Scenario movies or the underlying sequence of images into a video editing tool that allows you to add audio.

A movie records only what is visible in the 3D View window as it is recorded. By contrast when you save review data for after-action-review, when you play it back, you can navigate throughout the scenario.

To record a scenario to a movie:

1. Create the scenario.
2. Choose **Window** → **Movie Settings**. The Movie Settings dialog box opens (Figure 2-6).

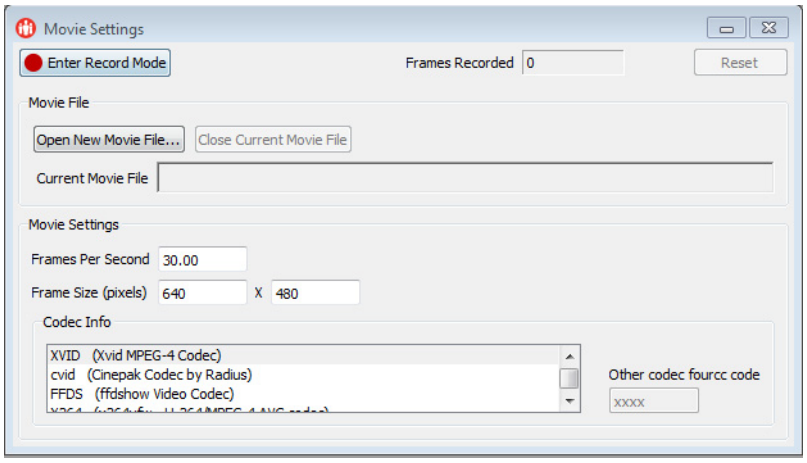


Figure 2-6. Movie Settings dialog box

3. Click Open New Movie File. A file browser opens.
4. Type a name for the movie.
5. Click Save.
6. Optionally, in the Codec Info list, select the codec you want to use.
7. Optionally, specify the frames per second and frame size.
8. Start the scenario.
9. Click Enter Record Mode. The button text changes to Exit Record Mode.
10. When you are finished recording, click Exit Record Mode.
11. Click Close Current Movie File.

2.4. The DI-Guy Scenario Log

The Log window prints runtime log information. You can adjust the level of detail of information printed. You can also clear the window. If you clear the window, all previously printed information is lost.

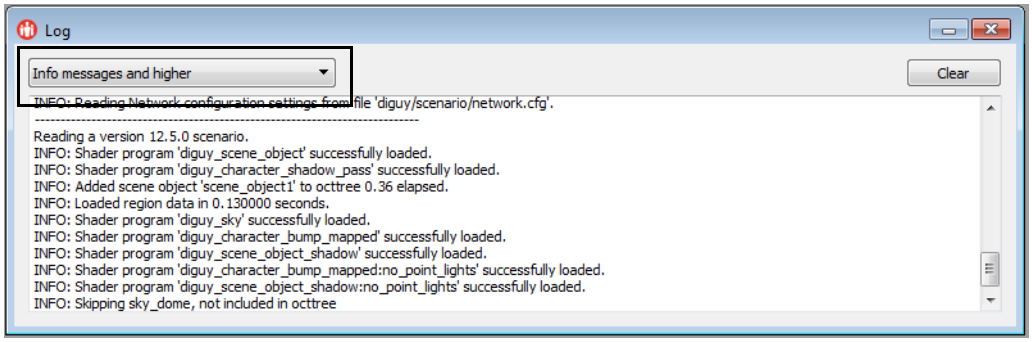


Figure 2-7. Log window

➤ To view the Log window, choose **Window** → **Log**.

➤ To set the notification level for the Log window, select an option from the list (Figure 2-7).

2.5. Setting DI-Guy Scenario Preferences

Preferences allow you to customize DI-Guy Scenario's behavior to your personal tastes.

To set preferences:

1. Choose **Edit** → **Preferences**. The Preferences dialog box opens (Figure 2-8).

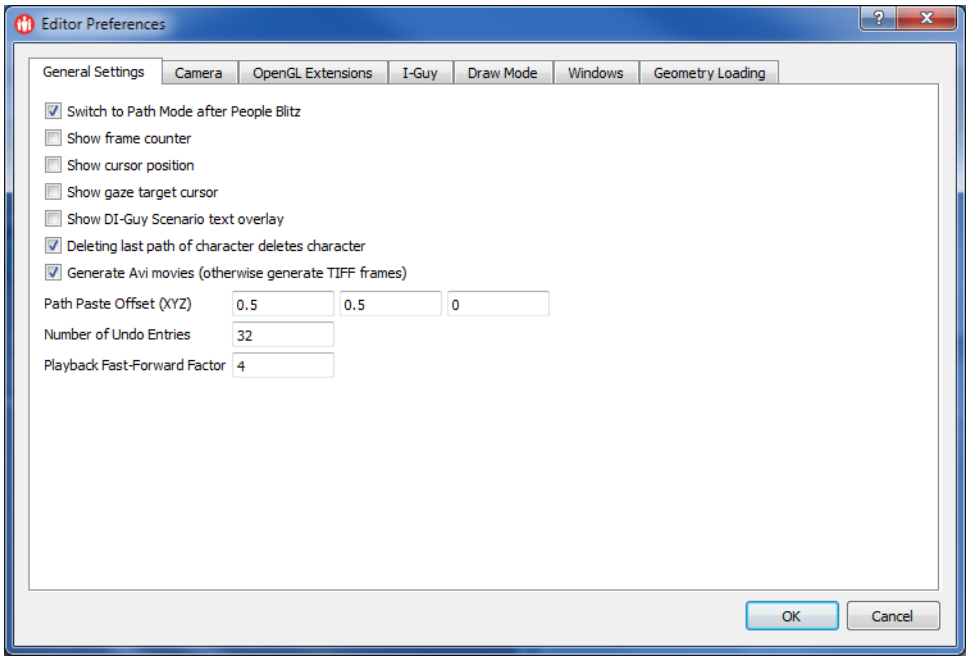


Figure 2-8. Preferences dialog box

2. Select the tab on which you want to change a preference. Table 2-3 describes the options on each tab.
3. Change the values as desired.
4. Click OK.

Table 2-3: Preferences dialog box settings

Parameter	Description
General tab	
Switch to Path Mode After PeopleBlitz	If selected, switch to Path Edit mode right after adding a character to the scene.
Show frame counter	If selected, the frame rate is displayed in the title bar of the 3D View.

Table 2-3: Preferences dialog box settings

Parameter	Description
Show cursor position	If selected, the 3D coordinates of the mouse cursor are displayed in an overlay of the 3D View. This tool is useful when examining the altitude of the terrain, or determining the location of an object in the scenario. Just point at the object and its coordinates are displayed.
Show gaze target cursor	If selected, a marker is displayed in the 3D View that shows the fixation point for the character's gaze.
Show DI-Guy Scenario Text Overlay	If selected, displays a text overlay in the 3D View.
Deleting last path of character deletes character	If selected, when you delete a character's last path, the character gets deleted.
Generate AVI Movies	If selected, generates AVI movies. If clear, use a TIFF Frame generator.
Path Paste Offset (XYZ)	The amount to move a path when it is pasted.
Number of Undo Entries	The amount of history that DI-Guy Scenario tracks for the Undo feature. Tracking more entries increases memory use.
Playback Fast-Forward Factor	The speed that fast forward runs at.
Camera tab	
Camera collides with scene	If selected, the camera cannot pass through walls or the terrain.
Moving camera with mouse disables camera script calls	If selected, when you are in Camera mode and put the cursor in the scene, the mouse takes over control of the camera and overrides scripted camera events, if any.
Moving camera with mouse clears Camera "Look From" character	If selected, when you are in Camera mode and put the cursor in the scene, the mouse takes over control of the camera and overrides the Look From setting on the Camera page.
Moving camera with mouse clears Camera "Look At" character	If selected, when you are in Camera mode and put the cursor in the scene, the mouse takes over control of the camera and overrides the Look At setting on the Camera page.
I-Guy tab	
Joystick settings	Specify gain for joysticks.
Draw Mode tab	
Rendering Quality	The options at the top of each list have the lowest performance cost.

Table 2-3: Preferences dialog box settings

Parameter	Description
Polygon Fill Mode	Specify how polygons are drawn. Figure 10-8 illustrates the different modes.
Visuals	Enable and disable display of textures and materials.
Debug	Select options to help debug scenarios.
Windows tab	
Float Windows	Selected windows are not bound by the DI-Guy Scenario main window. You can drag them anyplace on the desktop.
Show Times in HH:MM::SS	If selected use HH:MM::SS format in the Time Control window. If cleared, display time in seconds.
Show Exercise Time	Enable or disable the Exercise Time display in the Time Control window.
Geometry Loading tab	
Geometry Optimization	Enable or disable optimization parameters. Disabling them significantly affects performance.
Geometry Loading/Memory Management	Options for memory management. For details, please contact support@mak.com.

2.6. Using Multiple 3D Windows

DI-Guy Scenario can display up to nine secondary 3D Views in addition to the primary 3D window. Each secondary 3D View provides its own view into the scenario, simultaneously with the main view. Each 3D View has its own camera which can be set to any of the camera settings.

The location and display status of secondary 3D Views is saved as part of the scenario.

➤ To open a secondary window, choose **Window** → **Secondary 3D View** *n*, where *n* is 0 through 8.

2.7. Common GUI Controls

Most user-level management of character and scenario objects is done on the pages in the Scenario Objects window. These pages have a set of common controls. Rather than repeat some of the common procedures for every page, this section describes generic procedures that you can apply to almost every page.

Common procedures include:

- Adding objects.
- Deleting objects.
- Renaming objects.
- Copying objects.
- Viewing the selected object.
- Exporting objects to libraries.
- Importing objects from libraries.

Figure 2-9 identifies the buttons for the common procedures.

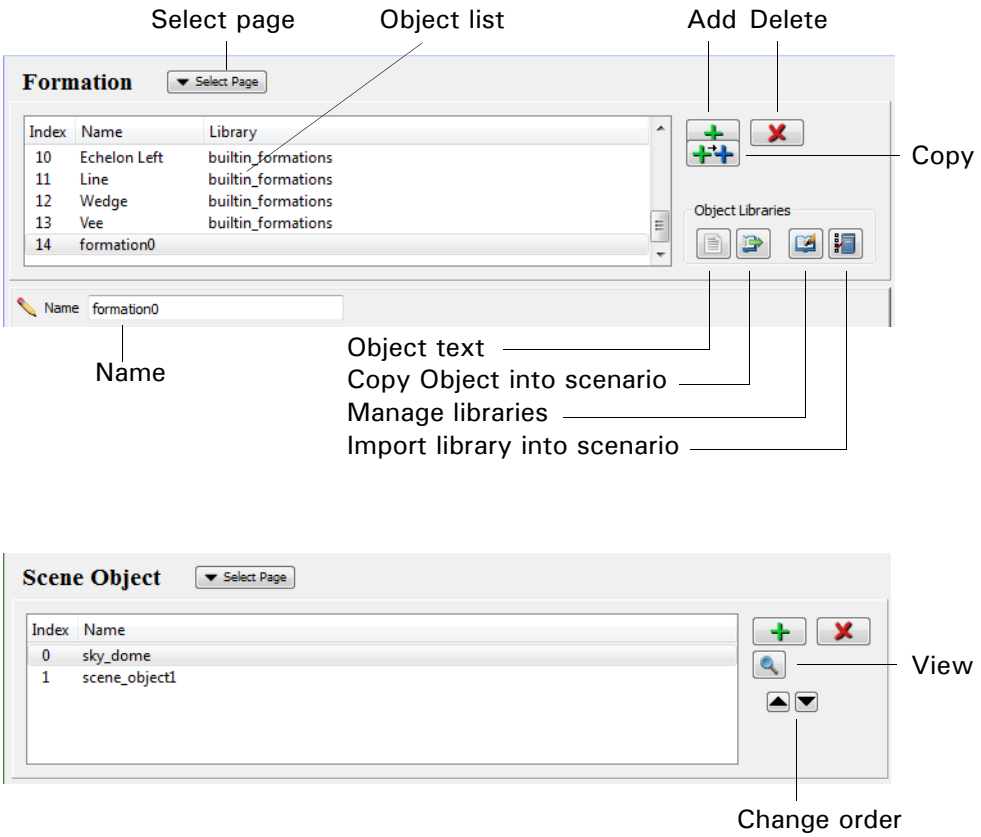


Figure 2-9. Common GUI controls

2.7.1. Adding a New Object

You can add a new instance of an object on its page. In some cases you must go to the 3D View to complete the creation process. See the individual object procedures for further instructions.

To add a new object:

1. Select the page for the object you want to add.
2. Click the Add button (+). A new entry is added to the list.
3. Optionally, type a new name for the objects in the Name box.

2.7.2. Deleting Objects

To delete an object:

1. Select the object you want to delete.
2. Click the Delete button (x).

2.7.3. Copying Objects

Sometimes you want to create a new object that is very similar to an existing one. It may be quicker to copy the existing object and then change it than it would be to create a new object and completely configure it.

To copy an object.

1. Select the object you want to copy.
2. Click the Copy button (++). A new object is added to the list with the default name originalObjectName_copy0.
3. Type a new name for the object and edit it as desired.

2.7.4. Teleporting the Camera to a Specific Object

You can make the camera jump to a specific object.

To teleport the camera to an object:

1. Select the object you want to view.
2. Click the teleport button (). The camera moves to the object.

2.7.5. Changing the Index of an Object

Objects are listed in the object list window by index. You can change the order of the objects. Changing the order changes an object's index.

To change the index of an object:

1. Select the object whose index you want to change.
2. Click the Up or Down arrow (▲▼) to move the object up or down in the list.

2.7.6. Sharing Libraries Among Scenarios

DI-Guy object libraries let you share scenario objects among scenarios. The object library buttons have the following functions:

- Display Object Text ( ). Shows what the internal description of the current profile looks like. This is the code that is saved with the scenario file.
- Copy Object Into Scenario ( ). Moves an object from an external library such as "builtin_crowd_profiles" to a local copy.
- Manage Object libraries ( ). Copies objects from local to libraries or between libraries.
- Select Object Libraries to include in scenario ( ). Lets you browse for new libraries to include. Useful when you want to share libraries across projects.

Displaying Object Text

The text displayed for an object depends on the object type. Figure 2-10 shows object text for a particle system.

To display object text:

1. Select the object whose text you want to display.
2. Click the Display Object Text button (). Text is displayed in the Log window. For some object types, an additional window opens (Figure 2-10).

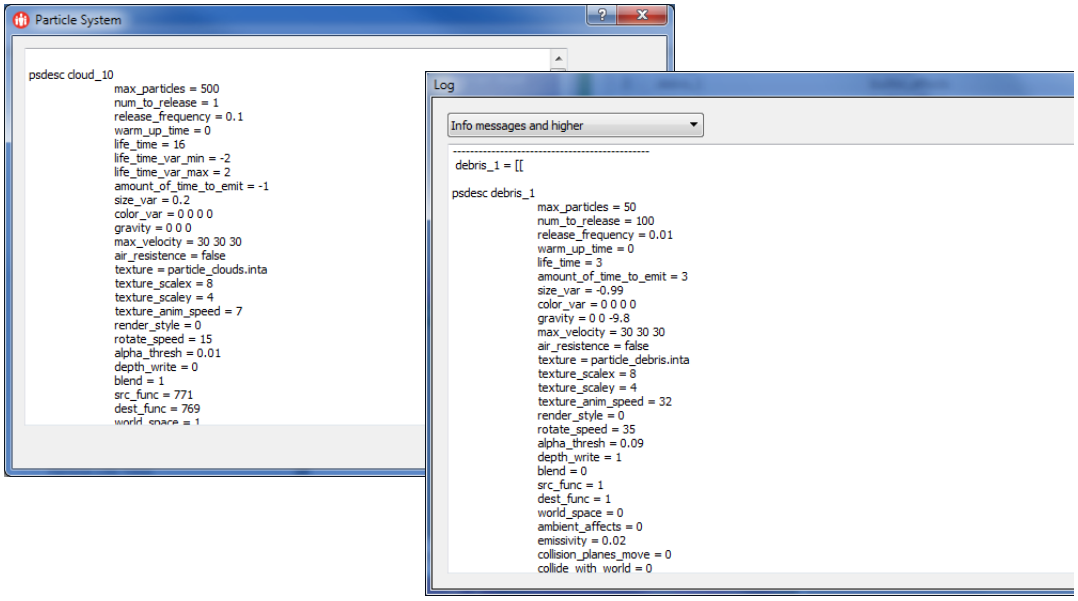


Figure 2-10. Object text

3. If text is displayed in a specific window, after viewing the text, click OK.

Copying an Object from a Library into a Scenario

Some objects are available to a scenario because they are part of a library. For example, on the Scenario Objects:Particle System page for a default scenario, most of the particle systems are listed as being part of the builtin_effects library (Figure 2-11). You can copy an object from a library into a scenario and edit it for use specific to the scenario.

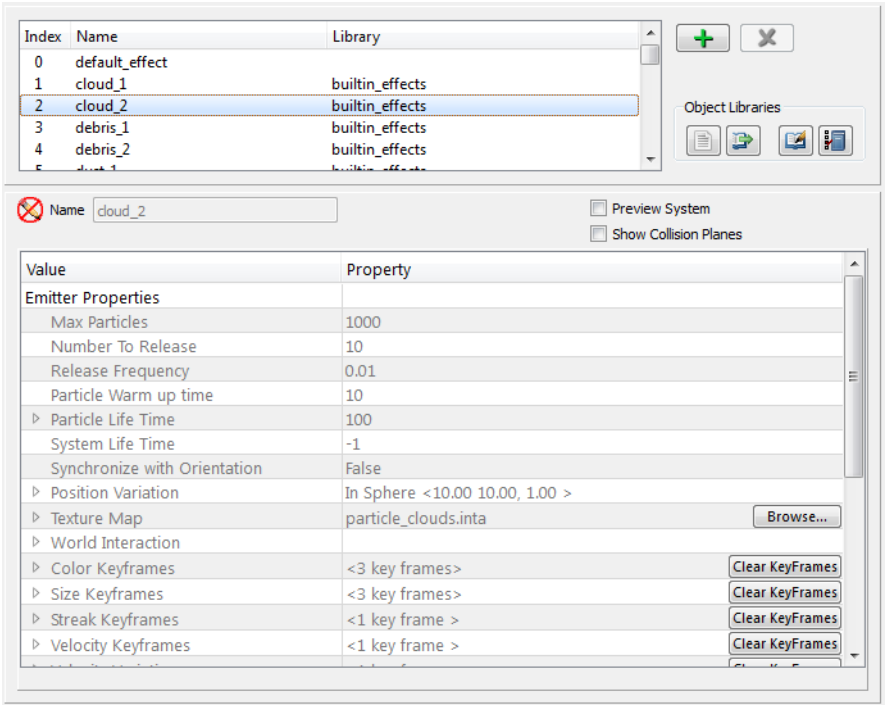


Figure 2-11. Particle systems from a library

To copy an object from a library to a scenario:

1. Select the object you want to copy.
2. Click the Copy Object Into Scenario button ( ). The object is copied into the scenario. It is no longer listed as being part of a library. Compare Figure 2-12 to Figure 2-11.

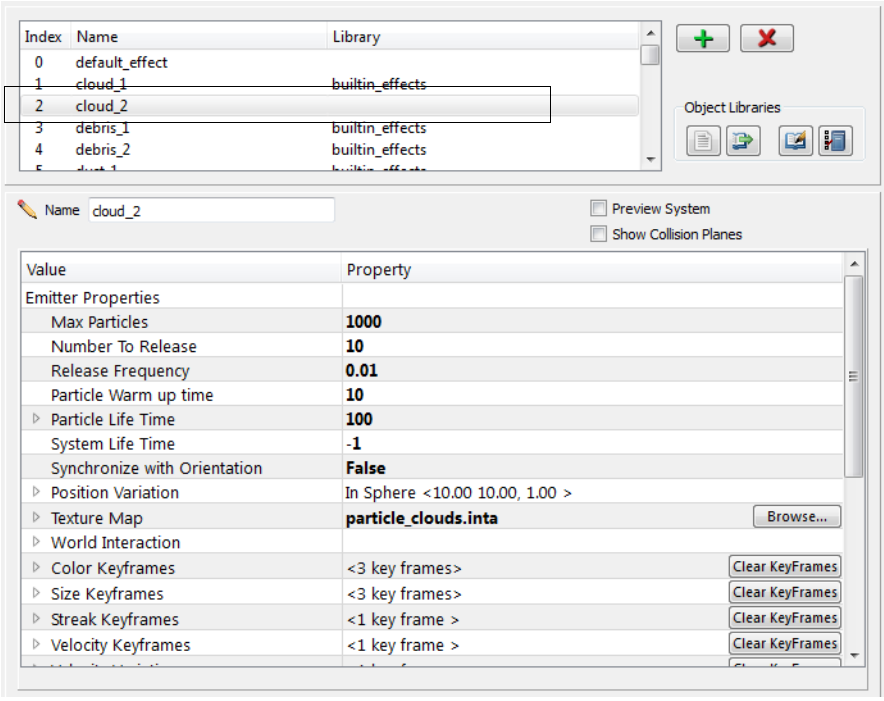


Figure 2-12. Object copied into scenario

Managing Object Libraries

The Object Library Manager lets you copy entire libraries or selected objects in libraries into new or existing libraries. You can merge objects into a library or completely replace its contents. You can create new libraries. You cannot rename or delete built-in libraries. You can rename or delete libraries that you create.

To manage object libraries:

1. Click the Manage Object libraries button (). The Object Library Manager opens (Figure 2-13).

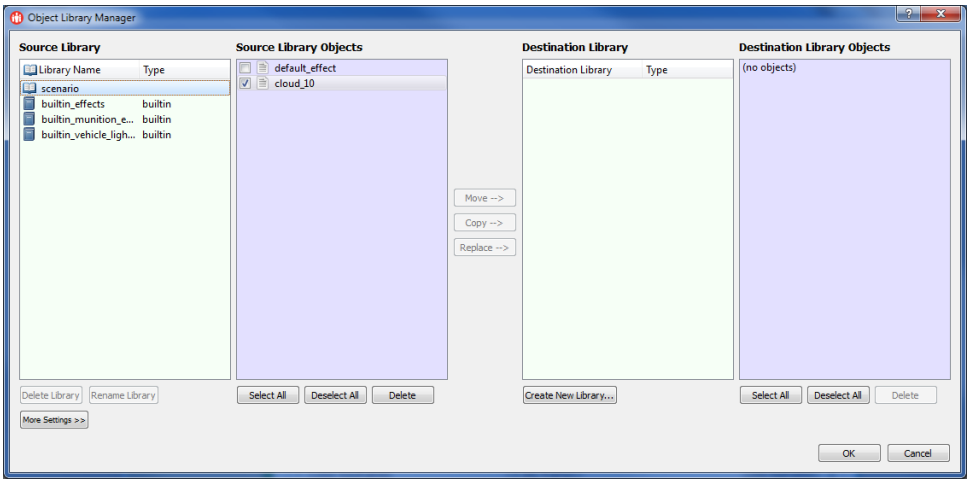


Figure 2-13. Object Library Manager

2. In the Source Library list, select a source library. The Source Library Objects list displays the objects in the selected library.
3. In the Source Library Objects list, select the objects that you want to copy to another library.
4. If you want to copy the objects to a new library, click Create New Library and type a name for the new library.

> Newly created libraries get added to the Source Library list as well as the Destination Library list. Once you add objects to the new library, it could be a source for copying objects to some other library.

5. In the Destination Library list, select the destination library for these objects.
6. To copy the selected objects into the selected destination library, click Copy–>. The selected objects are added to the destination library.
7. To replace the contents of the destination library with the selected source objects, click Replace–>.

8. Click OK.

Importing Libraries into a Scenario

A new scenario includes some object libraries by default, but may not include all available libraries. For example, a new scenario includes the builtin_effects library and the builtin_munition_effects library, but it does not include the builtin_vehicle_lights_effects library. If you want the objects in a library to use in a scenario, you can import the library. This is particularly useful if you build libraries of custom effects that are not provided with DI-Guy Scenario. You can save them and share them amongst your scenarios and with other DI-Guy Scenario users.

To import a library into a scenario:

1. Click the Select Object Libraries to Include in Scenario button (). The Object Library Selector dialog box opens (Figure 2-14). It shows which libraries are already part of the scenario.

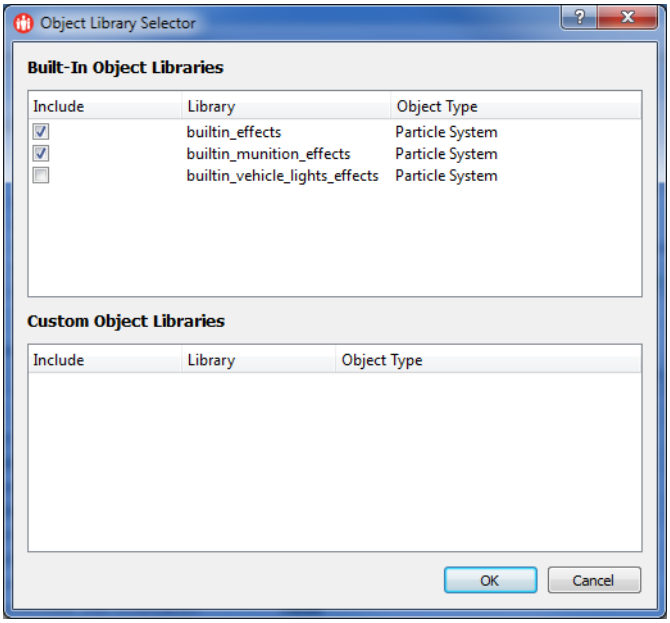


Figure 2-14. Object Library Selector

2. Select the libraries that you want to import.
3. Click OK. Verify that the objects from the library are listed in the list window for the page you are editing.

2.8. Hiding and Displaying Objects

You can hide many of the objects that DI-Guy Scenario simulates, such as characters, paths, regions, and particle systems. In most cases you can choose one of the following options:

- None. All objects of the type are visible.
- Current. The selected object is hidden.
- All. All objects of the type are hidden.

To hide or display objects:

1. Select the View Settings:Visibility page (Figure 2-15).

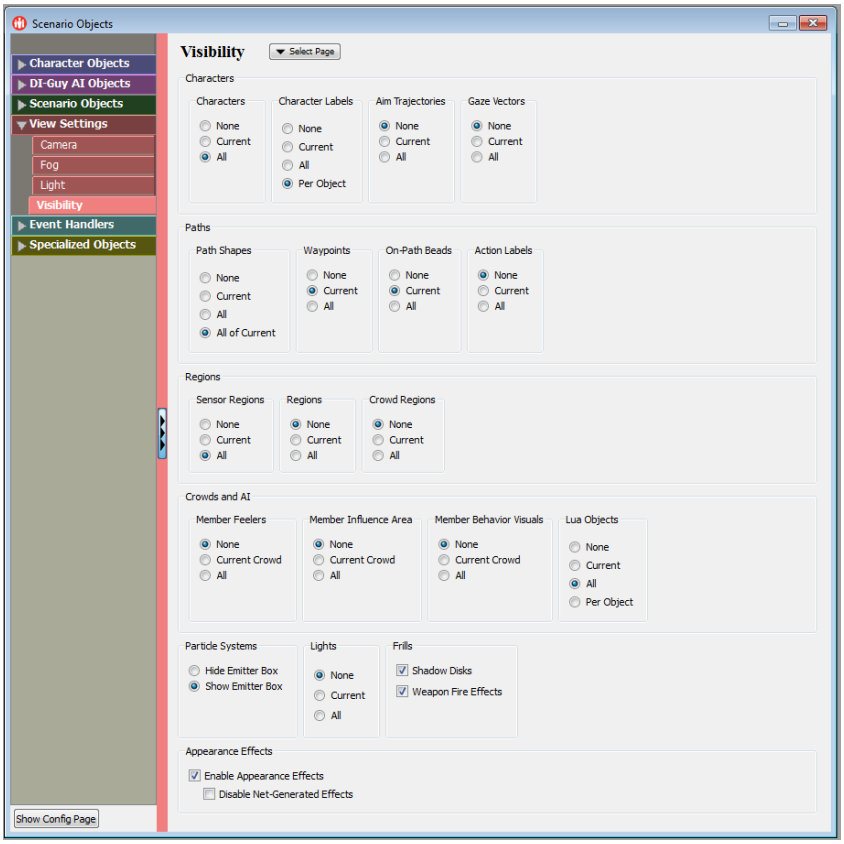


Figure 2-15. Visibility page

2. If you want to hide a particular object, select it.
3. In the group box for the object you want to display or hide, select the appropriate option.

3. DI-Guy Scenario Tutorials

This chapter uses tutorials to introduce you to many DI-Guy Scenario features.

A Simple Scenario 3-3
- Start DI-Guy Scenario 3-3
- Move the Camera 3-5
- Add People to the Scenario 3-6
- Play the Scenario 3-8
- Save the Scenario 3-8

Selecting Characters and Changing their Appearance 3-8
- Choose a Character Type 3-9
- Choose an Appearance 3-10
- Choose a Hand Item 3-10
- Choose an Action 3-10
- Add the Character 3-11
- Other Types of Characters 3-11
- Change a Character's Appearance 3-12
- Selecting a Character 3-13

Creating Paths for Characters 3-14
- Selecting a Path 3-14
- Adjust a Path 3-15
- Extend a Path 3-17
- Insert Waypoints 3-18

Actions and Behaviors 3-18
- Action Beads 3-19
- Adding Actions 3-20
- Dragging Actions 3-21
- Adding Stationary Actions 3-22

Change an Action 3-23
Decision Logic 3-23
Sensor Regions 3-24
Creating Decision Logic 3-25
Cameras 3-30
Saved Camera Settings 3-31
Teleport the Camera 3-32
Tether the Camera to a Character 3-33
Tracking a Character 3-34
Camera Point-of-View (POV) 3-34
Load Your Own Terrain 3-35
Change the Environment 3-37

3.1. A Simple Scenario

The easiest way to learn DI-Guy Scenario is to use it. The tutorials in this chapter cover the basic features and functions, with a small dose of theory of operation. In this tutorial, you will:

- Start a new scenario using a default terrain.
- Learn how to move the camera.
- Add characters to the scenario.
- Run the scenario.
- Save the scenario.

3.1.1. Start DI-Guy Scenario

➤ To start DI-Guy Scenario, double-click the DI-Guy Scenario desktop icon or on the Start menu, choose **All Programs** → **DI-Guy Applications 13** → **DI-Guy Scenario**.

When DI-Guy Scenario starts, it loads a new scenario with a default terrain. The terrain is displayed in the Primary 3D View window (Figure 3-1). (You can open up to nine secondary windows. In this manual we refer to the Primary 3D View window as just the 3D View.)

DI-Guy Scenario may also open some other windows and dialog boxes. For this tutorial, you will need the Time Control window and the Input Mode window. If either of them is not visible, select them on the Window menu. When a window is open, there is a check mark next to it on the Window menu.

Close any other windows that might be open.

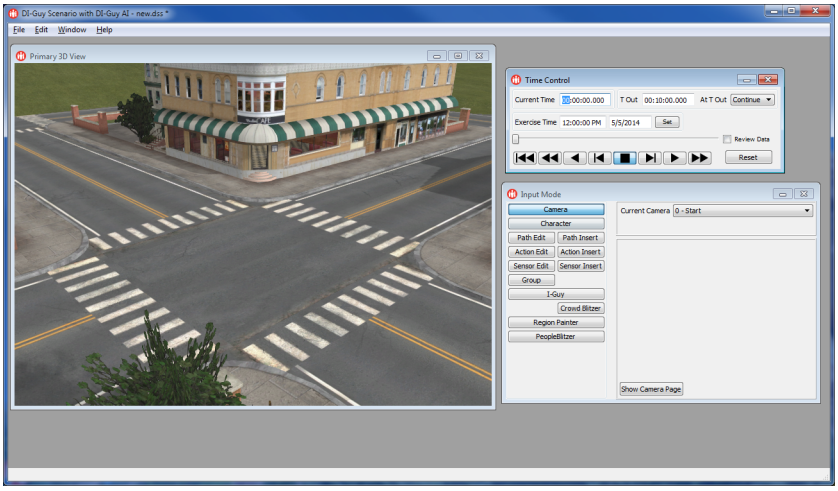


Figure 3-1. DI-Guy Scenario window at startup

Terrain Models

Terrain models include the ground, roads, trees, buildings, and furnishings that make up a space. Typically the first step in creating a scenario is to load the terrain to be populated. For this tutorial, we will use the default terrain model, the DI-Guy Scenario Stage (Figure 3-2). The Stage is a simple model that has a small-town intersection with a park and a warehouse and a mid-eastern walled town. When DI-Guy Scenario starts, the Stage is displayed in the 3D View.

Figure 3-2. DI-Guy default terrain

3.1.2. Move the Camera

The location from which you view the terrain in the 3D View is called the camera. The camera has a location and an orientation in the 3D world. You can move the camera to look at the world from any perspective that you want. To move the camera you must be in Camera mode, or use a hot key to escape whatever other mode you are using.

1. In the Input Mode window, click Camera.
2. Move the mouse cursor into the 3D View. Notice that the cursor shows some eyeballs and the text F+B. The eyeballs indicate that you are in Camera mode. F+B means you can move the camera forward and backward.
3. Press the left mouse button. The camera moves forward through the town.
4. Press the right mouse button. The camera moves backward through the town.
5. As you hold down a mouse button, move the mouse left and right to change the direction of movement.
6. As you hold down a mouse button, move the mouse forward and backward to change the orientation of the camera.
7. Hold down both the left and right mouse buttons at the same time and move the mouse to rotate around the current location.

If you get lost, press **0** to return to the original camera location. DI-Guy Scenario can map up to 10 different camera settings to the number keys.

If the camera moves too slowly or too quickly, type **–** or **+** to adjust the speed. Each key press changes the speed. You do not need to hold down the speed keys.

You can also move the camera left and right and up and down, as follows:

- To change camera movement to left and right, press **s**.
- To change camera movement to up and down, press **d**.
- To return to forward and backward movement, press **f**.

3.1.3. Add People to the Scenario

We will put a person in the scene and have him walk from one place to another. Using the PeopleBlitzer you can add a person with just one click of the mouse.

1. In the Input Mode window, click PeopleBlitzer. (We will use the default character, soldier_07 and its default attributes.)
2. Move the cursor over the 3D View. The cursor has changed to show you are in creation mode.
3. Click on one of the street corners. A character gets placed. It has a red line under it and a rotating yellow object (Figure 3-3). In the Input Mode window, the selection is now Path Edit and the cursor changes to show path edit mode.
4. Press **`Alt`**. Notice that the cursor changes to Camera mode. When you release the key, it returns to Path Edit mode.

When you clicked to place the character, DI-Guy Scenario did the following things:

- Created a character using character type `soldier_07`.
- Created a path that has one waypoint (the red line).
- Created an action bead (the yellow object) with a default action (`stand_ready`).

If you wanted to edit the path so that the character can walk someplace, you could. That is why the Input Mode window went into Path Edit mode.

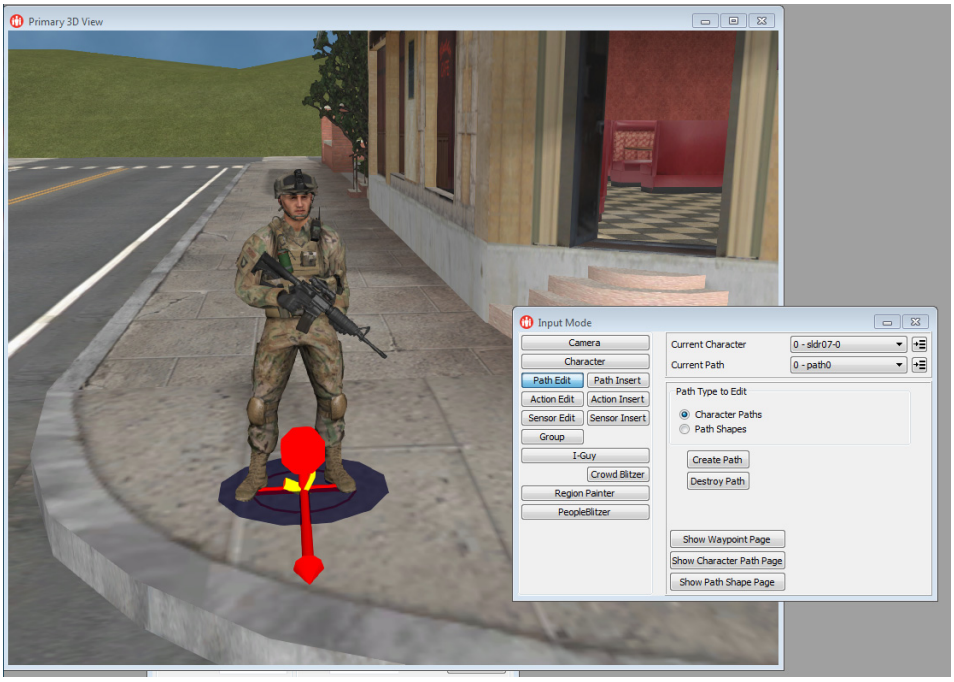


Figure 3-3. New character

You can create a new character and give it a path at the same time.

1. In the Input Mode window, click PeopleBlitzer.
2. Click at the start of the crosswalk and while holding down the left mouse button, drag a line across the street to the other corner. A character gets placed (Figure 3-4). It has a path, highlighted in white, a waypoint, in green, and an end point, the active waypoint, in red. Click-and-drag is useful because it allows you to specify a location, direction and distance, all in one operation.
3. Place some additional characters in the scene. If a character is created without a path, you can add one later.

Figure 3-4. New character and path

3.1.4. Play the Scenario

Now that you have placed people in the scenario, let's see them do something.

1. Find the Time Control window.
2. Click the Play button (▶). The characters with paths will walk along them.
3. When the characters have finished walking, click the Stop button (■).
4. Click Reset. The scenario returns to the starting point.
5. Move the scenario forward frame by frame by clicking the Frame button (▶|).
6. Run the scenario at four times the normal speed using Fast Forward (▶▶).
7. Move the camera around while the characters move.
8. Click the Stop button.

As the controls in the Time Control window imply, you can run a scenario backwards, but this requires some configuration that we will not cover in this tutorial.

3.1.5. Save the Scenario

You can save your work to a DI-Guy Scenario scenario file (*.dss*).

To save the scenario:

1. Choose **File** → **Save**. The Save As dialog box opens.
2. Type a name for the scenario.
3. Click Save.

You cannot save files if you are running DI-Guy Scenario in Demo Mode without a license.

3.2. Selecting Characters and Changing their Appearance

DI-Guy Scenario includes many characters, which have a variety of appearances and behaviors. When you used the PeopleBlitzer earlier, you added a character with a default character type, appearance, and action. Now we will select character type, appearance, and action using the Input Mode window and add a new person with desired characteristics.

3.2.1. Choose a Character Type

The people in DI-Guy Scenario are organized by character type and appearance. Character type determines which set of actions in the motion library are used for the person, and therefore it determines what the person can do. For instance, people of type female_pedestrian can stand, walk, jog, stroll, powerwalk, and run. People with character type soldier_07 can stand, walk, jog, and run, but they can also crawl, walk_lo (sneak), and do things with their weapons.

1. In the Input Mode window, click PeopleBlitzer. The right side of the dialog box displays a set of drop-down lists (Figure 3-5).

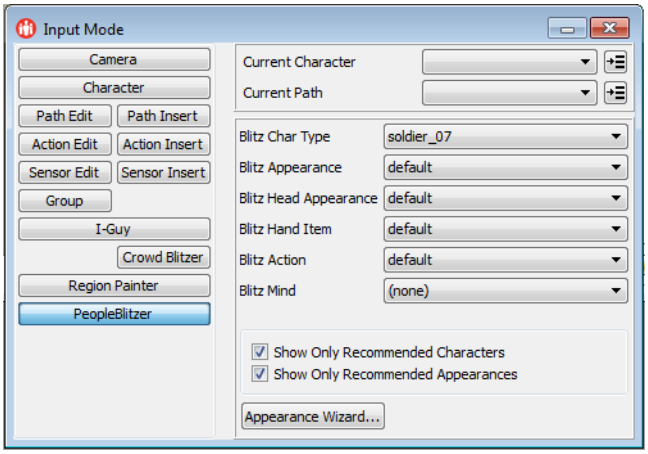


Figure 3-5. Input Mode window, PeopleBlitz mode

2. In the Blitz Char Type list, select `mped_07` (male pedestrian).

- The mped_07, fped_07, and soldier_07 are the best character types to start with because of their many appearance options.
- Appearances with the "sk_" prefix use superior deformable mesh graphics (skinned characters), while other appearances use older segmented polyhedra.
- You can survey all of the characters and their actions with the DI-Guy Character Viewer.

3.2.2. Choose an Appearance

Appearances specify what a DI-Guy person looks like. Appearances include body shape, hair color, and clothing. Appearances can also include certain accessories, such as goggles, backpacks, canteen, and so on. For each character type, there is a defined set of appearances.

➤ In the Blitz Appearance list, select `sk_male_suit_1`. (Appearances that start with sk indicate that it is a skinned character. Skinned characters look more natural than older, segmented characters.))

3.2.3. Choose a Hand Item

Some characters can carry objects, such as cell phones or weapons.

➤ In the Blitz Hand Item list, select `cell`.

3.2.4. Choose an Action

When you create a character, you can give it an action. The default action is usually just to stand. This was the case with the soldiers you created in the first tutorial.

➤ In the Blitz Action list, select `walk_talk_on_cell_phone`.

3.2.5. Add the Character

1. In the 3D View, click to place the new character and drag a path. The character is a male in a suit and is holding a cell phone.
2. Run the scenario. The character walks along the path talking on the cell phone (Figure 3-6).

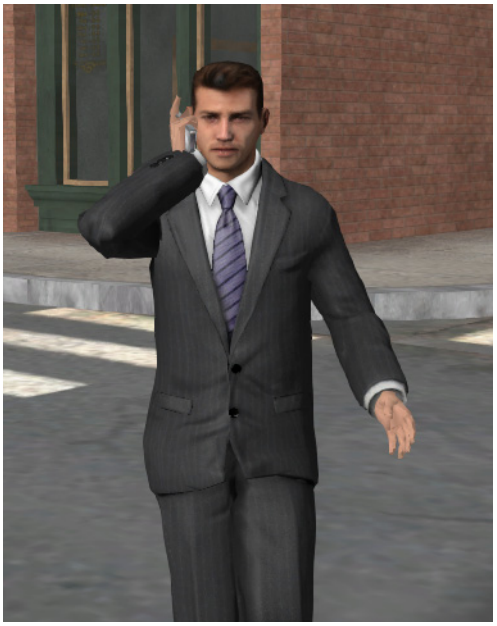


Figure 3-6. New character with cell phone

3.2.6. Other Types of Characters

Besides the many human characters that DI-Guy supports, you can create some non-human "characters" – vehicles, props, and effects.

Vehicles

The vehicle character type includes cars, trucks, tanks, boats, and aircraft that move about the scene, but do not have any moving joints. vehicle_wheels characters have extra joints that allow the wheels to steer and rotate along the path. Similarly, vehicle_technical and vehicle_turret are vehicles with weapons mounted on them that respond to aim commands.

Vehicles can travel on paths. Vehicles do not have actions, but you can specify different travel speeds by setting the action attribute. DI-Guy Scenario comes with a library of vehicles.

Props

Props are static objects you can put into your scenario to enrich the scene. Examples of props are fences, furniture, food, and houses. You can add props to the scenario using the People Blitzer. Once a prop is placed in the scenario, you can move it to another location by dragging its waypoint. You can reorient a prop by reorienting its waypoint. DI-Guy Scenario comes with a prop library.

Effects

Effects are particle-based graphics for creating explosions, fire, smoke, dust trails, weather effects, and so on. They use character_type "effect". For information about adding effects to a scenario, please see "Adding Special Effects (Particle Systems)," on page 7-5.

3.2.7. Change a Character's Appearance

Once a character is in the scene, you can read or modify its appearance using the Input Mode window.

1. In the Input Mode window, click Character.
2. In the 3D View window, click a character. The Input Mode window lists the character as its Current Character. It displays its character type, appearance, and hand item (if applicable) (Figure 3-7).

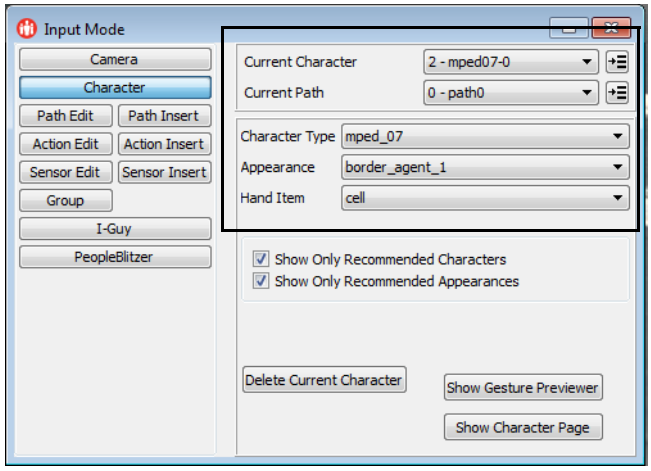


Figure 3-7. Character information

3. In the Appearance list, select a different appearance, such as `firefighter`. The character changes to the new appearance.
4. If the character has a hand item, select a different item in the Hand Item list.
5. See what happens if you change the character type.

3.2.8. Selecting a Character

In the previous section, you selected a character to change its appearance. DI-Guy Scenario has several ways to select characters. Try each of the following selection procedures:

- In the 3D View window, click a character. You can do this in Character mode and in Path Edit mode.

  Since paths are specific to characters, in Path Edit mode, you can select the character whose path you want to edit. Notice that the selected character is listed in the Current Character list, but there is no information about the character type or appearance. Since you are in Path Edit mode, information about the character's path is displayed.

- In the Input Mode window, select a character from the Current Character list. It becomes the current character. The input mode does not change. This option is available in Character mode, Path Edit mode, Action Edit mode, and PeopleBlitzer mode.
- Select a character in the Scenario Objects window:

  a. Choose **Window** → **Scenario Objects**. The Scenario Objects window opens.

  b. Click the Character Objects tab. The list expands to show tabs associated with characters.

  c. Click the Character page. The Character page lists all of the characters in the scenario (Figure 3-8).

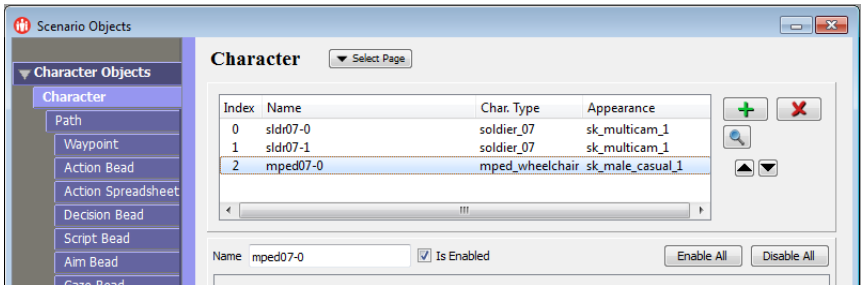


Figure 3-8. Character page

  d. In the list of characters, click a character. It is selected in the 3D View.

For more information about creating characters, please see Chapter 4, *Creating and Editing Characters*.

3.3. Creating Paths for Characters

One way DI-Guy Scenario characters can travel through a scenario is by following a path. The shape of each path is controlled by a set of waypoints. Waypoints are shown as green or red post-like structures. At the bottom of the waypoint is a slope-adjuster bar that determines the slope of the path where it passes through the waypoint. You can change the location and shape of a path by repositioning the waypoint, reorienting the waypoint, or changing the slope adjuster.

DI-Guy Scenario has two kinds of paths, character paths and path shapes. Character paths are specific to the characters for which they were created. Path shapes are scenario-level objects that are accessible from decision and script events. This tutorial covers character paths.

3.3.1. Selecting a Path

To interact with a path or any of its waypoints or actions, the path must be selected. Selected paths are displayed with a white line; unselected paths are either blue or not displayed.

To select a path:

1. In the Input Mode window, click Path Edit.
2. Click the person whose path you want to edit.

3.3.2. Adjust a Path

Let's change the path for a character by repositioning the waypoints.

1. If you have the previous tutorial open, select the character that has a path that crosses the street at the crosswalk. Otherwise, create a new character that has a path from one corner to the other.
2. In the Input Mode window, click Path Edit.
3. Using the left mouse button, click and drag the end waypoint to a different corner.
4. Now move the other end of the path so that the soldier starts in the crosswalk for the new direction of movement (Figure 3-9).

Figure 3-9. Editing a path

Each waypoint has a slope adjuster. You can change the curvature of a path with the slope adjusters.

➤ To change the curvature of a path with a slope adjuster, click the end of the slope adjuster while dragging and rotating the mouse.

Changing the length of the slope adjuster affects the degree of curvature. Changing the angle of the slope adjuster affects the angle of curvature at the waypoint. Figure 3-10 shows the path in the crosswalk with an S-curve instead of a straight path.

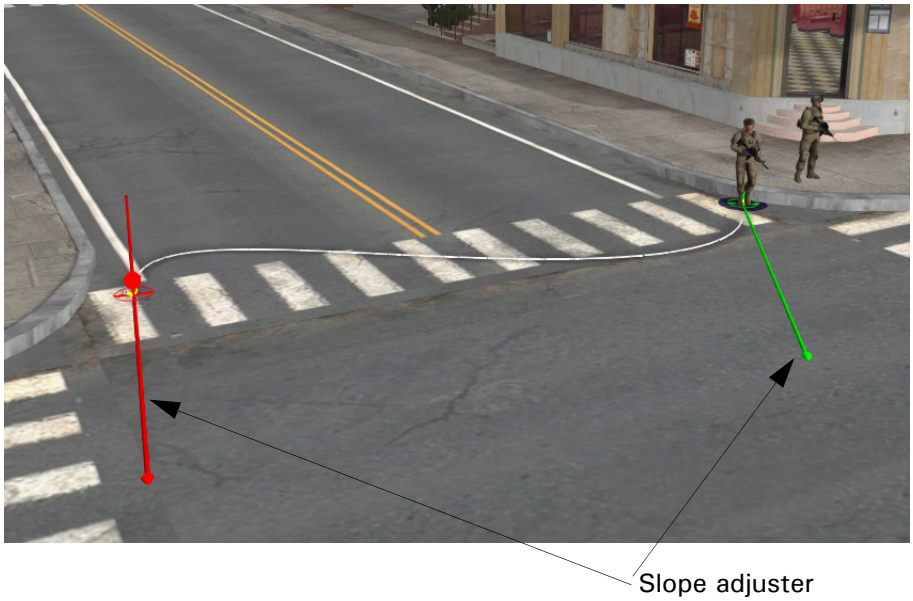


Figure 3-10. Using slope adjuster on a path

It is easiest to grab the slope-adjuster at its endpoints. You may have to click once on a waypoint to select it, in order to make its slope-adjuster active. (The waypoint and its slope-adjuster turn red when the waypoint is selected.) If you have trouble manipulating a waypoint, zoom in on it by moving the camera and try again.

3.3.3. Extend a Path

Paths can have more interesting shapes when they have more waypoints. You can extend an existing path by adding waypoints.

1. In the Input Mode window, select Path Edit.
2. Select the character with the path in the crosswalk.
3. Click the last waypoint on the current path. It turns red.
4. In the Input Mode window, select Path Insert.
5. Click on each corner of the intersection until the path returns to the starting point (Figure 3-11).

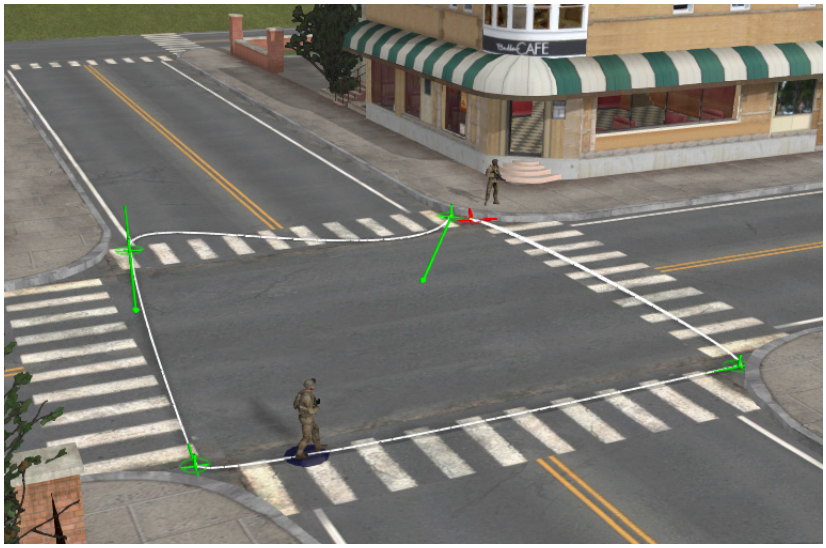


Figure 3-11. Extended path

To exit Path Insert mode, select a different mode in the Input Mode window, such as Camera.

6. Run the scenario. The soldier will walk across all four crosswalks.
7. Stop the scenario and rewind it.

3.3.4. Insert Waypoints

You can add waypoints to the middle of a path. The waypoints that define a path have an order that progresses from the beginning of the path to the end. When you insert a waypoint it is placed in the sequence right after the selected waypoint.

1. In the Input Mode window, select Path Edit.
2. Select the character with the path in the crosswalks.
3. Click one of the intermediate waypoints on the path. It turns red.
4. In the Input Mode window, select Path Insert. The path segment between the waypoint you selected and the next waypoint on the path becomes active.
5. Click along the existing path in a couple of places. Each time you click a waypoint is added and the editable path segment changes.

For more information about paths and waypoints, please see Chapter 5, *Paths and Waypoints*.

3.4. Actions and Behaviors

People in DI-Guy Scenario are always performing actions, even when they are just standing. For instance, walking, standing, and dying are all actions. Typically a person acts out a whole sequence of actions during a complex scenario. A person might start walking, then have a conversation, then stand in one spot, then stroll at a slow pace. DI-Guy Scenario makes it easy for a person to perform a series of actions smoothly and coordinate their actions with their location in the terrain model and with the locations of other people.

Some actions cause the person to travel along a path, such as walking, jogging, and crawling. These are called traveling actions. Other actions do not cause the person to travel along the path, such as standing, kneeling, or gesturing with the hands. These are called stationary actions. There are some differences in how DI-Guy Scenario handles traveling and stationary actions.

3.4.1. Action Beads

Actions can be specified and controlled in DI-Guy Scenario by using action beads. Action beads are displayed in the 3D View as yellow, gear-like elements located along the paths. Action beads are located at the start of each path and at points along the paths, wherever a person changes from one action to a new one. There is also a special action bead, the end bead, located at the end of a person's last action. The end bead is visually represented by a small red stop sign (Figure 3-12).

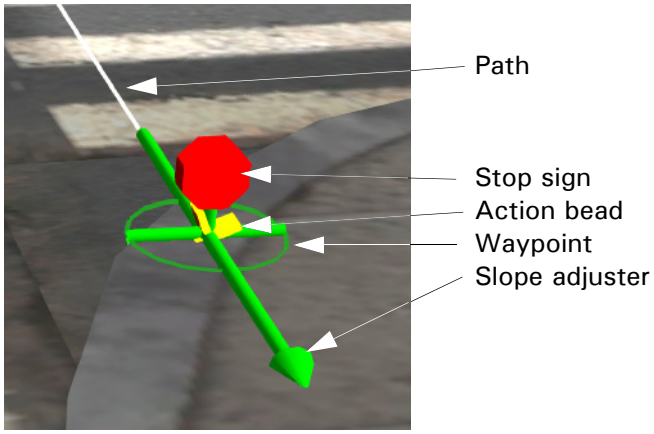


Figure 3-12. Stop sign and action bead

3.4.2. Adding Actions

Typically, you will want each person to perform several actions, one after another.

1. Create a new scenario and add an mped07 character with a path. Set the Blitz Action to `walk`. Give it a cell phone like you did with the border guard in the previous tutorial.
2. Play the scenario to see that with the default settings, the character walks along the path to the end.
3. Reset the scenario.
4. In the Input Mode window, click Action Insert. The window redraws and adds a list called Insert Action.
5. In the Insert Action list, select `walk_talk_on_cell_phone`.
6. Click on the path a few feet from the starting point. An action bead is inserted where you clicked (Figure 3-12).

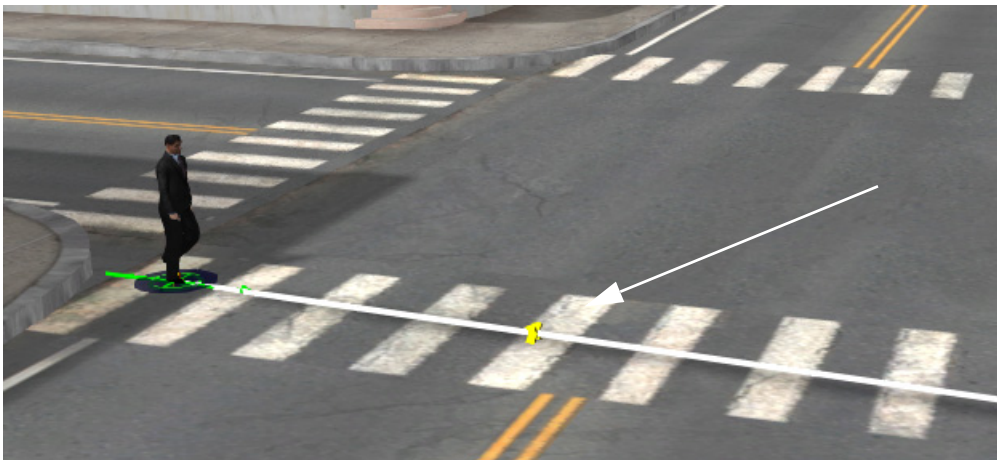


Figure 3-13. Action bead on path

7. Add a second action bead between the first one and the end of the path. For this action bead, use the action `walk_text`.
8. Play the scenario. The character walks along the path to the first action bead. It stops for a second, raises the cell phone to its ear, then continues to walk along the path talking on the cell phone. When it reaches the second action bead, it pauses again, then walks while texting.
9. Reset the scenario.

3.4.3. Dragging Actions

You can change when and where an action starts by repositioning the action bead on its path.

1. In the Input Mode window, click Action Edit.
2. Click and drag the first action bead you created (`walk_talk_on_cell_phone`) along the path to reposition it. The action bead can only move along the path. It cannot move past the second action bead.

You can only drag an action bead along the path as far as a neighboring action bead. When you bump into a neighboring action bead, the two action beads stack up. From time to time, you may have stacks of action beads that are several beads tall. You can separate the action beads of a stack by pointing to one side or other side of the stack when click-dragging the action beads.

When manipulating stacked action beads, point at the path below the action bead, not at the action bead itself.

3.4.4. Adding Stationary Actions

Examples of stationary actions are standing, kneeling, or sitting. The key distinguishing feature of stationary actions is that the person does not travel along the path during their execution. Stationary and traveling actions are often used in the same path. For example, a character will walk 5 meters, kneel, and then resume walking.

When you specify a stationary action, its key parameter is duration (for example, stand for 5 seconds) rather than a distance (for example, walk 5 meters). The Character Objects:Action Spreadsheet page, gives you a numeric means of specifying times as well as distances. For information about the Action Spreadsheet, please see "The Action Spreadsheet," on page 6-10.

1. Select the character you have been using for the action tutorial.
2. Click Action Insert.
3. In the Insert Action list, select `stand_hands_up`.
4. Click on the path to place the action.

   Two new action beads are placed on the path in a stack. One action bead specifies the stationary action you inserted and the second action bead resumes the traveling action that was interrupted. Because stationary actions keep a person in one place along the path, their action beads will typically be found stacked up with at least one other action bead.

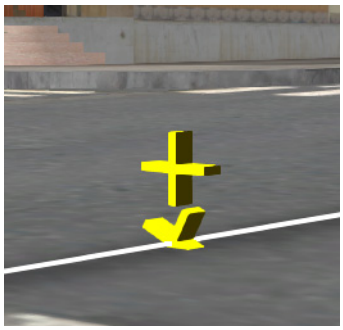


Figure 3-14. Stationary action bead

5. Play the scenario. The character walks along the path, stops and raises its hands, then continues walking.

3.4.5. Change an Action

Once an action is specified, you can change it to another action.

1. In the Input Mode window, click Action Edit mode.
2. Select the stationary action bead you created. You know it is selected because it rotates. It is also listed as the Current Action in the Input Mode window.
3. In the Input Mode window, in the Current Action list, select a different action, such as `stand_come_here`.
4. Reset the scenario.
5. Play the scenario.

3.5. Decision Logic

DI-Guy Scenario has a decision-making capability that can be attached to any character. It uses sensor regions, decision beads, script beads, and signals. In this tutorial we show you how to create a sensor region and some decision logic that uses it. For more information about decision logic, scripts, and signals, please see Chapter 6, *Character-Level Actions* and Chapter 7, *Scenario Objects*.

3.5.1. Sensor Regions

A sensor region is an axis-aligned 3D box that is sensitive to characters. When a character enters a sensor region, the sensor region detects its presence and provides the information to the decision logic system.

1. Create a new scenario.
2. In the Input Mode window, click Sensor Insert.
3. In the 3D View, click on the corner in front of the cafe and drag across the street to the warehouse so that you create a sensor region that covers the crosswalk. As you drag, DI-Guy Scenario displays a box that expands to the size you drag it (Figure 3-15).

Figure 3-15. Sensor region

Once you have created a Sensor Region, you can change its size or position by editing it. Sensor regions are listed on the Scenario Objects:Sensor Region page. You can give them names and set a variety of parameters. For more details, please see "Sensor Regions," on page 7-12.

3.5.2. Creating Decision Logic

DI-Guy Scenario lets you create reactive behaviors for people and assign them to events in the scenario. The Character Objects:Decision Bead page makes it easy to create decision logic without any programming. Decision logic is enabled through decision beads.

Decision beads are specified on a per-character and per-path basis. When you create a new decision bead it is associated with the currently selected character and path. You can also create decision logic at the scenario level. For details, please see "Creating Scenario-Level Decision Logic," on page 9-3.

You can also create decision logic by writing scripts using the Lua programming language. For details, please see "Writing Script Beads," on page 6-20 and "Writing a Scenario-Level Script," on page 9-7.

1. Using the scenario you created in the previous section (with a sensor region), add an mped_07 character with a walk action and a path that runs through the sensor region.
2. Add a character of type `vehicle`, appearance `van`, with a path that runs through the intersection (Figure 3-16).

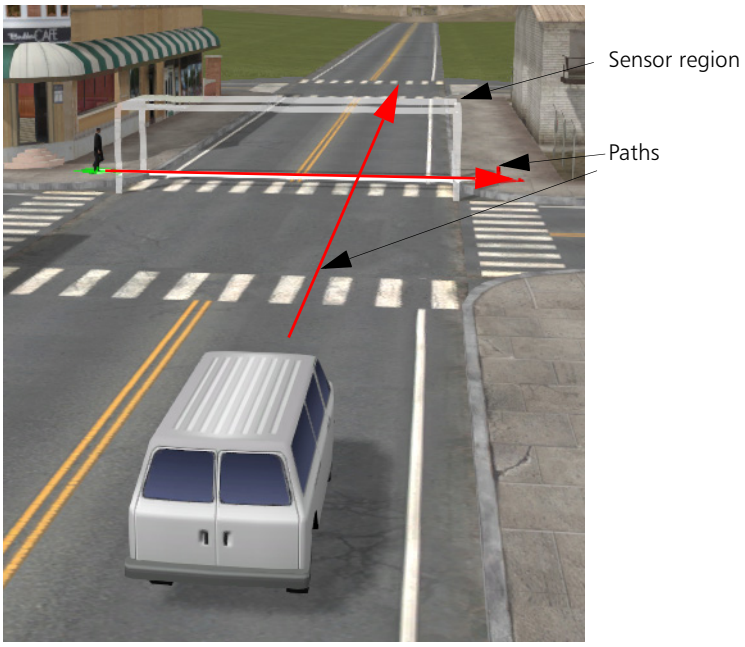


Figure 3-16. Sensor region and characters.

3. Play the scenario. The pedestrian crosses the street and the van drives through the intersection. Depending on exactly where you placed the characters, the van may or may not pose a threat to the pedestrian, but we want to be sure that it will not drive through the intersection if a pedestrain is in the crosswalk. So we will set up some decision logic to have it take into account the pedestrian.
4. Select the van.
5. Select the Character Objects:Decision Bead page. The Decision Bead page is displayed (Figure 3-17).

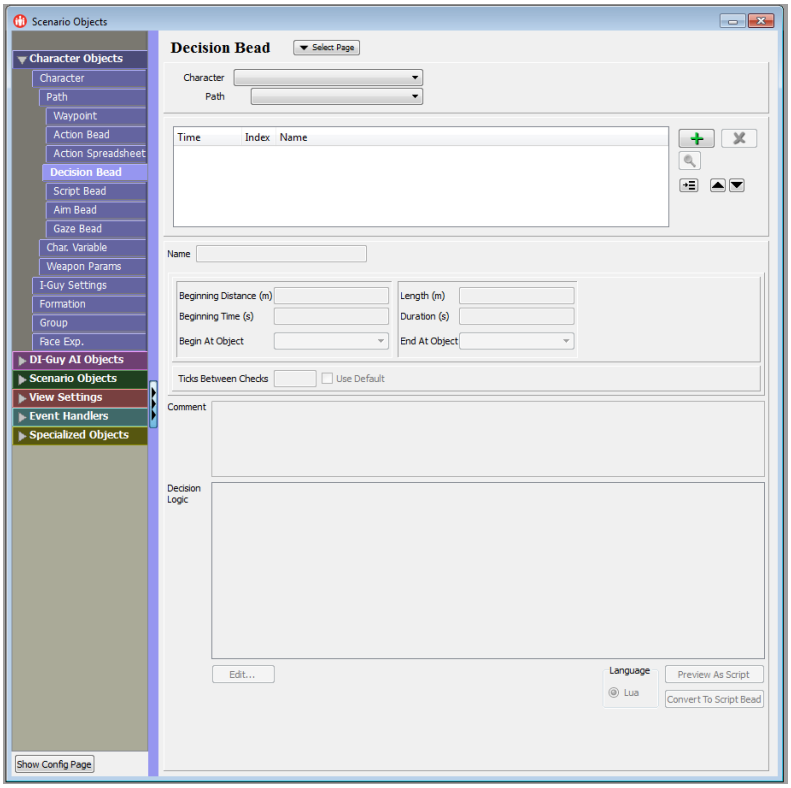


Figure 3-17. Decision Bead page

6. Click the Add button (+). A decision bead is added to the list. It is a rotating red cross. By default, it is placed at the beginning of the character's path.
7. We want the van to decide whether or not to enter the intersection when it gets close to it. Set the Beginning Distance value to 10 and press **`Enter`**. The decision bead moves 10 meters from the start of the path (Figure 3-18). Depending on where you placed the van, you might want the decision bead to be closer to the start of the path or farther along.

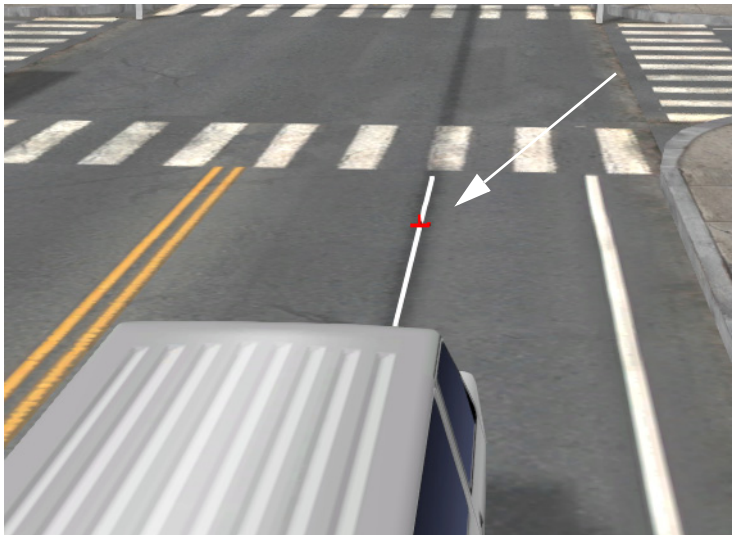


Figure 3-18. Decision bead

8. In the Length box, specify 10 (meters). This means that the van will continue to test the decision logic for 10 meters. We want it to keep testing so that when the test becomes false, it will continue driving.

   Now we need to add the decision logic for the van. We want it to check to see if the pedestrian is in the sensor region. If it is, we want it to wait for the pedestrian to cross the road before continuing.

Decision beads take effect during the period or region specified on the Decision Bead tab. A typical problem for beginning users is specifying a decision that only happens at time 0. The Distance into Path field determines when the decision bead becomes active for the character, specified either as a distance along the path or as the time from the beginning of the scenario. The Length field determines the period over which the decision logic is applied, specified either as a distance along the path or as a duration.

10. Click Edit (at the bottom of the page). The Decision Editor opens (Figure 3-19).

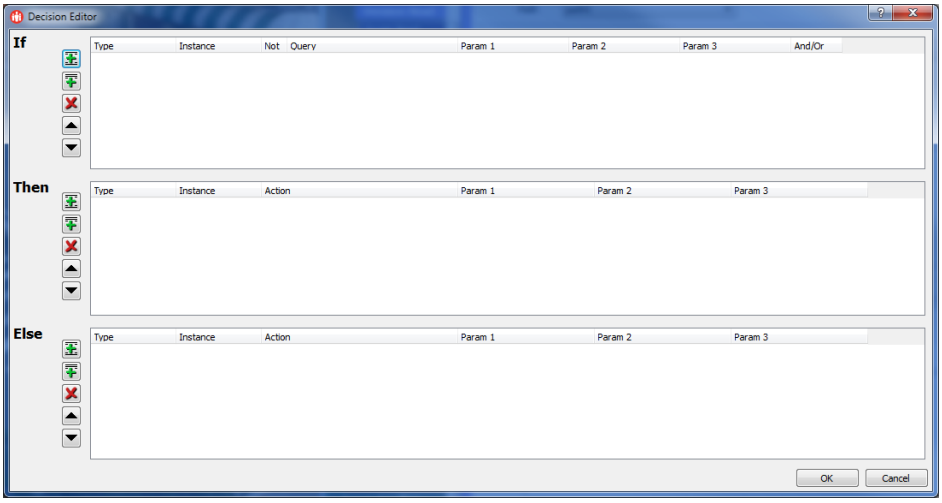


Figure 3-19. Decision Editor

11. In the If section, click the Add button. (For the first line it does not matter which button you click.) A list is added in the Type column.
12. In the list, select `sensor_region`. A list is added to the Instance column.
13. In the Instance column list, select the region you added, probably region0. A list is added to the Query column.
14. In the Query column list, select `contains_character`. A list is added to the Param 1 column.
15. In the Param 1 column list, select the mped07 character. The If condition is complete (Figure 3-20). You could read this condition as: "If character mped07-2 is in sensor region region0."

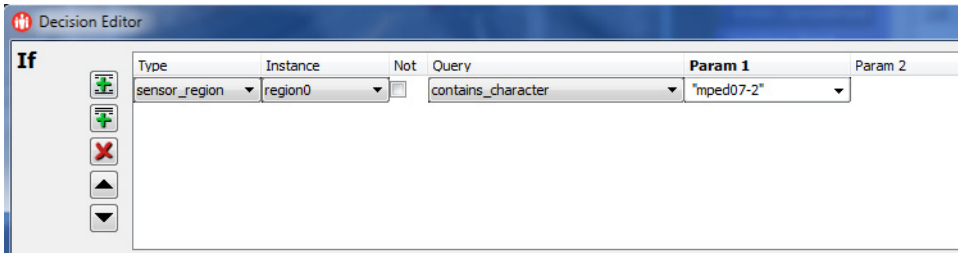


Figure 3-20. If condition

16. In the Then section, click the Add button. A list is added to the Type column.
17. In the Type list, select `character`. A list is added to the instance column.
18. In the Instance column, accept the default – `this_character`. That means the character for whom you are writing the decision logic (the van). A list is added to the Action column. It has all the actions that are available for the character.

19. In the Action column, select `force_action_with_duration`. A list is added to the Param 1 column. The actions available for vehicles are various speeds and we want the van to stop.
20. In the Param 1 list, select `still`. This makes the van stop. A list is added to the Param 2 column.
21. In the Param 2 list, select the text `(duration, must specify)` and replace the text with 10. A larger number is OK too. We want the van to stop long enough for the pedestrian to cross the street. The decision logic is finished (Figure 3-21). You could read the entire logic as: "If character mped07-2 is in sensor region region0, then make this character (the van) stop for 10 seconds."

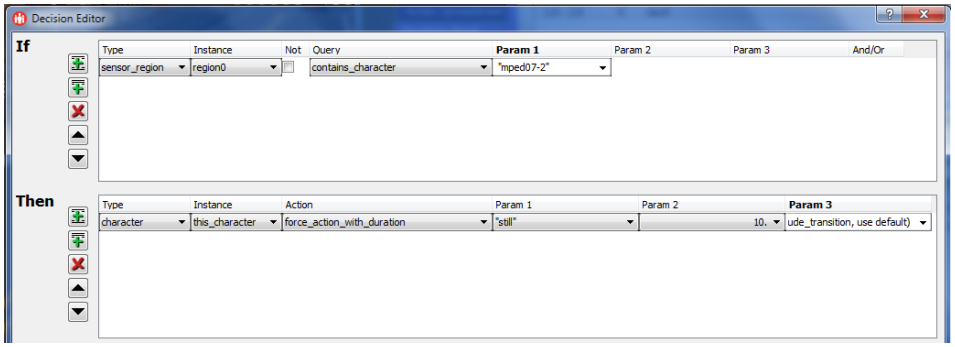


Figure 3-21. Completed decision logic in Decision Editor

22. Click OK. The decision logic is displayed in the Decision Logic window on the Decision Bead page (Figure 3-22).

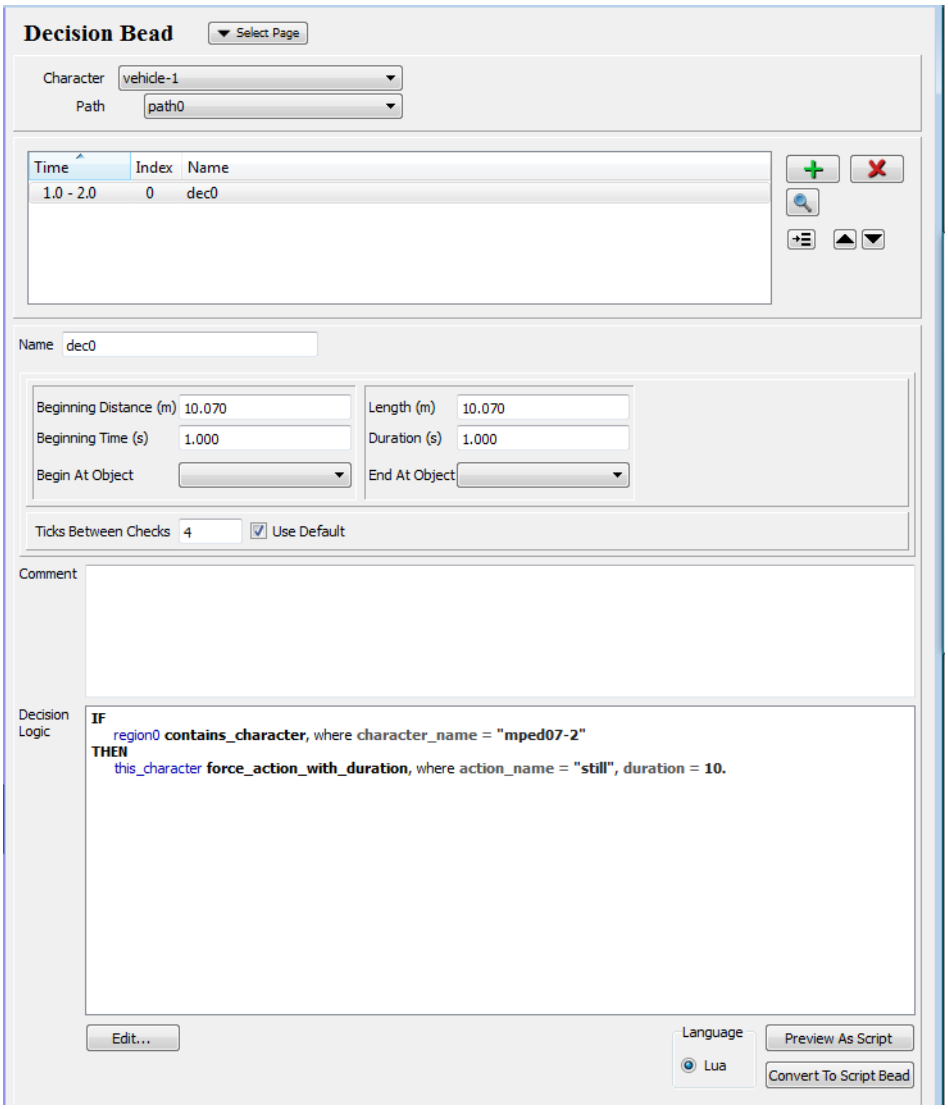


Figure 3-22. Decision logic

23. Play the scenario. See if the van stops to let the pedestrian cross.

3.6. Cameras

The view shown in the 3D View is determined by the location and settings of the camera. Earlier you learned how to reposition the camera to change the view. This section covers additional camera topics that you may find useful.

3.6.1. Saved Camera Settings

DI-Guy Scenario lets you define and save as many camera settings as you like. You can load any of the first 10 camera settings into the camera with one keystroke.

1. Click in the 3D View to make it the active window.
2. Press 1. The camera changes to the camera settings saved to setting 1.
3. Press 2 through 9 to cycle through the rest of the settings.
4. Press 0 to return to the default initial view.
5. Select the View Settings:Camera page (Figure 3-23).

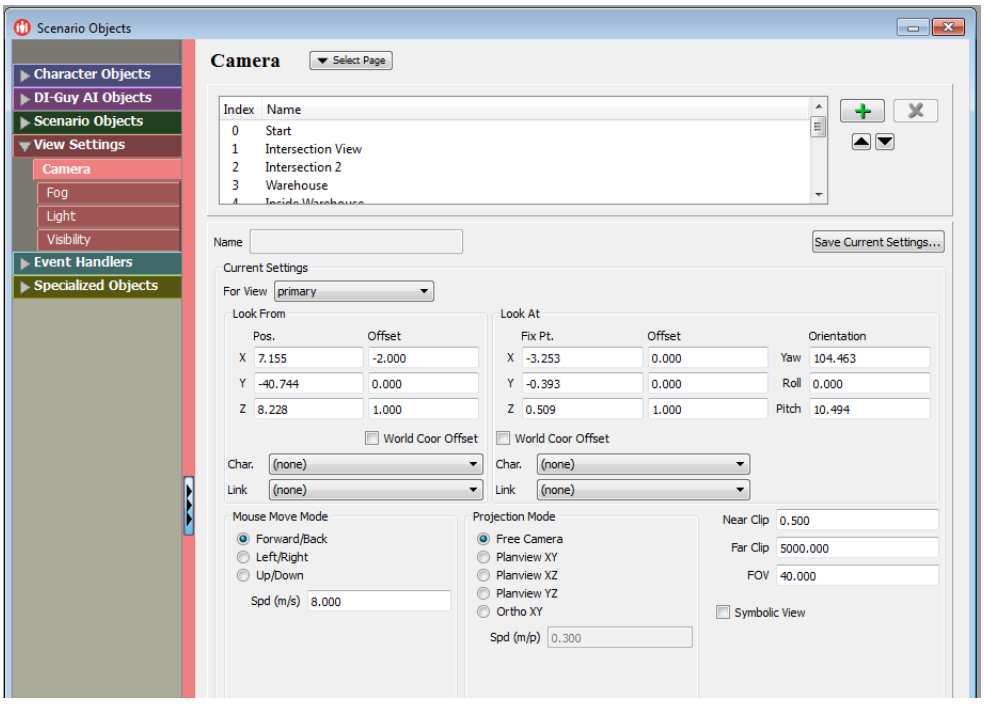


Figure 3-23. Camera page

6. In the 3D View, cycle through the saved settings again and note how each setting is listed in the window on the Camera page and that the various parameters change for each setting.

Save a Camera Setting

Although DI-Guy Scenario comes with a set of saved camera settings for each terrain, you can also save your own. In fact, you will probably want to do so as you create your own scenarios and need to view them from particular locations.

1. With the cursor in the 3D View window, type 3. This moves the camera to saved setting 3. If you are using the default terrain and have not made any changes, this camera looks at the warehouse.
2. Move the camera to some interesting location, such as facing the cafe on the corner.
3. Type **`Shift`**+3 (use the number keys above the letter keys, not the numeric keypad). This saves the current camera to setting 3. It overwrites the old setting.
4. Type 4. This should move you inside the warehouse.
5. Press 3. Remember that originally you would have been looking at the warehouse, but now you are looking at the cafe (or wherever you pointed the camera). You created a new camera setting.

You can add as many settings as you want on the Camera page. The additional settings are not mapped to number keys, but you can access them from the Camera page and can rearrange the list of settings so that different settings are accessible from number keys. For details, please see "Saving and Loading Camera Settings," on page 8-5.

3.6.2. Teleport the Camera

In an earlier tutorial, you learned how to move the camera in various directions. You can also cause the camera to fly from one location to another.

➤ To teleport the camera, point the mouse where you would like to place the camera, then **`Shift`**+left-click. The camera flies to the new location, and spins around to face where it came from.

3.6.3. Tether the Camera to a Character

You can attach a camera to any person or vehicle in the scenario. When attached, the camera travels with the person.

1. In your scenario, create a character with a path.
2. Select the View Settings:Camera page (Figure 3-23).
3. In the Look From group box, in the Char. list, select the character you just created. The camera will jump to a view of the back of the character.
4. In the 3D View, move the camera so that you can see a bit more of the character, such as looking over its shoulder (Figure 3-24).

Figure 3-24. Tethered camera

5. Play the scenario. As the character walks along the path, the camera moves with it.

3.6.4. Tracking a Character

You can also create a camera setting that tracks a person or vehicle as it travels in the scenario. When the camera is tracking a character, it stays in the same place and just changes the orientation to keep the character in view.

1. In your scenario, create a character with a path.
2. Orient the camera so it is along the path perpendicular to it.
3. Select the View Settings:Camera page (Figure 3-23).
4. In the Look From group box, set the Char. value to `(none)`.
5. In the Look At group box, in the Char. list, select the character you just created. The camera changes to look at the character.
6. Play the scenario. As the character walks along the path, the camera changes orientation to track its movement.

You can combine the Look At and Look From parameters to show the relationship between two characters as they travel.

1. Add another character to your scenario.
2. On the View Settings:Camera page, select the first character in the Look At character list and the second character in the Look From character list.
3. Play the scenario. The camera is oriented as if the second character is watching the first one walk.

3.6.5. Camera Point-of-View (POV)

DI-Guy Scenario has camera modes that support 1st person point of view (POV). You can attach the camera to travel with any entity, showing the world from their point of view. In POV mode, the camera travels with the selected person or vehicle and maintains the specified relative position and orientation.

1. Select the View Settings:Camera page (Figure 3-23).
2. Select a character in the Look From character list.
3. Select the same character in the Look At character list.
4. In the 3D View, adjust your view to get the location you want.
5. Play the scenario.

3.7. Load Your Own Terrain

Terrain models form the surroundings in which a scenario takes place. Up to now you have been using the default example terrain provided with DI-Guy Scenario. Once you start using DI-Guy Scenario for your own simulations, you will probably want to use your own terrain.

DI-Guy Scenario loads terrain models in OpenFlight format (*.flt*). Terrain models and tools for creating them are available from a number of sources. We are happy to help create terrains for you, and can also recommend terrain building specialists.

Since we do not know what terrains you have available, to demonstrate loading your own terrain, you will load one of the other terrains provided with DI-Guy Scenario. Feel free to load your own terrain instead.

To load your terrain model:

1. Select the Scenario Objects:Scene Object page (Figure 3-25).

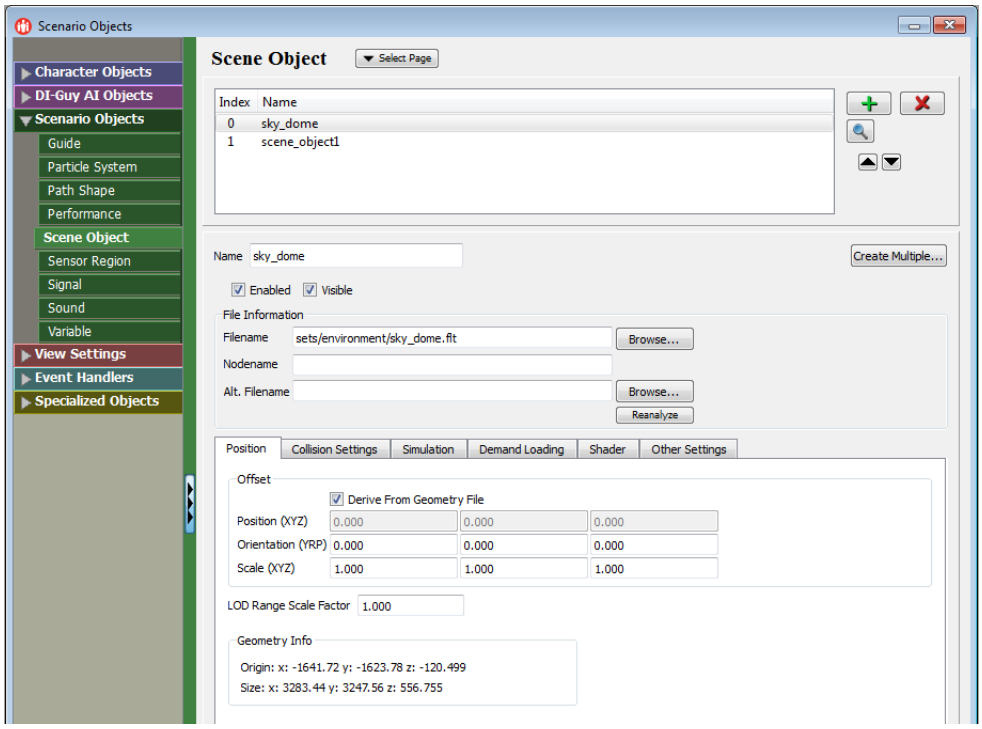


Figure 3-25. Scene Object page

2. In the list, select scene_object1. By default, this is *default_stage11.flt*, as indicated in the Filename box.
3. On the Filename line, click Browse. A file chooser dialog box opens to the *./geometry/sets* directory.
4. Browse to the *sunnyvale2/flt* directory.

5. Select *sunnyvale.flt*.
6. Click Open. The Sunnyvale terrain loads into the 3D View.
7. Press 0 to select a saved camera setting (Figure 3-26).

Figure 3-26. Sunnyvale

3.8. Change the Environment

The default environment is a clear day with overhead lighting. You can add lighting effects in the 3D scenario. You can also add fog or haze.

1. Create a new scenario.
2. Select the View Settings:Light page (Figure 3-27). The default light settings for the scenario are listed in the window.

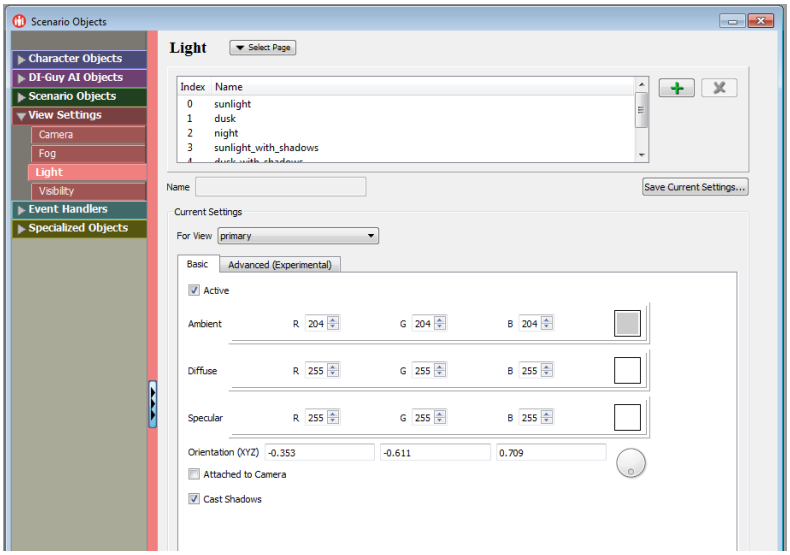


Figure 3-27. Light tab

3. Click each setting and see how it changes the lighting in the scene.
4. Pick one of the light settings and change some parameters. For example change the Diffuse color to red.

Do not worry about messing up settings. You can always create a new scenario and get back to the default settings.

5. Select the View Settings:Fog page (Figure 3-28). The fog settings for the scenario are listed in the window.

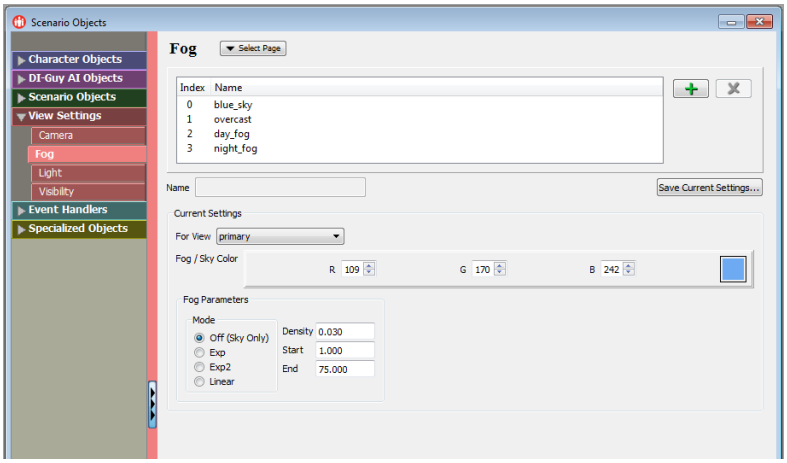


Figure 3-28. Fog tab

6. As you did with lighting, view the fog settings and experiment with changing them. For dramatic effect, change the Fog/Sky Color to red or pink. Change the Density setting to a higher number.

4. Creating and Editing Characters

This chapter explains how to create and manage characters. Character paths and actions are described in succeeding chapters.

Creating a Character 4-3
- Creating a Character and a Path 4-5
- Creating a Character on the Character Page 4-6
- Using the Appearance Wizard to Choose a Character 4-6

Selecting a Character 4-8

Deleting a Character 4-9

Changing a Character's Name 4-9

Enabling Characters 4-9

Hiding Characters 4-10

Editing Characters 4-10
- Changing a Character's Character Type 4-10
- Changing a Character's Appearance 4-10
- Setting the Initial Path 4-11
- Initial Position and Orientation 4-11
- Setting a Character's Scale 4-12
- Setting Time In and Time Out 4-12
- Specifying that a Character is a Scene Object 4-13
- Apply Actor Scale to Action Bead Travel 4-13
- Advanced Character Editing 4-13

Cutting, Copying, and Pasting Characters 4-20

Undoing Changes 4-20

Creating Character Variables 4-21

Parenting One Character to Another 4-22

Specifying Weapon Parameters 4-23
Using a Joystick or Keyboard to Control a Character 4-24
Using a Joystick 4-24
Keyboard and Mouse Mode 4-27
Creating Formations 4-28
Formation Subgroups 4-31
Formation Roles 4-32
Creating Groups 4-33
Expressive Faces 4-34
Character Labels and 3D View Labels 4-35
Enabling and Disabling Default Character Labels 4-36
Editing the Style of Character Labels 4-37
Labels and the DI-Guy API 4-38
Creating Screen and Button Labels 4-39
Configuring Chains 4-44

4.1. Creating a Character

When you create a character, it is given a default name based on the character type and the sequence in which it has been created. You can change a character's name and other attributes. The new character is listed on the Character Objects:Character page and in the character lists in other windows.

You can create characters using the PeopleBlitzer or on the Character page.

The organization of pages in the Character Objects section implies their relationships. For example, the Path page is indented under the Character page because paths are specific to characters. The various waypoint and bead pages are specific to individual paths, so they are indented further.

For convenience, the pages in the Scenario Objects window are identified by their top level section name and page name, for example, Character Objects:Waypoint. We do not worry about the other subordinate relationships in this naming convention.

To create a character using the PeopleBlitzer:

1. In the Input Mode window, click PeopleBlitzer. The right side of the dialog box lists options for character type, appearance, and other attributes (Figure 4-1).

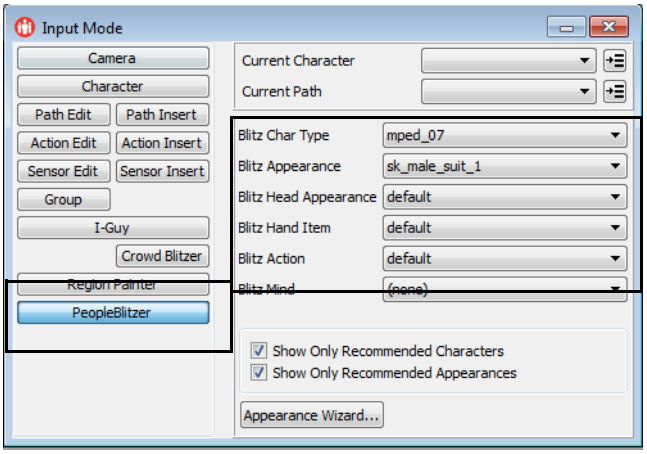


Figure 4-1. Input Mode window

2. In the Blitz Char Type list, select the type of character that you want to create.
3. In the Blitz Appearance list, select the appearance you want for this character.

If you are not sure which combination of character type and appearance you want to use, the Appearance Wizard can help you build a character. For details, please see "Using the Appearance Wizard to Choose a Character," on page 4-6.

4. Optionally, if applicable, select a hand item for the new character.
5. Optionally, if applicable, select a Blitz Action. This is an action that the character will do as soon as the scenario starts to run. If you select an action, an action bead is placed at the location where you create the character.
6. Optionally, if applicable, select a Blitz Mind. (DI-Guy AI only.)
7. In the 3D View, click where you want to place the character. The character is added to the scene.

Figure 4-2. New character

By default, when you create a character, DI-Guy Scenario switches to Path Edit mode. Therefore you cannot place several characters in succession without switching back to PeopleBlitzer mode in the Input Mode window. However, this behavior can be changed in the Editor Preferences dialog box. If you turn it off, you can create multiple characters one after the other without switching modes.

4.1.1. Creating a Character and a Path

When you add a character to a scene using the PeopleBlitzer, you can give it an initial path at the same time.

To create a character and a path at the same time:

1. Follow the instructions for specifying the character to create as described in "Creating a Character," on page 4-3.
2. When you add the character to the scene, click on the starting point of the path and while holding down the left mouse button, drag a line to the end point of the path (Figure 4-3).

Figure 4-3. New character with path

To create a more complex path, hold down **`Ctrl`** and continue clicking to create more waypoints.

4.1.2. Creating a Character on the Character Page

You can add characters on the Character Objects:Character page. They are created in the center of the terrain and use the last character type specified in the PeopleBlitzer. If you create multiple characters in succession, they are placed on top of each other.

To create a character on the Character page:

1. In the Scenario Objects window, select the Character Objects:Character page (Figure 4-5).
2. Click the Add button (+). A character is added to the list and displayed in the 3D View.
3. Optionally, change the character type, appearance, and so on.
4. Optionally, in the 3D View, select Character mode and move the character to the desired location.

4.1.3. Using the Appearance Wizard to Choose a Character

The Appearance Wizard helps you choose a character and appearance based on the type of character you want. It lets you choose from a variety of criteria and lists character types that match the cumulative criteria. At any point you can select a character type.

To use the Appearance Wizard:

1. In the Input Mode window, click PeopleBlitzer.
2. Click Appearance Wizard. The Appearance Wizard opens (Figure 4-4).

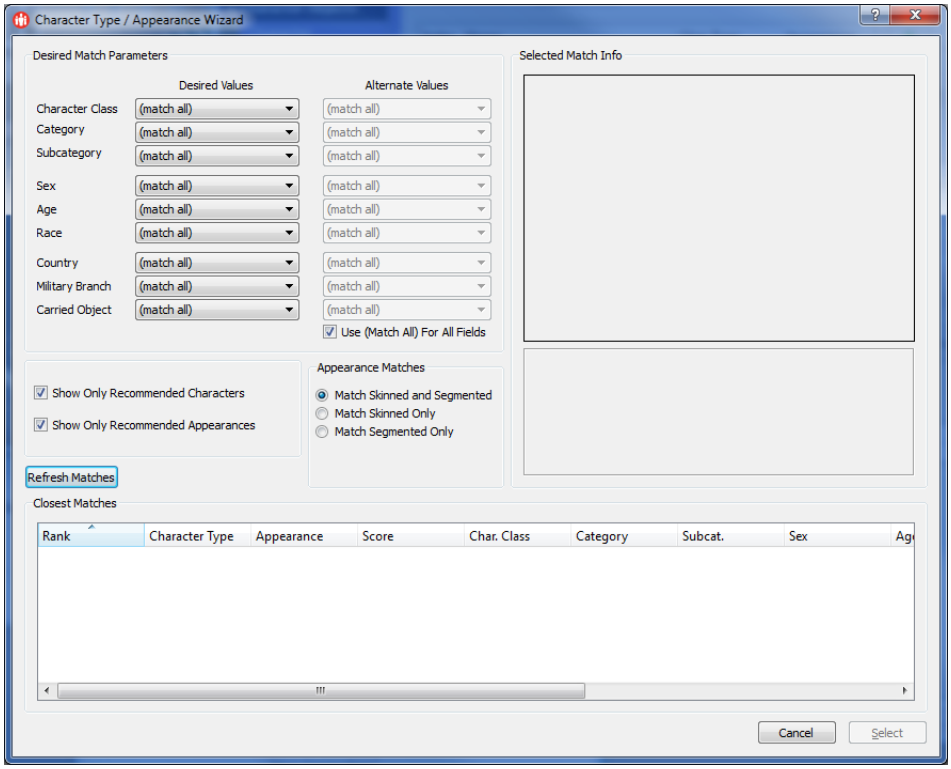


Figure 4-4. Appearance Wizard

3. In the Character Class list, select the type of character you want, such as human, animal, or vehicle. If you select an option other than `match all`, the Closest Matches window lists all character types that match your choice.
4. In the Category list, select a category, such as civilian or military. The list of categories is determined by the character class.
5. Continue choosing options from each parameter list that is appropriate to the character you want to create.
6. Optionally, limit the character matching to skinned or segmented characters.
7. Select a character from the Closest Matches list.
8. Click Select.

4.2. Selecting a Character

To select a character, use one of the following methods:

- On the Character Objects:Character page (Figure 4-5), select the character in the list.
- In the Input Mode window, select a character in the Current Character list. You must be in Character mode, Action Edit mode, Path Edit mode, or PeopleBlitzer mode.
- In the Input Mode window, click Character and click a character in the 3D View.

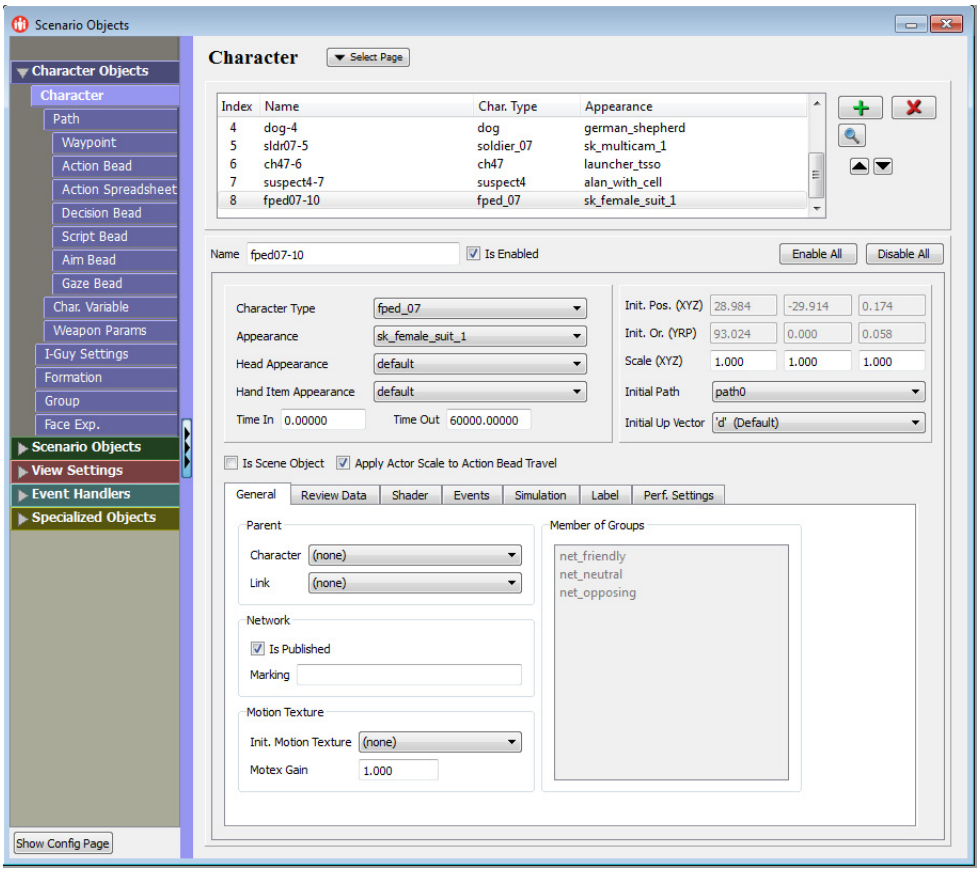


Figure 4-5. Character page

4.3. Deleting a Character

When you delete a character, the indices of all the following characters are decremented by one.

To remove a character from the scenario:

1. Select the character to be deleted.
2. On the Character Objects:Character page, click the Delete button (✖) or in the Input Mode window, click Delete Current Character.

4.4. Changing a Character's Name

New characters are given default names. You may want to give your characters more descriptive names. Names must be unique within the scenario.

To change a character's name:

1. Select the Character Objects:Character page (Figure 4-5).
2. Select the character whose name you want to change.
3. Type a new name in the Name text box.

4.5. Enabling Characters

By default, characters are enabled, which means they can participate in the scenario. You can enable and disable characters individually or all at once. Disabled characters are not visible in the 3D View and do not participate in the scenario. Disabled characters are still listed on the Character page and in character lists. In the list on the Character Objects:Character page, disabled characters are listed in light gray text.

Enabling and disabling a character is not the same thing as changing visibility. Disabled characters do not participate in the scenario. Invisible characters are still in the scenario and get updated, they are just hidden from view. For details about hiding characters, please see "Hiding Characters," on page 4-10.

To enable or disable a character:

1. On the Character Objects:Character page (Figure 4-5), select the character.
2. Select or clear the Is Enabled check box.

➤ To enable or disable all characters, on the Character page, click Enable All or Disable All.

4.6. Hiding Characters

You can hide the characters in the scenario. They continue to be simulated, but are not visible.

To hide characters:

1. Select the View Settings:Visibility page.
2. In the Characters group box, select one of the following options:
   - None. No characters are visible.
   - Current. Only the selected character is visible.
   - All. All characters are visible.

4.7. Editing Characters

Characters have many attributes. You can edit most of them.

4.7.1. Changing a Character's Character Type

Every character in DI-Guy Scenario has a character type that defines the actions and appearances available for that character. You can change the character type for characters that are already created. Changing the character type may affect the appearances available for the character and the actions that it can do.

To change a character's type:

1. Select a character.
2. In the Input Mode window or the Character Objects:Character page (Figure 4-5), select a different option from the Character Type list.

4.7.2. Changing a Character's Appearance

Characters of a particular character type share the same link and joint hierarchy and can perform the same set of actions. But these characters can have a wide range of appearances. The appearance determines what the character looks like, including body shape, clothing, and equipment. Equipment can be such things as a backpack, canteen, or weapon. The available appearances depend on the character type of the character and the models installed on your system.

To change a character's appearance:

1. Select a character.
2. In the Input Mode window or the Character Objects:Character page, select a different option from the Appearance list.

Changing the Head Appearance

When you create a person, it has a default head. You can substitute a different head. If you are using the DI-Guy Expressive Faces option, you can replace a static head with one that can make facial expressions, move its eyes, and synchronize the mouth to speech.

To change a character's head appearance:

1. Select a character.
2. In the Input Mode window or the Character Objects:Character page, select a different option from the Head Appearance list.

4.7.3. Setting the Initial Path

Characters can have zero or more paths. The initial path is the path used when the scenario starts.

To change a character's initial path:

1. Select a character.
2. On the Character Objects:Character page, select an option from the Initial Path list.

4.7.4. Initial Position and Orientation

The initial position and orientation values are read-only.

4.7.5. Setting a Character's Scale

Characters can have their size scaled in height, width, depth, or a combination of these dimensions. Motions are scaled accordingly. Changes in scale can also apply to action bead travel. This is configurable. Figure 4-6 shows a character scaled to 4, 4, 4 next to characters at the default 1, 1, 1 scale.

Figure 4-6. Scaled character

To change a character's scale:

1. Select a character.
2. On the Character Objects:Character page (Figure 4-5), in the Scale boxes, set X (depth), Y (width), and Z (height) values.
3. If you want the scale to affect action bead travel, select the Apply Actor Scale to Action Bead Travel check box.

4.7.6. Setting Time In and Time Out

Characters can have a time in and time out, in seconds, independent of the scenario time in and time out. This can be useful for effects. The default time in is 0 (the start of the scenario). The default time out is 60000.

To change a character's time in and time out:

1. Select a character.
2. On the Character Objects:Character page (Figure 4-5), change the values, in seconds, in the Time In box and the Time out box.

4.7.7. Specifying that a Character is a Scene Object

A character that is a scene object acts as part of the terrain, both in terms of collision and altitude functions for AI characters. A scene object does not move. If you specify that a character be a scene object while a scenario is running, it stops moving.

➤ To specify that a character be a scene object, on the Character Objects:Character page, select the Is Scene Object check box.

4.7.8. Apply Actor Scale to Action Bead Travel

You can specify whether or not changes in scale affect the distance a character travels. Default: On.

➤ To specify that a characters scale affect action bead travel, on the Character Objects:Character page (Figure 4-5), select the Apply Actor Scale to Action Bead Travel check box.

4.7.9. Advanced Character Editing

The Character Objects:Character page has many advanced character attributes that are arranged on tabs at the bottom of the page. These settings are summarized in this section.

Character Page: General Tab

- Parent. Parent the current character to another character. You specify the character that is to be the parent and how they are linked. For more information, please see "Parenting One Character to Another," on page 4-22.

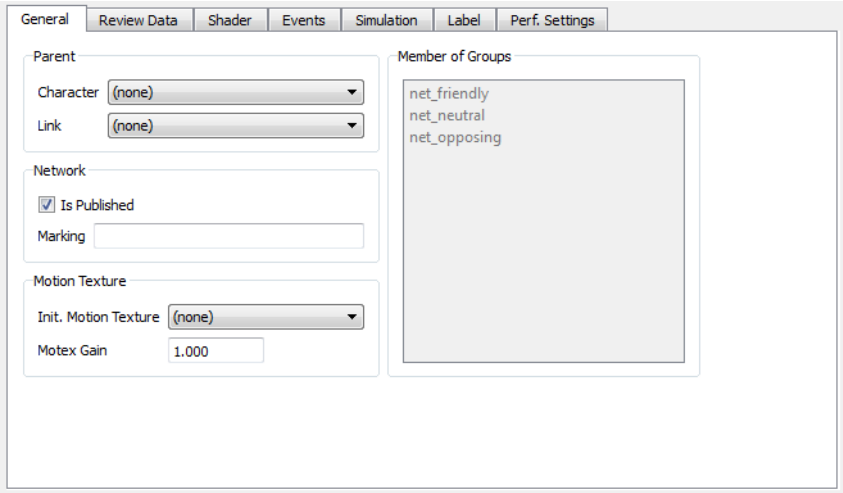


Figure 4-7. General tab

- Network. The Is Published check box determines whether this character should be broadcast when DI-Guy Scenario is networked using HLA or DIS. You can specify marking text.
- Motion Texture. You can specify a motion texture that is added onto whatever motion is generated by an action. A typical motion texture varies each of the joints in a characteristic way to provide richness and randomness to the behavior. The Initial Motion Texture field specifies the motion to be used for this purpose. The Motex Gain parameter indicates how strongly to add the motion texture to the underlying motion. Any DI-Guy motion can be used as a motion texture.
- Member of Groups. Lists the groups the selected character belongs to. Groups are a collection of one or more characters. The networking module has three pre-defined groups, net_friendly, net_neutral, and net_opposing. You can use groups to classify characters, vehicles and props to facilitate applying decision logic to classes of entities, rather than to individual entities. For more information about groups, please see "Creating Groups," on page 4-33.

Character Page: Review Data Tab

When you run a scenario, DI-Guy Scenario saves data for future playback. The saved data is called review data. The Review Data tab lets you enable and disable saving of review data and configure how much data is saved. For characters, review data encompasses the state of all joints in the body at every DT.

- Size. The initial amount of data to be allocated for this purpose.
- Increment. The size of subsequent chunks of data to allocate if the initial amount of data from Size is surpassed.

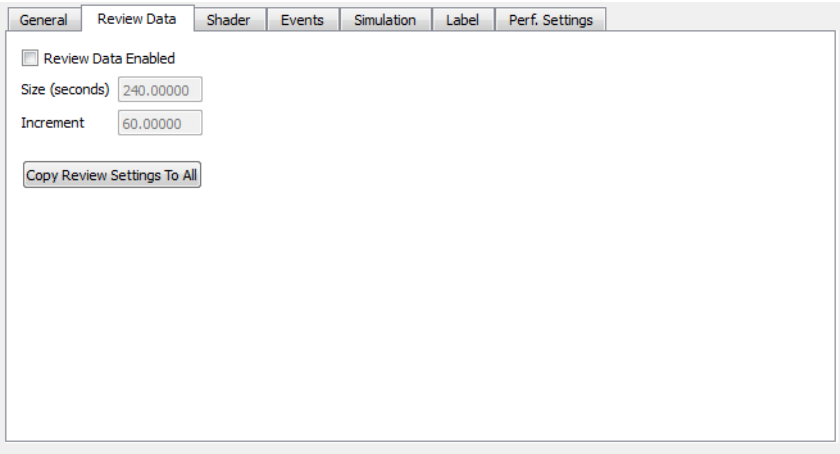


Figure 4-8. Character page, Review Data tab

Character Page: Shader Tab

This tab lets you specify the shader that is applied to the character for rendering (advanced users only). Shaders are computer programs that simulate the effect of light on objects in a simulation. Many characters are set up to use a variation system that lets different parts of the character be adjusted in real-time using GPU based shaders.

To change shader variations:

1. On the Shader tab, select Enable Variations. If the character supports variations one is randomly chosen and list of options is displayed (Figure 4-9). (If the character does not support variations, DI-Guy Scenario displays a message.)

   In the property sheet, under the Link:shape section the Variation Name line lists a variation.

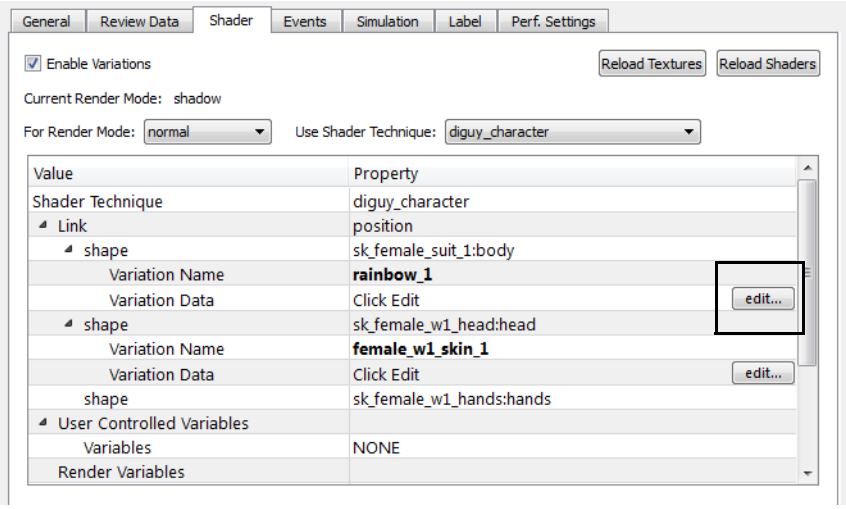


Figure 4-9. Character page, Shader tab

2. Optionally, select a different variation in the Variation Name list.
3. On the Variation Data line, click edit. The Ramp Editor opens (Figure 4-10).

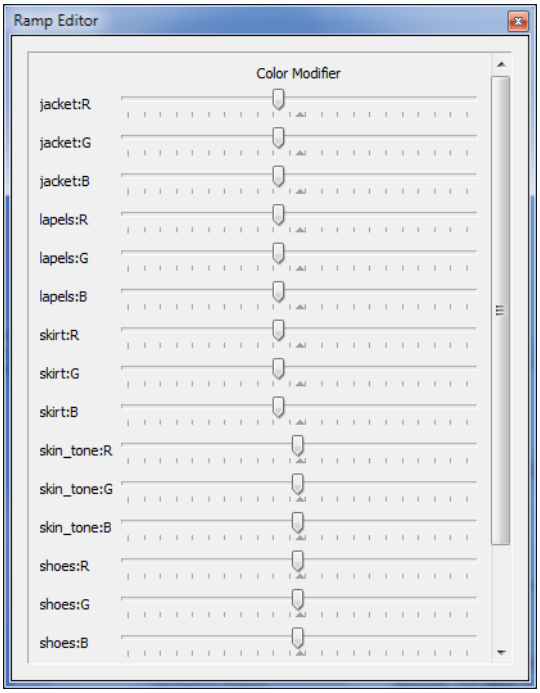


Figure 4-10. Ramp Editor

4. Change the colors as desired.
5. Close the Ramp Editor.

Variation changes are saved to disk as part of the character's description and are sent over the network using a custom PDU.

Character Page: Events Tab

The Events tab lets you map event handlers to callback events. For details, please see "Creating Screen and Button Labels," on page 4-39 and Chapter 9, *Events and Event Handlers*.

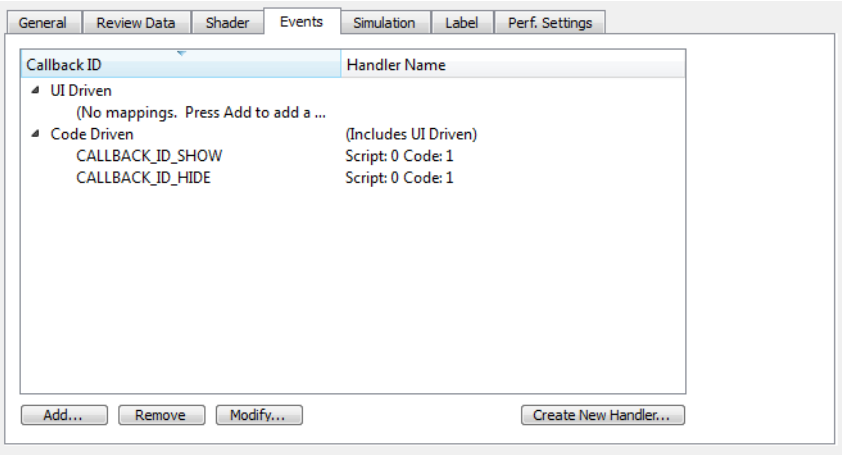


Figure 4-11. Events tab

Character Page: Simulation Tab

By default, the character performance is controlled by the DI-Guy Motion Engine. This page has options for a variety of simulation modes, most of which have been developed over time for specific DI-Guy purposes. If you have questions, please contact support@mak.com.

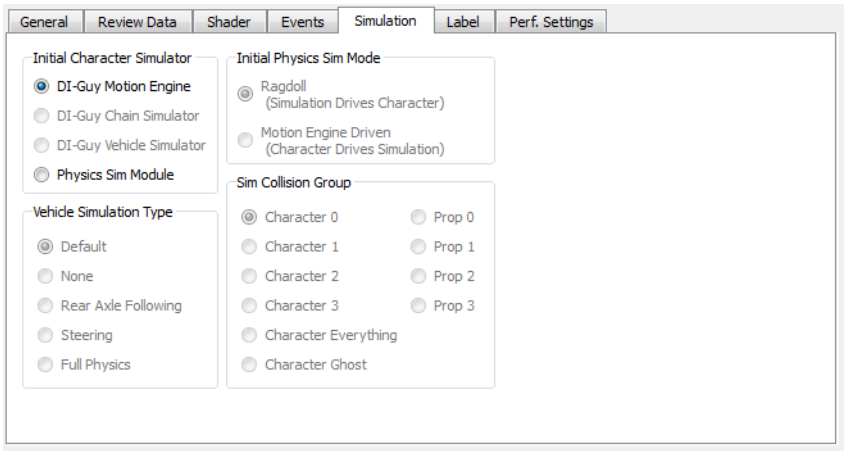


Figure 4-12. Simulation tab

Character Page: Label Tab

You can display a text label above a character. By default, the label is the name of the character. This tab lets you specify the label text, color, and other display characteristics. For more information about labels, please see "Character Labels and 3D View Labels," on page 4-35.

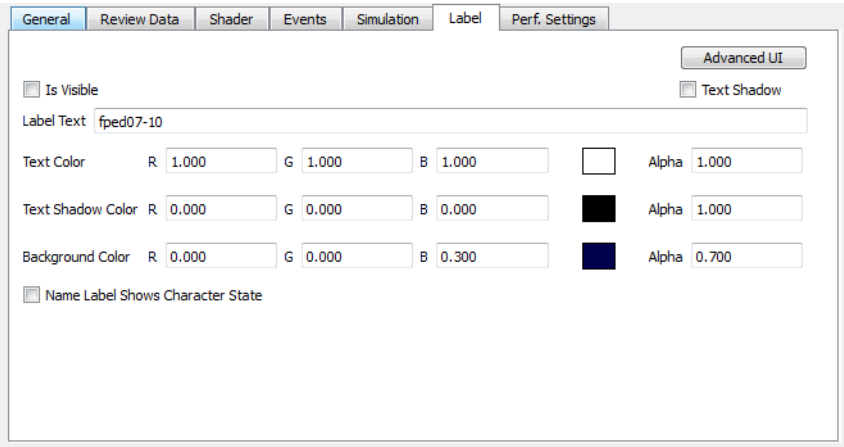


Figure 4-13. Label tab

Be sure to enable character labels in the View Settings:Visibility page.

Character Page: Performance Settings Tab

If you are not using the Load Manager, you can set performance parameters on this tab. For more information about DI-Guy Scenario performance, please see Chapter 10, *Configuring Performance*.

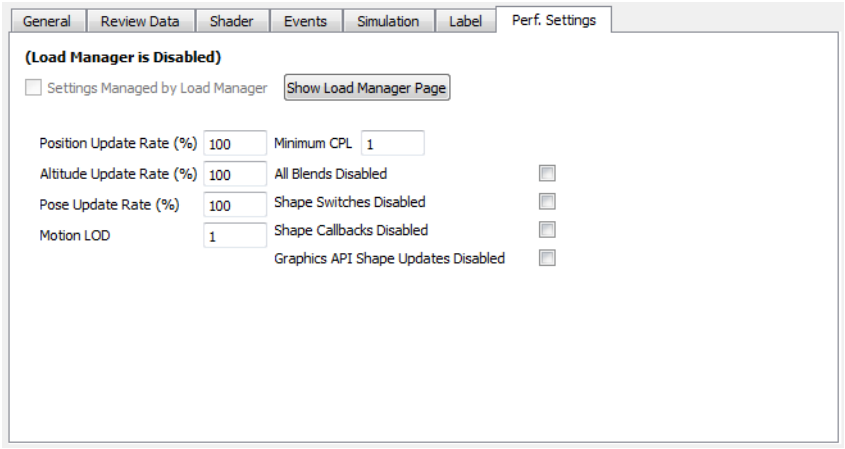


Figure 4-14. Performance Settings tab

4.8. Cutting, Copying, and Pasting Characters

You can cut, copy, and paste characters and their paths. When you copy a character, the character and its paths, waypoints, and actions are copied. Deleting a character puts it on the clipboard, so you can paste it back into the scene.

Cut, copy, and paste does not work for individual waypoints or action beads.

When you paste a character into the scene, DI-Guy Scenario may change its name to avoid name collisions with other characters.

Cut, copy, and paste are sensitive to the mode. To operate on characters and their paths, you must be in PeopleBlitzer mode. To cut a single path, you must be in Path Edit mode.

➤ To cut, copy, or paste, use standard keyboard shortcuts or options on the Edit menu.

4.9. Undoing Changes

DI-Guy Scenario has multiple levels of undo.

➤ To undo the last action, choose **Edit** → **Undo**, or type **`Ctrl`**+**`z`**.

4.10. Creating Character Variables

You can define variables that are associated with a character. Each variable should be given a descriptive name and an appropriate initial value.

There are many API functions for dealing with character variables. They all have the form diguyCharacter->variable_<etc>. Variables are useful when combined with decision and script logic.

To create a character variable:

1. Select the character for which you want to create a variable.
2. Select the Character Objects:Character Variable page (Figure 4-15).

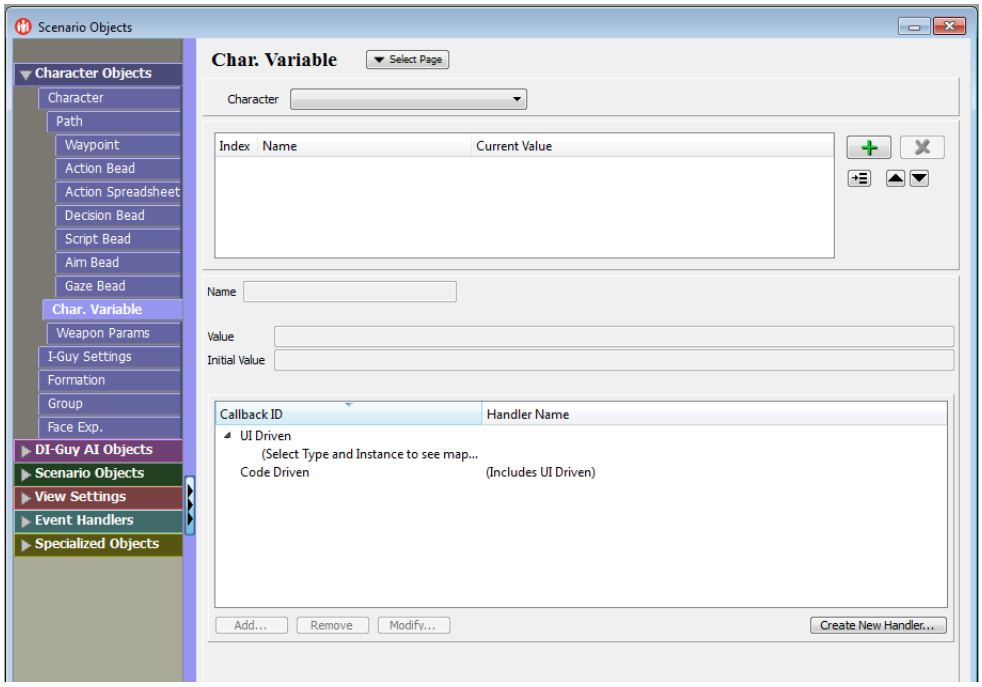


Figure 4-15. Character Variable page

3. Click the Add button (+). A new variable is added to the list.
4. Optionally, type a name for the variable in the Name box.
5. If you want to change the value of a variable while a scenario is running, set the value in the Value box.
6. In the Initial Value box, enter the value this variable will have at the start of the scenario or when it is reset.

4.11. Parenting One Character to Another

Suppose you want a person to travel on the deck of a moving ship or inside of a moving train. DI-Guy Scenario supports such hierarchical scenario behavior through parenting. You can specify that one character be parented to another character. Once parented, the child character will move with respect to the location of the parent, even when the parent moves.

Parenting of characters is done using decision beads. This procedure assumes you know how to create them. For details about creating decision beads, please see "Decision Beads," on page 6-13.

To parent a character to another character:

1. Select the child character.
2. Create a decision bead that uses the set_parent function in the THEN or ELSE sections of the Decision Editor. The set_parent action takes the name of the parent character as an argument.

Once parented, the child is positioned so that its origin is coincident with the local origin of the parent. Vehicles typically have their local origins set at ground level to simplify blitzing. So you may need to adjust the origin to get the child where you want it. One trick to help accomplish this is to create the parent character as a scene object at the origin (select Is Scene Object on the Character Objects: Character page), blitz the child character onto the scene object, and adjust the location until it is the way you want it to be. Then you can create the parent as a regular character and perform the parenting.

4.12. Specifying Weapon Parameters

Many DI-Guy Scenario characters can use weapons. The Character Objects:Weapon Parameters page lets you configure the weapons that a character supports.

DI-Guy Scenario often communicates munition information to other simulations when operating in a distributed simulation.

To configure weapon parameters:

1. Select the character for which you want to specify a weapon.
2. Select the Character Objects:Weapon Params page (Figure 4-16).

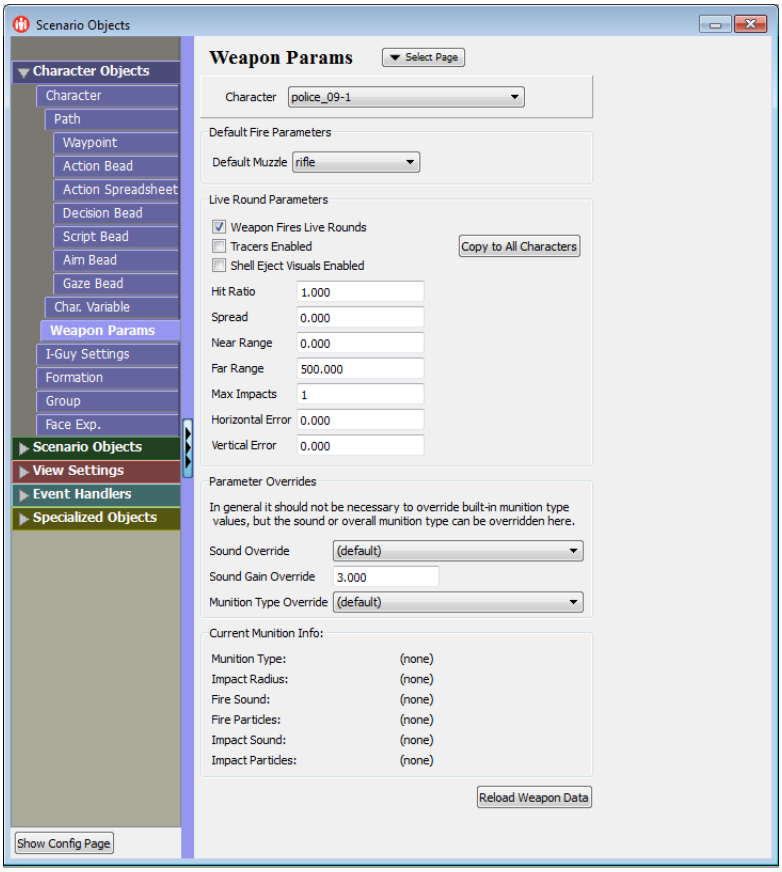


Figure 4-16. Weapon Params page

3. In the Default Muzzle list, select the weapon you want to configure.
4. Optionally, in the Live Round Parameters group box, change values as desired.
5. Optionally, in the Parameter Overrides group box, change values as desired.

4.13. Using a Joystick or Keyboard to Control a Character

When a character is in I-Guy mode, you can control it with a joystick, the keyboard, and the mouse. When DI-Guy Scenario is in I-Guy mode and the 3D View has focus, you can control characters with the joystick or keyboard.

➤ To enter I-Guy mode, in the Input Mode window, click I-Guy.

➤ To exit I-Guy mode, press **ESC**.

4.13.1. Using a Joystick

You can use a joystick to control the behavior of any character. Joystick functionality is based on a three-axis USB joystick with eight buttons and a throttle control. The Microsoft Sidewinder Precision 2 was used for testing (Figure 4-17).

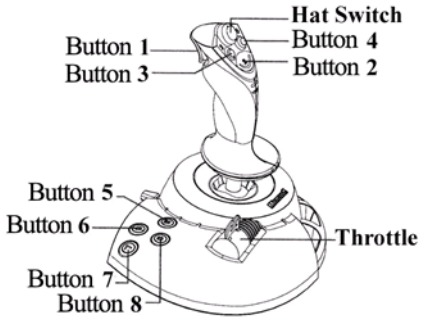


Figure 4-17. Joystick buttons

The joystick can control two kinds of actions:

- Body actions (walking, jogging, running, backing up, and so on.)
- Head actions (head position, gaze location, aim direction).

Body actions are controlled by the joystick itself. Head actions are controlled by the thumb-operated hat switch, mounted atop the joystick. You can also use the throttle control to adjust the speed of body actions.

To place a character under joystick control:

1. Select the character on the Character Objects:I-Guy Settings page (Figure 4-18).

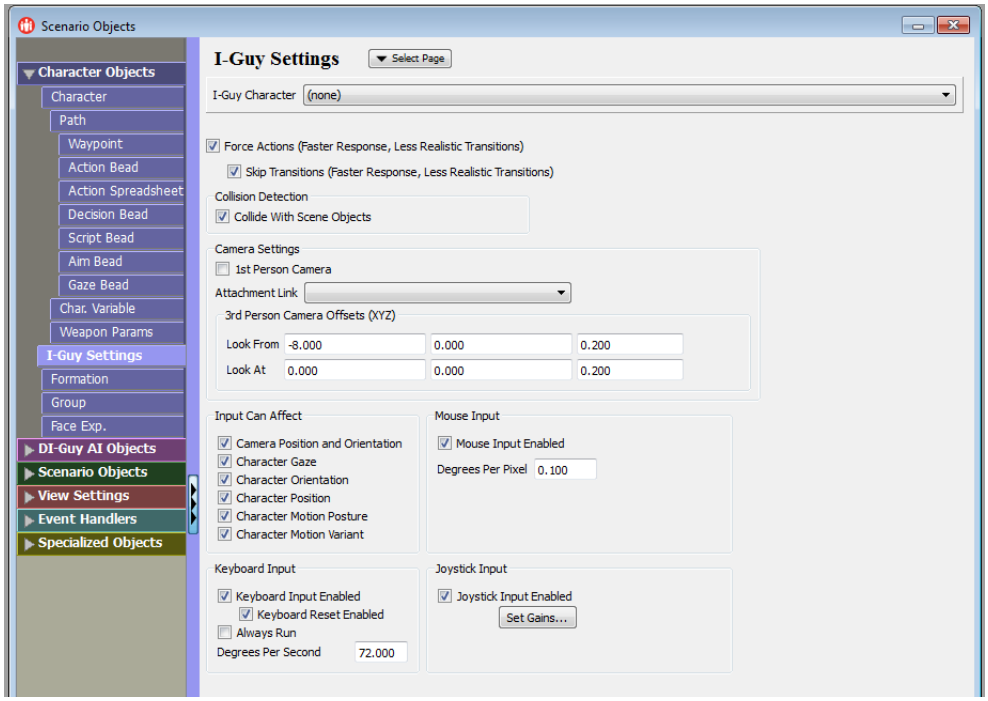


Figure 4-18. I-Guy Settings page

2. Make sure the play button is active on the Time Control window.

A simple collision detection mechanism is available for the joystick character. When enabled, collision detection prevents the character from walking through walls and other obstacles.

To turn on collision detection:

1. Select the I-Guy character.
2. On the Character Objects:I-Guy Settings page, in the Collision Detection group box, select the Collide with Scene Objects check box.

Moving Characters with the Joystick

Characters respond to joystick movement and buttons as follows:

- Joystick forward makes the character move forward.
- Joystick back makes the character back up.
- Joystick left/right slides the character left/right.
- Joystick twist left makes character turn left, while moving or stationary.
- Joystick twist right makes character turn right, while moving or stationary.
- Button 1 fires a weapon. Behavior depends on the point of view setting.
- Button 2 toggles aim mode. Behavior depends on the point of view setting.
- Button 3 toggles between first-person point of view and third-person point of view. You can adjust each of the views using the Look From and Look At fields on the camera page.
- Button 4 toggles sticky view mode.
- Button 5 causes the character to respond more smoothly to action requests, at the expense of responsiveness. There are three responsiveness levels.
- Button 6 causes the character to respond more quickly to action requests, at the expense of smoothness. There are three responsiveness levels.
- Button 7 moves the character further upright. If it is crawling, it crouches. If it is crouching, it stands upright. (Not all characters have all of these behaviors.)
- Button 8 moves the character closer to the ground. If it is standing, it crouches. If it is crouching, it crawls. (Not all characters have all of these behaviors.)

Aiming and Firing in Third Person Point of View

In third person POV, button 2 toggles aim mode. Entering aim mode stops the motion of the character, and it aims the weapon. In this mode, the hat switch controls the direction in which the weapon is aimed. Aim mode works only for characters that have a weapon.

While in aim mode, button 1 fires the weapon, generating muzzle flash and playing the gunfire sound. For sound to be heard, a sound element named “gunfire” must be created. For details, please see “Adding Sound Effects,” on page 7-18.

Aiming and Firing in First Person Point of View

In first person POV, button 2 toggles between aim mode and ready mode. In aim mode, the hat-switch aims the weapon. It moves the character’s head and your view up and down. Button 1 fires the weapon.

Configuring Joystick Gain

You can configure the gain on the joystick movement, throttle, and twist.

To configure gain for a joystick:

1. Choose **Edit** → **Preferences**. The Editor Preferences dialog box opens.
2. Select the I-Guy tab (Figure 4-19).

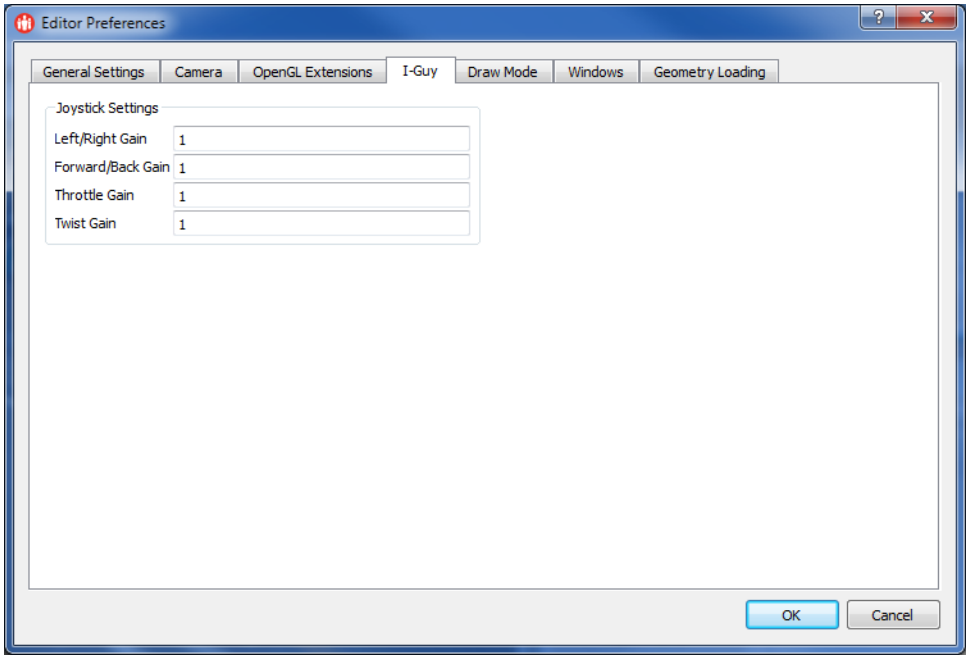


Figure 4-19. Preferences, I-Guy tab

3. Change the settings as desired.

4.13.2. Keyboard and Mouse Mode

In I-Guy mode, you can control characters with the W-A-S-D and other keys to control movement and posture, with the mouse dictating the gaze and orientation. For more information, please see Appendix B, *Keyboard Shortcuts*.

4.14. Creating Formations

Formations allow a group of characters (including vehicles) to follow a leader. The followers can follow in precise relative position to the leader, or they can follow in a more relaxed way.

DI-Guy Scenario comes with predefined formations. You can define one or more of your own formations for use in a scenario. Defining a formation involves choosing a pattern for placement of followers relative to the leader, assigning individual characters to each follower position, and choosing the guide algorithm that controls the travel of each follower with respect to their ideal following position.

Once you have defined a formation, use it by assigning decision logic to the leader that calls a formation or switches from one formation to another. If you are doing this with a decision bead, use the `call_formation` parameter.

Because you must assign followers to formation positions, formations have limited flexibility. DI-Guy AI provides improved formation behaviors. You do not have to assign characters to the individual follower positions and movement in formation is more natural.

To create a formation:

1. Select the Character Objects:Formations page (Figure 4-20).

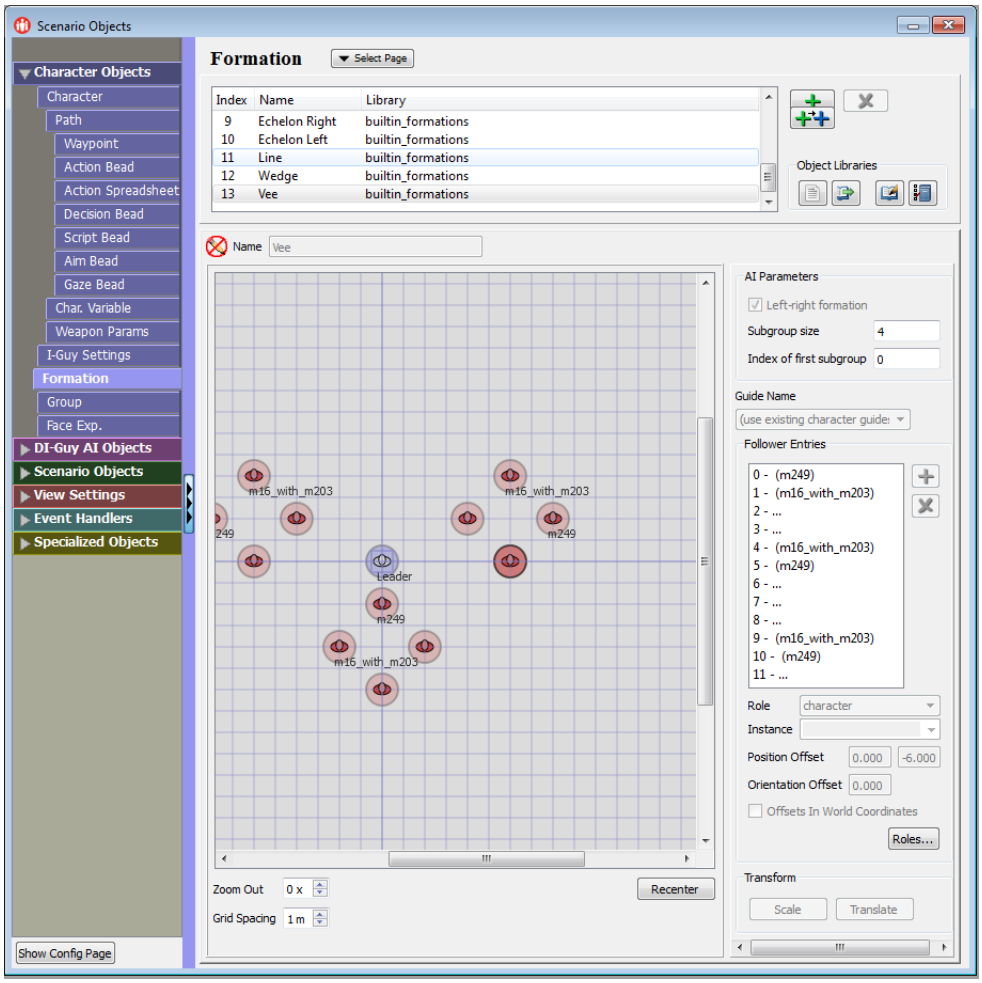


Figure 4-20. Formations page

2. Click the Add button (+). A new formation is added to the list.
3. Optionally, type a new name in the Name field.
4. Optionally, specify the spacing of the rows and columns by changing the Grid Spacing value.
5. In the Follower Entries group box, click the Add button for each member you want to add to the formation.
6. Position the followers. In the grid, drag each follower to the location you want it to have with respect to the leader.

7. Assign characters to followers.
   a. If you have not already created the required characters in the scenario, add them to the scenario (using the People Blitzer).
   b. Select a follower in the grid.
   c. In the Instance list, select one of the characters in the scenario.
   d. Repeat for each follower position.

You do not assign a character to the leader position. The leader is the character who calls a formation in its decision logic.

8. In the Guide Name list, select a guide. Guides determine the travel control algorithm used by each follower as it travels within the formation. Guides are defined on the Scenario Objects:Guide page. For details, please see "Guides," on page 7-3.
9. Optionally, set up subgroups, as described in "Formation Subgroups," on page 4-31.
10. Optionally assign roles to the followers, as described in "Formation Roles," on page 4-32.
11. To expand the size of the formation:
    a. Click Scale. The Scale Formation dialog box opens.
    b. Enter a value. For example, 2.0 doubles the size. 0.5 decreases the size by half.
    c. Click OK.
12. To move the formation as a group:
    a. Click Translate. The X Translation dialog box opens.
    b. Type a value for translation in the X plane.
    c. Click OK. The Y Translation dialog box opens.
    d. Type a value for translation in the Y plane.
    e. Click OK.

4.14.1. Formation Subgroups

Characters in a formation subgroup stick together, regardless of how the formation orients itself. If you specify use of subgroups, they are created automatically based on a group size and initial leader. For example, suppose a group has the following members:

- 0 - char0
- 1 - char1
- 2 - char2
- 3 - char3
- 4 - char4
- 5 - char5
- 6 - char6
- 7 - char7
- 8 - char8
- 9 - char9
- 10 - char10

If you specify a subgroup size of three, starting at index 2, DI-Guy Scenario creates the following subgroups:

- char2, char3, char4
- char5, char6, char7
- char8, char9, char10

To create subgroups:

1. In the AI Parameters group box, in the Subgroup Size box, specify the size of the subgroups.
2. In the Index of First Subgroup box, specify the index in the list of followers, of the leader of the first subgroup. DI-Guy Scenario divides the formation into subgroups of the specified size starting with the specified index.

4.14.2. Formation Roles

Having a role ensures that a character of a particular type fills a particular spot in a formation. For example, in a military application, you might always want a machine-gunner on the right flank. If it does not matter, choose "character" for the role. The formation code treats roles as a strong suggestion, but will ignore them when the actual characters in a formation do not have appropriate role settings.

To set up and assign roles:

1. Click the Roles button. The Select Available Roles dialog box opens.
2. Select the roles that you want to have available to followers.
3. Click OK.
4. Select the follower for whom you want to define a role.
5. Select a role from the Role list. The role is added to the name of the follower in the Follower Entries list.

4.15. Creating Groups

Groups are used to classify characters, vehicles and props. Decision logic can operate on groups, permitting more powerful functionality than can be accomplished on entities by themselves. For instance, you can create groups for friendly, hostile, and neutral characters and have characters decide how to respond to another entity based on what group or groups they belong to, for example, shoot hostiles, protect friendlies, observe neutrals. Groups facilitate applying decision logic to classes of entities, rather than to individual entities.

A character can be a member of more than one group. However, it is up to you to make sure that multiple membership makes sense. For example, it is possible for a character to be a member of both net_friendly and net_opposing, but this might lead to some nonsensical behavior.

To create a group:

1. Select the Character Objects:Group page (Figure 4-21).

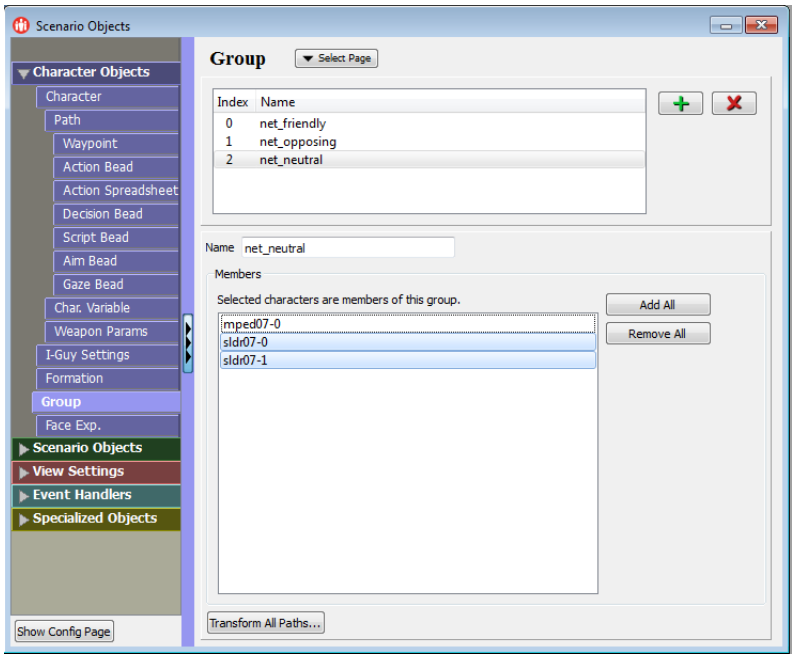


Figure 4-21. Group page

2. Click the Add button (+). A new group is added to the list.
3. Optionally, in the Name box, type a name for the group.
4. Select the group in the list.
5. In the Members group box select each character you want to include in that group. The highlighted characters are the members of the group.

4.16. Expressive Faces

The Character Objects:Face Expressions page is functional if you have purchased the Expressive Faces option. Expressive faces give characters the ability to show emotions, move their eyes, and move their mouths in synchrony with speech.

The list in the upper left corner contains already created facial expressions. New facial expressions can be created by mixing different amounts of the base expressions shown in the Parameters section. You can preview facial expressions on characters that have an expressive faces head.

➤ To preview a new face expression, click Preview on Current Character. It applies the selected expression to the selected person.

The preview button adds the specified expression to whatever face expression was already displayed on the selected character.

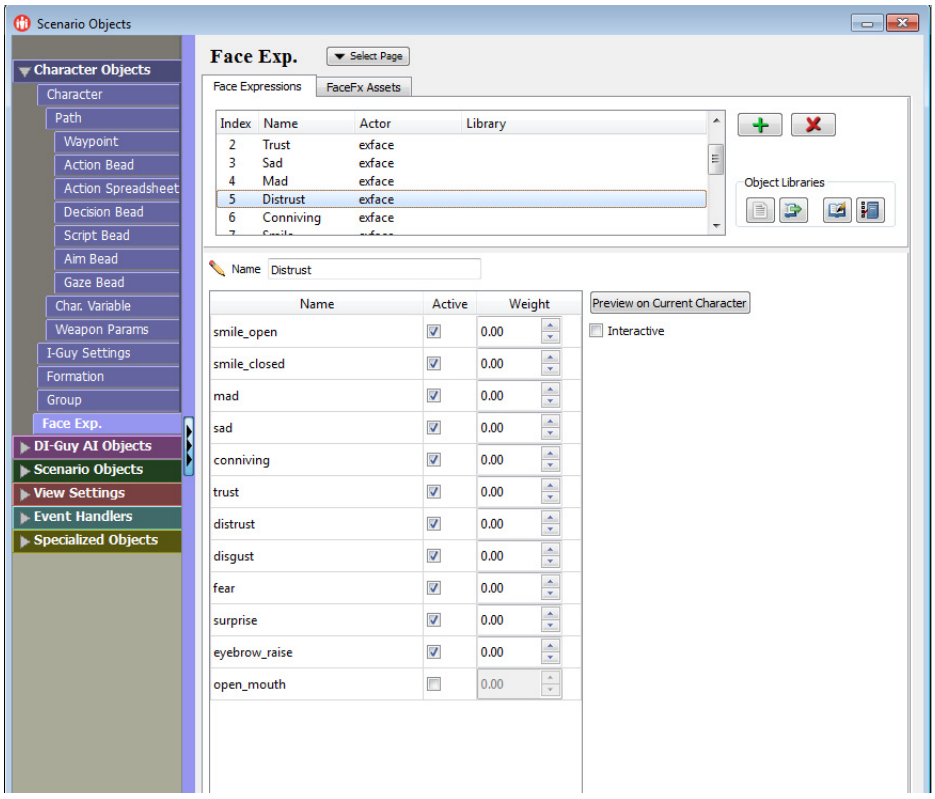


Figure 4-22. Face Exp. page

Once expressions are defined, they can be invoked using decision logic or script beads.

For more information, please see *DI-Guy Expressive Faces Users Guide*.

4.17. Character Labels and 3D View Labels

A label is a two-dimensional graphic that you can add to the rendered 3D scene. There are two basic types of labels available:

- Character labels. Character labels move with the character and float above their head. They can show the character name, detailed information about the character's current state, or user-specified text or graphics. There is extensive support both in the DI-Guy Scenario user interface and through the DI-Guy API for character labels. Figure 4-23 shows character labels with character state information.

Figure 4-23. Character label

- Screen and button labels. Screen labels stay at a fixed position on the screen. Screen labels can be read-only or they can serve as on-screen buttons to trigger events in a scenario. Support for these labels is exclusively through the API and is best implemented using scripts.

4.17.1. Enabling and Disabling Default Character Labels

You can enable and disable character labels globally or on a per character basis.

To enable character labels globally:

1. Select the View Settings:Visibility page (Figure 4-24).

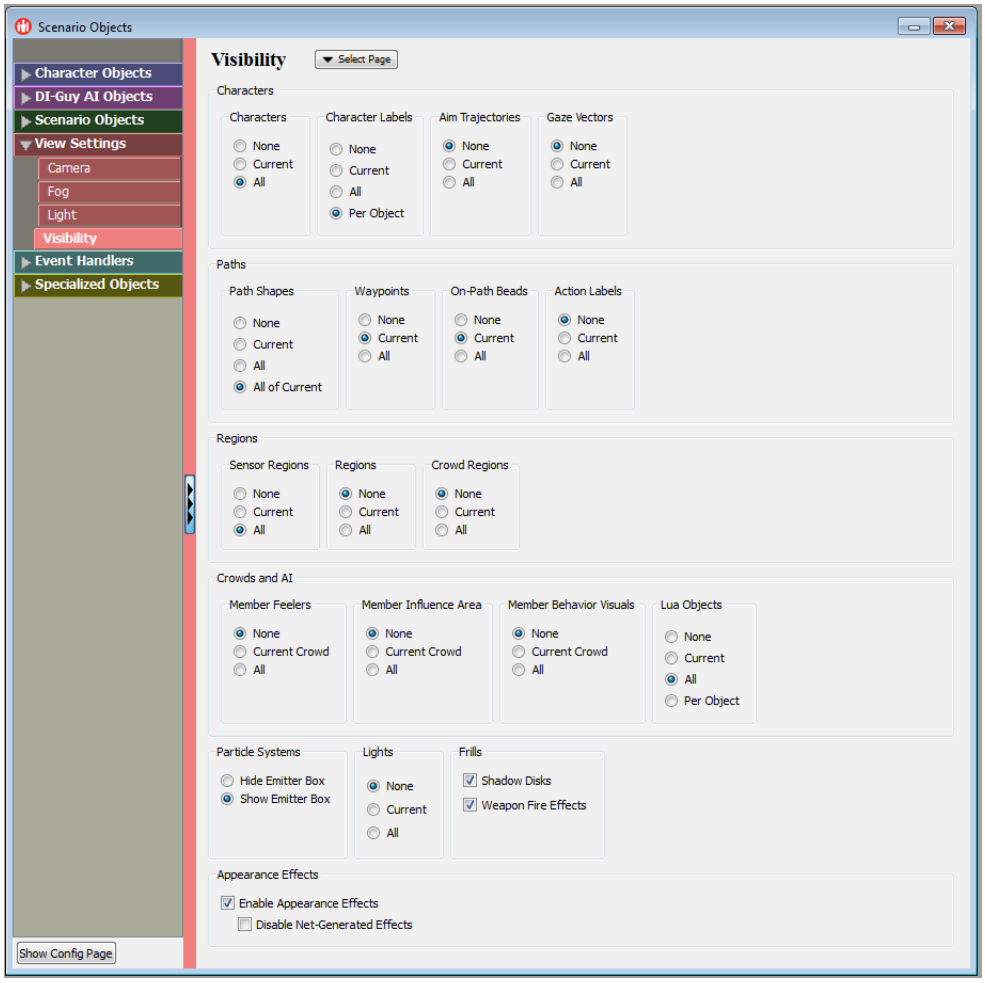


Figure 4-24. Visibility page

2. In the Character Labels group box, select one of the options:
   - None. Do not show any character labels, regardless of individual settings.
   - Current. Show a label for the currently selected character.
   - All. Show labels for all characters.
   - Per Object. Show labels for those characters that have enabled labels on the Character Objects:Character page.

To enable character labels for an individual character:

1. Select the character for which you want to enable a label.
2. Select the Character Objects:Character page (Figure 4-5).
3. Select the Label tab.
4. Select the Is Visible check box. (By default, the character text is the name of the character.)
5. To display state information, select the Name Label Shows Character State check box. (The character text is updated to match the state information.)

4.17.2. Editing the Style of Character Labels

You can edit the colors used for label text and background and can also add text shadows. You can edit labels in a simplified GUI and an advanced GUI.

To edit the style of character labels in the simplified GUI:

1. Select the character for whom you want to enable a label.
2. Select the Character Objects:Character page (Figure 4-5).
3. Select the Label tab.
4. Optionally, change the text color by changing the RGB values or by clicking on the color swatch and choosing a new color.
5. Optionally, change the text shadow color by changing the RGB values or by clicking on the color swatch and choosing a new color.
6. Optionally, change the background color by changing the RGB values or by clicking on the color swatch and choosing a new color.
7. Optionally, to display text shadow, select the Text Shadow check box.
8. Optionally change the Label Text. By default it displays the character name or state information.

Using the Advanced Label GUI

The advanced UI lets you control the label of a character using a property sheet.

To edit the style of character labels in the simplified GUI:

1. Select the character for whom you want to enable a label.
2. Select the Character Objects:Character page (Figure 4-5).
3. Select the Label tab.
4. Click Advanced UI. The page changes to show the property sheet (Figure 4-25).

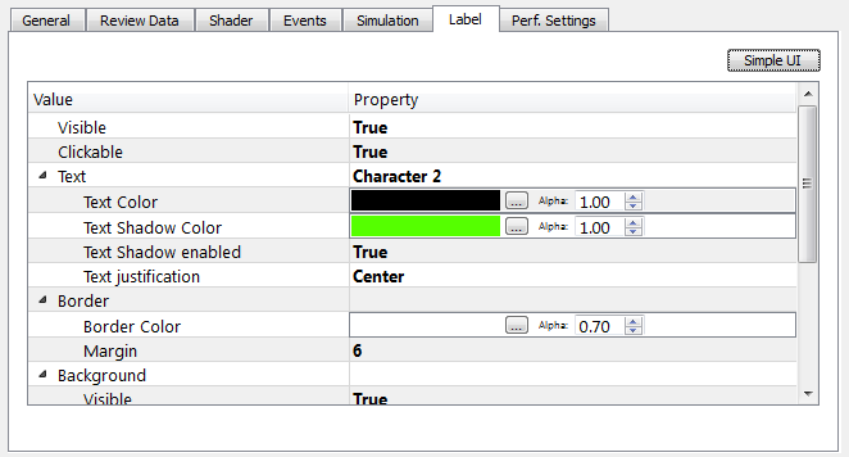


Figure 4-25. Character label Advanced UI

5. Change the properties as desired.

4.17.3. Labels and the DI-Guy API

The DI-Guy SDK has functions that let you manipulate labels directly from scripts or plug-ins. For details, please see *DI-Guy SDK Developers Guide*, particularly the documentation regarding the *diguyViewLabel* class. This class lets you set up labels and all the details inside of them including:

- Text
- Colors
- Justifications
- Position
- Background
- Border
- Bar graphs
- Interactivity and callbacks.

4.17.4. Creating Screen and Button Labels

You can write scripts that create labels in the 3D View. Labels can be informational only or can be active areas that trigger actions in the scenario.

The sample scenario AI_Crowd_Blitz_Demo is a good example of creating interactive labels (buttons). It also is a good demonstration of creating labels using the API instead of the DI-Guy Scenario GUI. (You can load this scenario and look at its scripts even if you do not have a license for DI-Guy AI.)

1. Open the AI_Crowd_Blitz_Demo scenario. This scenario has several labels on the left side of the 3D View (Figure 4-26).

Figure 4-26. Screen labels

2. Select the Event Handlers:Script page.
3. In the list of scripts, select initialize_labels. The script is displayed in the script window (Figure 4-27).

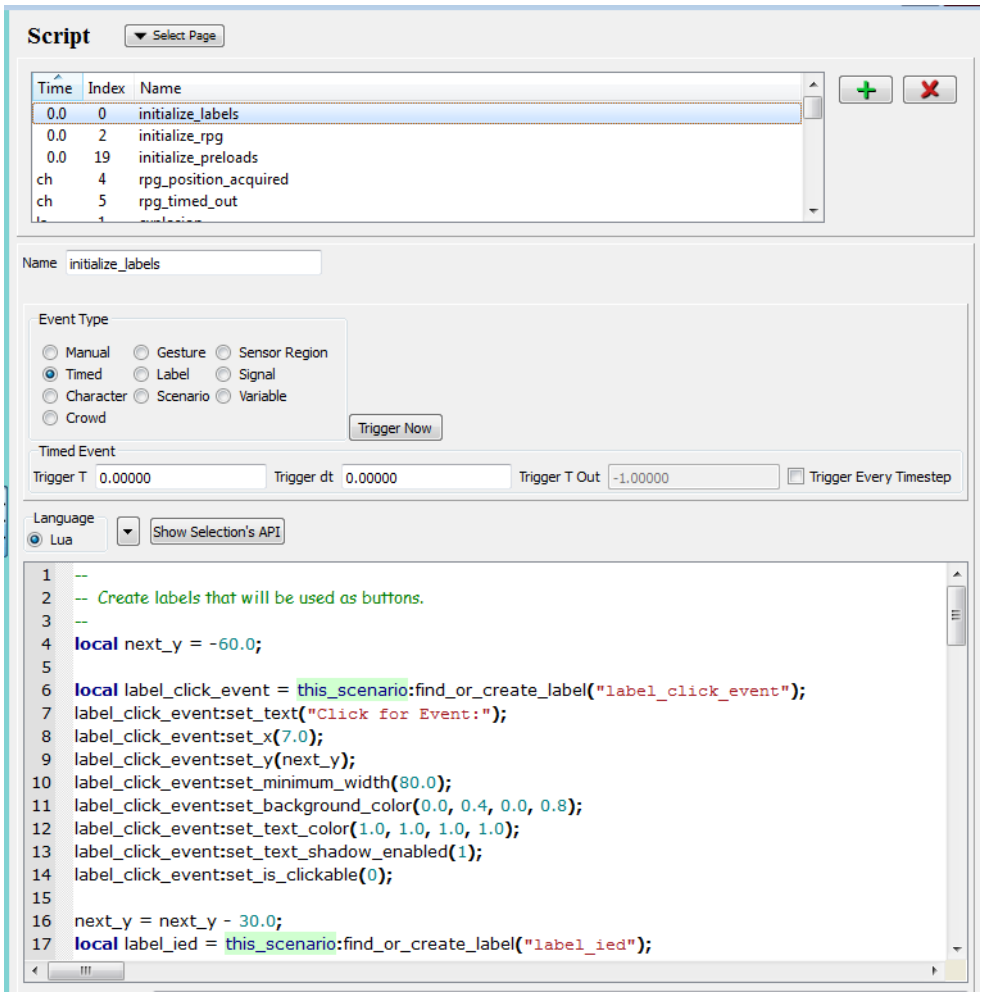


Figure 4-27. Script page

The script initializes the labels. The code for the Click for Event and IED Explosion labels are as follows:

```
label_click_event =
  this_scenario:find_or_create_label("label_click_event");
label_click_event:set_text("Click for Event:");
label_click_event:set_x(7.0);
label_click_event:set_y(next_y);
label_click_event:set_minimum_width(80.0);
label_click_event:set_background_color(0.0, 0.4, 0.0, 0.8);
label_click_event:set_text_color(1.0, 1.0, 1.0, 1.0);
label_click_event:set_text_shadow_enabled(1);
label_click_event:set_is_clickable(0);

next_y = next_y - 30.0;
local label_ied = this_scenario:find_or_create_label("label_ied");
label_ied:set_text("IED Explosion");
label_ied:set_x(15.0);
label_ied:set_y(next_y);
```

```
label_ied:set_minimum_width(110.0);
label_ied:set_background_color(0.0, 0.0, 0.4, 0.6);
label_ied:set_text_color(1.0, 1.0, 1.0, 1.0);
label_ied:set_text_shadow_enabled(1);
label_ied:set_is_clickable(1);
```

The code constructs the labels and then sets their text, position, width, and colors.

The first label is just a label. If you click it, nothing happens. The second label, IED Explosion, is interactive, so it calls the set_is_clickable() function with a value of one. To make it actually do something when clicked, you have to map callbacks to events (actually other scripts).

Callbacks are API functions that are invoked when certain things happen. The relevant events for a clickable button are getting clicked and having the mouse button released. These are handled through the callbacks CALLBACK_ID_USER_SELECTED and CALLBACK_ID_USER_UNSELECTED. You have to tell DI-Guy Scenario what to do when these callbacks are invoked. You do that by mapping these callbacks to other scripts.

This scenario has additional scripts for handling button clicks called `pressed_red` and `released_red`. These just change the color of the button when you click it. There is another script, called `explosion`, that controls what we want to happen when this button is clicked.

The callback script invoked by the IED Explosion button is `explosion`, also listed in the Scripts page. Note that it is a script of type `label`. It simply calls some functions defined in other scripts.

1. In the Script list, select `explosion`. Notice that in the Event Type group box it is of type `label`.
2. Select the `pressed_red` and `released_red` scripts. They are also type `label`.

You now have the background knowledge of the scripts to understand how they are hooked together.

1. Select the Event Handlers:Event Mapper page (Figure 4-28).

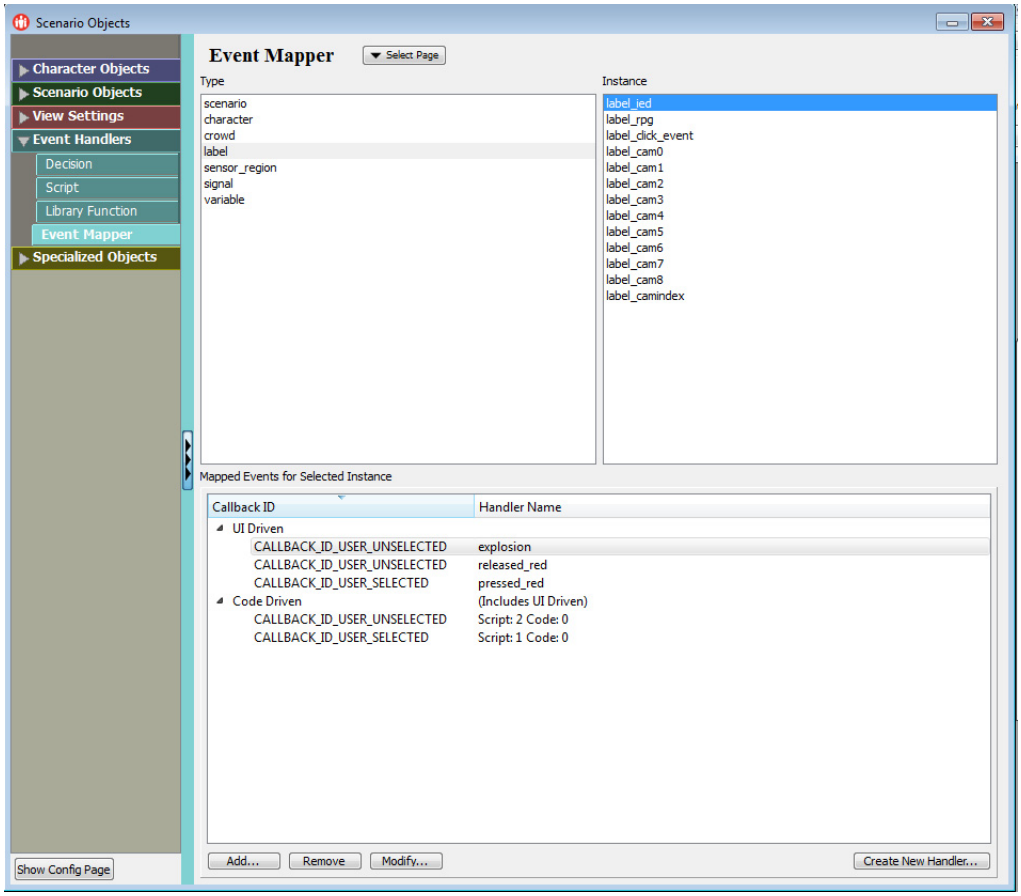


Figure 4-28. Event Mapper page

2. In the Type list, select `label`. All labels created by the scripts are listed in the Instance list.
3. Select `label_ied`, the ID for the IED Explosion label. Go back and check the code in the `inialize_labels` script to see where the ID was specified when the label was created.

4. In the Mapped Events for Selected Instance window, look at the callbacks listed for this instance. The callback that we discussed in an earlier paragraph are listed along with the scripts we discussed, as follows:
   - CALLBACK_ID_USER_SELECTED. When the button is selected (clicked), the `pressed_red` script runs and the button turns red.
   - CALLBACK_ID_USER_UNSELECTED. When the mouse button is released, that is when the button is now unselected, the `released_red` script runs to change the color of the button back and the `explosion` script runs to set off an explosion.

Although computer users experience clicking a button as being one action, a computer treats it as two – a click and release.

4.18. Configuring Chains

Some DI-Guy characters have job-related props, such as chains or hoses. You can configure the attributes of these objects.

To configure chains:

1. Select the Specialized Objects:Chain Settings page (Figure 4-29).

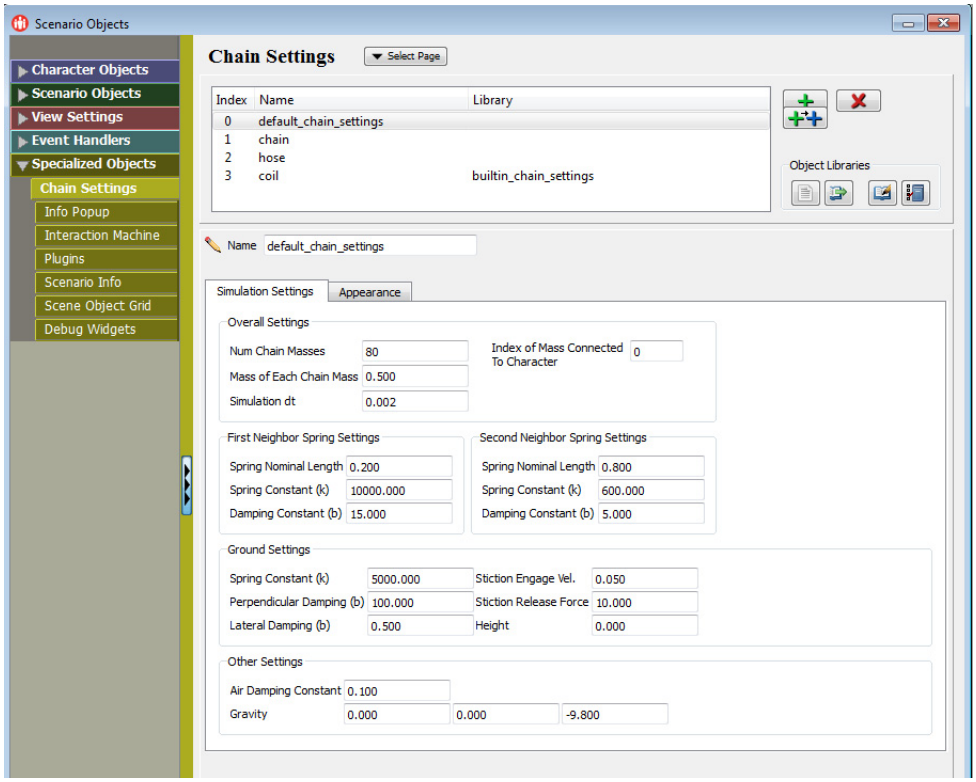


Figure 4-29. Chain Settings page

2. Click the Add button (+). A new chain is added to the list.
3. Optionally, change the name of the chain.
4. Edit the parameters on the two tabs as desired.

5. Paths and Waypoints

This chapter explains how to create and manage character-specific paths and waypoints.

Creating and Managing Paths 5-2
- Creating a Path 5-2
- Selecting a Path 5-4
- Hiding Path Elements 5-4
- Path Draw Style 5-4
- Dead Space 5-5

Creating and Editing Waypoints 5-5
- Creating Waypoints in the 3D View Window 5-6
- Creating Waypoints Using the Waypoint Page 5-6
- Selecting Waypoints 5-7
- Editing Waypoints in the 3D View 5-8
- Editing Waypoints on the Waypoint Page 5-9
- Moving a Group of Waypoints 5-10
- Ground Clamping Waypoints 5-10

Using Branched Paths 5-11

5.1. Creating and Managing Paths

Characters can travel along paths as they move through the scenario.

The shape of a path is a spline curve defined by one or more waypoints. Each waypoint has location, slope, and weight information that determines the shape of the spline. Paths contain not just topological information as derived from the waypoints, but also behavior information as defined by its action beads, decision beads, script beads, aim beads, and gaze beads. It is helpful to think of these beads literally as beads that can be slid along the path.

Paths are tied to specific characters. They do not exist independently of them. You can also create path shapes, which are independent of characters. For information about path shapes, please see "Creating Path Shapes," on page 7-11.

Paths are represented in the 3D View by thin, curved lines. The current path is white. All other paths are blue.

5.1.1. Creating a Path

You can create paths dynamically in the 3D View and you can create them on the Path page. When you create a new path, the first waypoint in the path is located at the active waypoint in the character's currently selected path. This makes it look like you have just extended the existing path. However, the new path is independent and you can move the starting waypoint to another location if you want to.

Each new path is created with an index one greater than the currently selected path. All path indices following the current path are increased by one when the new path is created. The new path becomes the current path.

To create a path on the Path page:

1. Select the character for whom you want to create a path.
2. Select the Character Objects:Path page (Figure 5-1).

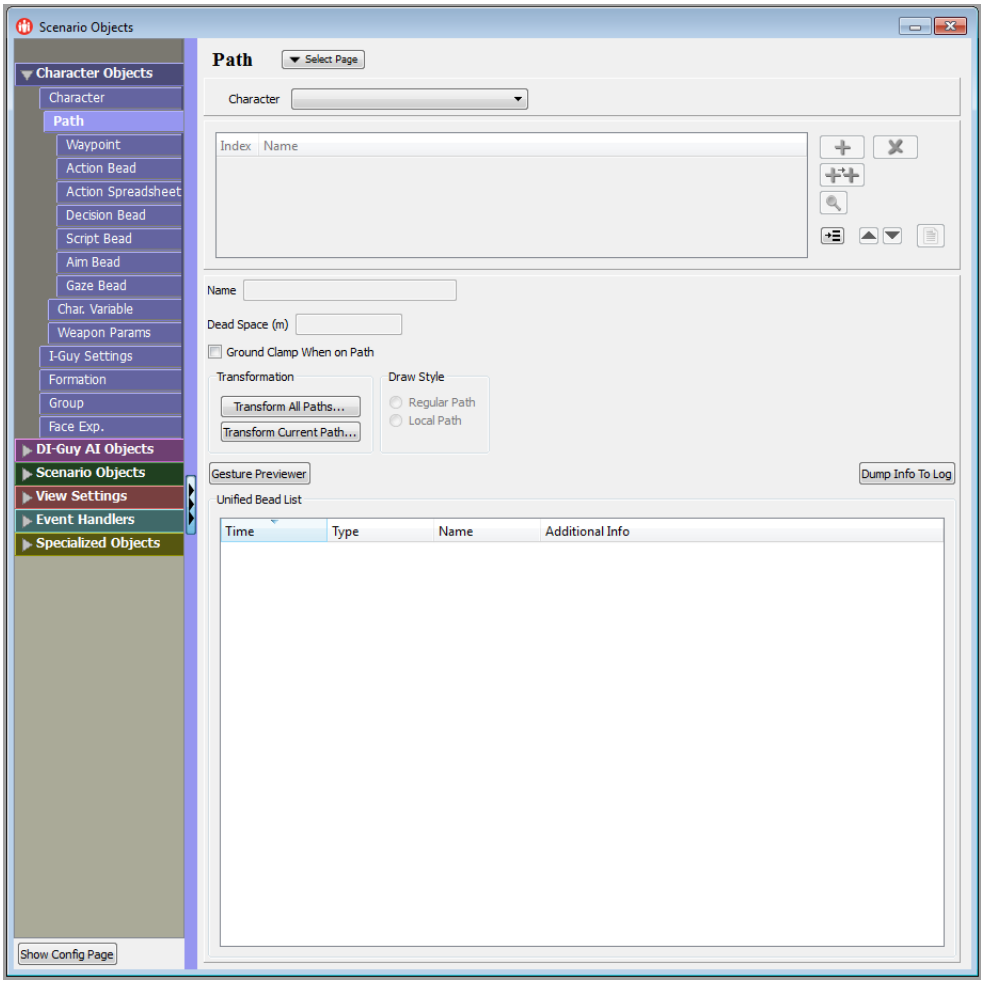


Figure 5-1. Path page

3. Click the Add button (+). A path is added to the list. It is given a default name. The first waypoint in the path is the waypoint selected in the previously active character path. The new path has two waypoints.
4. Optionally, type a new name in the Name box. The name must be unique for this character.
5. Optionally edit the waypoints in the path. For information about editing waypoints, please see "Editing Waypoints in the 3D View," on page 5-8 and "Editing Waypoints on the Waypoint Page," on page 5-9.

To create a path in the 3D View:

1. Select the character for which you want to create a path.
2. In the Input Mode window, click Path Insert.
3. In the 3D View, click to set the waypoints of the path.
4. When you create the last waypoint, exit Path Insert mode by selecting a different mode in the Input Mode window or by moving focus into another window or page.

5.1.2. Selecting a Path

To select a path:

1. Select the character whose path you want to select.
2. Select the Character Objects:Path page (Figure 5-1).
3. Click a path on the list.

5.1.3. Hiding Path Elements

You can hide paths in the scenario.

To hide path elements:

1. Select the View Settings:Visibility page.
2. In the Paths group box, select one of the following options for path shapes, waypoints, beads, and action labels:
   - None. No objects of this type are visible.
   - Current. Only the objects of this type for the current path are visible.
   - All. All objects of this type for all characters are visible.
   - All of Current. All paths of the current character.

5.1.4. Path Draw Style

Paths can be categorized as regular or local. Regular paths are created at a particular place in the terrain. Local paths are created so that they can be moved anyplace in the terrain.

5.1.5. Dead Space

Each action uses a length of the path. Dead space is the length of the unused portion of the path. This is a read-only value used to help understand the relationship between the action beads on the path and the overall path. A negative dead space value indicates that too many motions have been specified and will cause the character to jump to the beginning of the path. A large positive value will cause a jump to the next path when the character proceeds to another path.

5.2. Creating and Editing Waypoints

Waypoints define the shape and location of paths. You can move a path, make it turn, or make it smoother by putting waypoints in the right places. The character starts at the first waypoint and travels forward along the path over time.

Waypoints are represented in the 3D View window by green circles. The tangent lines of the current waypoint are red. All other tangent lines are green.

Each waypoint has a slope adjuster that controls the tangent line of the path where it passes through the waypoint. You can change the orientation and strength of the slope adjuster to influence the orientation, curvature, and smoothness of the path.

Figure 5-2. Waypoint

5.2.1. Creating Waypoints in the 3D View Window

It is easier to create and place new waypoints in the 3D View window than on the Character Objects:Waypoint page. New waypoints are added to a path right after the currently selected waypoint. If you have selected waypoint two and insert a new waypoint, then the new one will be waypoint three and the index of all subsequent waypoints will be increased by one.

To insert a waypoint:

1. Select the path you want to edit.
2. In the Input Mode window, click Path Edit (or press **x**).
3. Select a waypoint. (The new one will go right after the one you select. If no waypoint is selected, the new one will be the first on the path.)
4. Press **Ctrl** and click on the path where you want to insert the waypoint.
5. Optionally, drag the slope-adjuster to change the orientation and smoothness of the path.
6. Repeat steps four and five to create additional waypoints. New waypoints are added after the most recently created one.

5.2.2. Creating Waypoints Using the Waypoint Page

You can create waypoints on the Character Objects:Waypoint page.

The default position of a new waypoint created on the Waypoint page depends on whether it will be the first waypoint (index 1), the last waypoint, or somewhere in between. Table 5-1 describes where new waypoints are located relative to existing ones.

Table 5-1: Default waypoint position

Location	Position
First	The new waypoint is located at the origin.
Last	The new waypoint is placed 4 meters beyond the selected waypoint, continuing in the direction of the path.
Intermediate	The new waypoint is placed along the existing path halfway between the two neighboring waypoints.

1. Select the character for which you want to add waypoints to a path.
2. Select the Character Objects:Waypoint page (Figure 5-3).

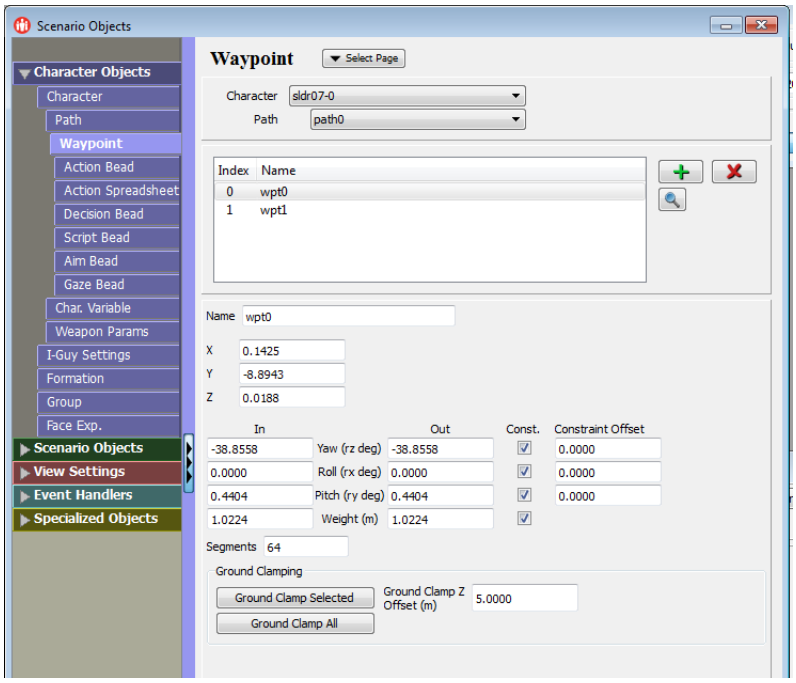


Figure 5-3. Waypoint page

3. Select the waypoint that precedes where you want to add a new one. For example, to create a waypoint between existing points 2 and 3, set the current waypoint index to 2.
4. Click the Add button (+). A new waypoint is added using a default location.
5. Optionally, move the waypoint.

5.2.3. Selecting Waypoints

You can select waypoints in the 3D View or the Character Objects:Waypoint page.

To select a waypoint in the 3D View:

1. Select the path whose waypoint you want to edit.
2. In the Input Mode window, select Path Edit mode (or press **x**).
3. Click the waypoint you want to edit. The selected waypoint's slope-adjuster turns red.

➤ To select a waypoint on the Character Objects:Waypoint page, select a waypoint in the list of waypoints.

5.2.4. Editing Waypoints in the 3D View

Editing a waypoint in the 3D view is the easiest way to change its position and orientation. Other editorial changes are available on the Waypoint page.

To edit a waypoint in the 3D View:

1. Select the path whose waypoint you want to edit.
2. In the Input Mode window, select Path Edit mode (or press **x**).
3. Click the waypoint you want to edit. The selected waypoint's slope-adjuster turns red.
4. Drag the waypoint to change its position.
5. Drag the slope-adjuster to change the orientation of the waypoint. The slope-adjuster is sensitive to the mouse at its endpoints.

5.2.5. Editing Waypoints on the Waypoint Page

You can edit a waypoint's name, location, orientation, and weight.

To edit a waypoint:

1. Select the character for which you want to edit waypoints.
2. Select the Character Objects:Waypoint page (Figure 5-3).
3. In the list of waypoints, select the waypoint you want to edit.
4. Edit values as follows:
   - Name. Type a name in the Name box. It must be unique for this character.
   - Location. Specify coordinates in the X, Y, and Z boxes.
   - Orientation. Enter values for Yaw, Pitch, and Roll in the appropriate boxes. If the In and Out values differ, the path will be discontinuous at the waypoint. If these values are the same, the path will stay smooth through the waypoint. Table 5-2 describes the meaning of the different values.

Table 5-2: Orientation values

Coordinate	Meaning
Yaw	Rotation around the z (up) axis.
Roll	Rotation around the x (forward) axis.
Pitch	Rotation around the y (side) axis.

   - Weight. The weight determines how much influence the waypoint has on the shape of the path. In and Out values have the same effect as for orientation.
   - Smoothness. Waypoints are created with the Constraint checkboxes selected. If you want to vary the In and Out values separately for any of the orientations or the weight, clear the corresponding checkboxes. You can then manipulate the path to smooth it out.

5.2.6. Moving a Group of Waypoints

You can move several waypoints at one time in the 3D View window.

To move a group of waypoints:

1. Select the path whose waypoints you want to edit.
2. In the Input Mode window, select Path Edit mode (or press **x**).
3. In the 3D View window click the first waypoint to be moved.
4. **Shift**+click the last waypoint to be moved. This selects all waypoints between the first and last waypoints. All selected waypoint slope-adjusters turn red to indicate selection of the waypoint.
5. **Shift**+drag the mouse to move the waypoints.

When you drag a group of waypoints, the altitude of the group may be recalculated based on the first waypoint of the group. If the waypoints are located on uneven terrain, some waypoints may penetrate the ground or float above the ground when the group is moved. Re-apply ground clamping or edit these waypoints manually if this is an issue.

5.2.7. Ground Clamping Waypoints

If you have waypoints floating above or below the surface, DI-Guy Scenario can adjust the waypoints so that they are at the same altitude as the terrain. This is called ground clamping. When you ground clamp waypoints, you can adjust the altitude used by the ground clamping algorithm.

To ground clamp a waypoint:

1. On the Waypoint page, select the waypoints that you want to ground clamp.
2. Click Ground Clamp Selected.
3. To adjust the altitude of the ground clamping algorithm, type a value in the Ground Clamp Z Offset box.

➤ To ground clamp all waypoints, on the Waypoint page, click Ground Clamp All.

5.3. Using Branched Paths

Suppose you want a character to decide which way to go in a scenario, based on a decision. While using DI-Guy Scenario AI to create such an autonomous character is one option, this behavior can also be done with branched paths. Once you have created a character you can give it additional paths, and then use decision logic to switch the character among them.

Branched pathing can also work using local paths. If a new local path starts at the origin and moves in the x-direction, the push_local_path() function call will smoothly start playing the new path from the end of the current path.

Once you have created the network of branched paths, you can assign decision logic that determines which branch to take. The push_path action causes the character to move from the current path to the path named in the argument to the action. Once on the new path, the character executes the actions and decision beads assigned to that path. For an example of a scenario that uses branched paths and decision logic in conjunction with sensor regions, see *Decision_pedestrians.dss*, which is part of the DI-Guy Scenario installation.

Functions like force_path() are also useful for this style of branched pathing.

6. Character-Level Actions

This chapter explains how to create and manage character-level actions, scripts, and other decision logic.

Creating and Managing Action Beads 6-2
- Creating Action Beads 6-3
- Editing Action Beads 6-5
- Moving Action Beads 6-7
- Pointing the Camera at an Action 6-7
- Non-Forward Actions and Companion Waypoints 6-8

The Action Spreadsheet 6-10
- Inserting an Action Using the Action Spreadsheet 6-11
- Editing an Action in the Action Spreadsheet 6-11
- Deleting an Action in the Action Spreadsheet 6-12
- Stretch and Shrink 6-12

Decision Beads 6-13
- Creating a Decision Bead 6-14
- Creating Character-Level Decision Logic 6-16
- Converting Decision Beads to Script Beads 6-19

Writing Script Beads 6-20

Creating Aim Beads 6-21

Creating Gaze Beads 6-23

Gestures 6-25
- Head Nods and Shakes 6-25
- Generic Gestures 6-25
- Previewing Gestures 6-26

6.1. Creating and Managing Action Beads

Each action bead along a path tells the character to perform a new action. You can edit actions in the 3D View, on the Character Objects:Action Spreadsheet page, and on the Character Objects:Action page. It is usually easier to edit actions by manipulating them graphically using the Action Edit and Action Insert modes, or by changing values on the Action Spreadsheet.

The Action page provides access to the action beads of the current path. If there is no current path, you cannot create, modify, or delete action beads.

Action beads are represented in the 3D View window by yellow gear-shaped objects on the path. The current action bead spins on its axis. Others are stationery. The last action bead is a red octagonal symbol, similar to a stop sign.

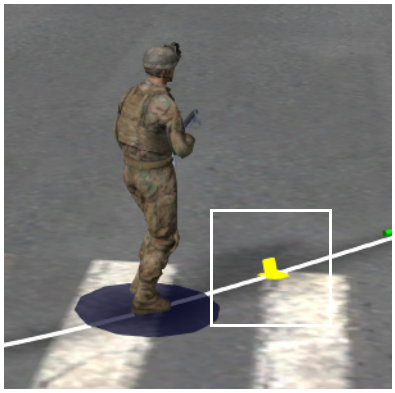


Figure 6-1. Action bead

There are two basic types of actions – stationary and traveling. Stationary actions, such as standing, sitting, and waving, do not cause the person to travel along the path. Traveling actions, such as walking, jogging, and crawling, cause the person to travel along the path.

6.1.1. Creating Action Beads

You can create action beads in the 3D View or on the Action Bead page.

To create an action bead in the 3D View:

1. Select the character for whom you want to add an action.
2. In the Input Mode window, click Action Insert. The right side of the window displays new options (Figure 6-2).

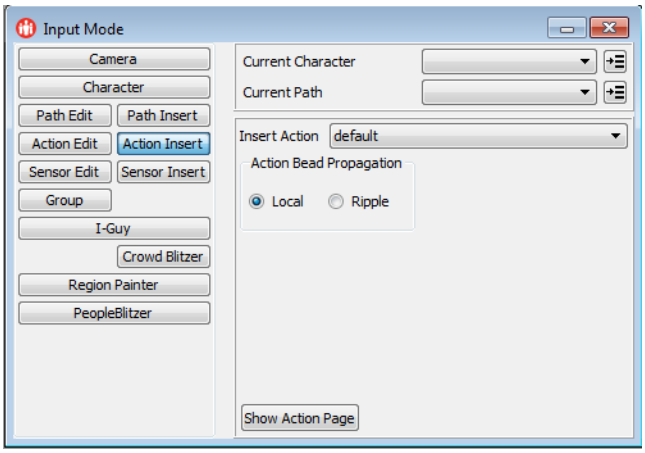


Figure 6-2. Input Mode window, Action Insert

3. In the Insert Action list, select the action you want to insert.
4. Select the Action Bead Propagation options you want to use. For details, please see "How Action Bead Insertion or Movement Affects Other Action Beads," on page 6-5.
5. In the 3D View, click on the path where you want to insert the action. The action bead is added to the path. It is now the active action bead. An action bead is added to the list on the Character Objects:Action Bead page.

To create an action bead on the Action Bead page:

1. Select the character or vehicle whose action you want to edit.
2. Select the Character Objects:Action Bead page (Figure 6-3).

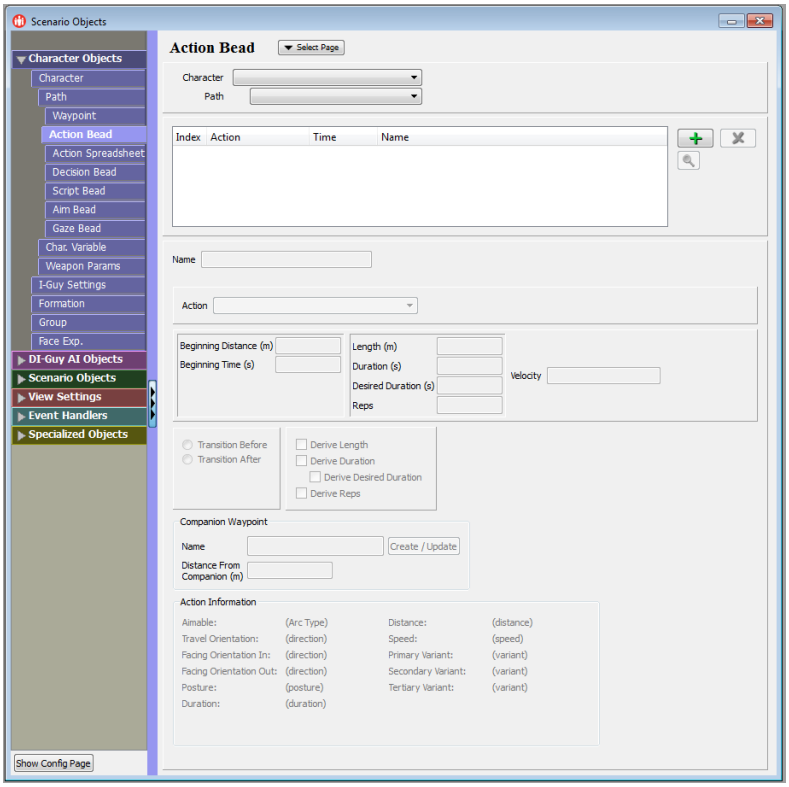


Figure 6-3. Action Bead page

3. Select the action that occurs before where you want the new action to occur. For example, to create an action between actions two and three, select action two.
4. Click Create. The new action bead becomes the current one. The indices of the subsequent action beads are incremented to make room for the new bead.
5. Optionally, change the default name.
6. In the Action list, select the action to associate with this action bead.
7. In the Beginning Distance box, specify the distance along the path at which the action should start.
8. In the Beginning Time box, specify the time at which the action should start.
9. In the Length box, specify the distance that is covered by this action.

Editing length or duration for an action bead will cause a ripple effect on all subsequent action beads in the path. See the "The Action Spreadsheet," on page 6-10 for more information on this effect.

10. In the Duration box, type the amount of time required for the action.
11. In the Desired Duration box, type the amount of time you would like the action to take.

How Action Bead Insertion or Movement Affects Other Action Beads

When an action is inserted into the middle of a sequence of actions, (or when a length or duration constraint is changed), the effect of that action on subsequent actions is governed by the setting of the action bead propagation option buttons, as follows:

- Local Mode. When you place or move an action bead, none of the surrounding action beads move.
- Ripple Mode. The spacing between actions is preserved wherever possible. When a new action is added, all subsequent actions moved down the path by the displacement of the added action. When an action is deleted, all subsequent actions move toward the beginning of the path by the amount of the deleted action. If you drag an action bead to a new position, subsequent beads move down the path.

6.1.2. Editing Action Beads

You can edit the parameters of an action bead. You can also move action beads in the 3D View.

To edit an action bead on the Action Bead page:

1. Select the character or vehicle whose action bead you want to edit.
2. Select the Character Objects:Action Bead page (Figure 6-3).
3. In the list of action beads, select the one you want to edit.
4. Change any of the attributes. They are described in "Action Bead Attributes," on page 6-6.

Action Bead Attributes

In addition to the attributes discussed elsewhere, you can set the following values for action beads:

- Reps (repetitions). Every action has a natural distance and time associated with it. Each time an action is specified, the DI-Guy Scenario motion engine calculates how many repetitions of the action will most closely fit in the available distance (traveling) or duration (stationary). After determining the best fit, DI-Guy Scenario stretches or shrinks the action to make it fit exactly.
- Transition Before/Transition After. When one action is complete and another action is starting, a transition motion occurs to smoothly and naturally have the character change its motion. DI-Guy Scenario automatically performs this transition for you. The distance or duration of these transitions are calculated when determining repetitions, length, and duration of action beads on a path. Advanced users may want to change the default values when trying to achieve exacting performances from their characters. The average user should not need to change the default settings.
- Derive Length. DI-Guy Scenario derives the displacement for the action based on the time you specify or the number of repetitions you specify.
- Derive Duration. DI-Guy Scenario derives the time required for the action, based on the displacement you specify or the number of repetitions you specify.
- Derive Desired Duration. The desired time is set equal to the actual duration of the action. When this box is cleared, the desired time is used as a guide to calculate the duration of the action. DI-Guy Scenario selects an integer number of repetitions of the specified motion that will most closely match the desired time.
- Derive Reps. DI-Guy Scenario automatically calculates the number of times the action is looped to best fill in the available space or time.

6.1.3. Moving Action Beads

You can move action beads in the 3D View. When you move an action bead, other action beads are affected based on the Action Bead Propagation setting. For details, please see "How Action Bead Insertion or Movement Affects Other Action Beads," on page 6-5.

If you have trouble placing action beads by hand, you can set their beginning distance on the Action Bead page.

To move an action bead:

1. Select the character whose action beads you want to edit.
2. In the Input Mode window, click Action Edit.
3. Select the desired option in the Action Bead Propagation group box.
4. In the 3D View, drag the action bead to a new location on its path. You cannot drag it directly past another action bead. If you drag it up to an action bead, it stacks on top of it.
5. To drag an action bead out of a stack, click to the side and drag.

6.1.4. Pointing the Camera at an Action

You can quickly point the camera at the selected action.

To point the camera at the selected action:

1. Select the Character Objects:Action Bead page (Figure 6-3).
2. In the list of actions, select the one you want to view.
3. Click the View button ().

6.1.5. Non-Forward Actions and Companion Waypoints

Companion waypoints are waypoints that are associated with action beads that require people to face in a different direction from that in which they are traveling.

People usually face in the same direction they travel. For instance, a person walking forward faces forward as they go, keeping their face, chest, and pelvis aligned with their forward motion. Similar rules generally apply for strolling, jogging and running. These are all called forward actions.

Sometime a character travels in a different direction from their facing direction. Stepping sideways is an example, as is stepping backward. These are called non-forward actions.

DI-Guy Scenario provides support for using non-forward actions. To create a smooth transition from a forward action to non-forward action, DI-Guy Scenario provides a special waypoint called a companion waypoint and a process for creating it. Companion waypoints break the path at the action bead, reorient the path according to the travel direction of the action bead, and reorient the character with respect to the path. This is all done automatically when the companion waypoint is created.

To create a companion waypoint:

1. Select the Character Objects:Action Bead page (Figure 6-3).
2. Select the action for which you want to create a companion waypoint.
3. In the Companion Waypoint group box, click Create/Update. A companion waypoint is added to the scenario with a default name.
4. Optionally change the name of the companion waypoint.
5. Optionally, specify an offset. An offset is used to smooth out transitions between actions. This option is not used often.

Updating a Companion Waypoint

If you change the action associated with an action bead that has a companion waypoint, you should update the waypoint. This ensures that the slope adjuster bar is oriented properly.

To update a companion waypoint:

1. Select the Character Objects:Action Bead page (Figure 6-3).
2. Select the action for which you want to update a companion waypoint.
3. In the Companion Waypoint group box, click Create/Update.

Companion Waypoint Tutorial

This tutorial demonstrates the creation and effect of a companion waypoint.

1. Blitz a character of character type controller into the scene, drawing a path as part of the procedure. (The controller has non-forward traveling actions.)
2. In the Input Mode window, click Action Insert.
3. In the Insert Action list, select action bead `side_step_L`.
4. Insert the action at the center of the path.
5. Select the Character Objects:Action Bead page (Figure 6-3).
6. Select the `side_step_L` action.
7. In the Companion Waypoint box, click the Create/Update button. A new waypoint is placed at the path at the location of the selected action bead. The new waypoint is attached to the action bead and will move with it as you change the shape of the path. It is called a companion waypoint.
8. In the Time Control window, click Reset.
9. Play the scenario. The character walks to the waypoint, then turns sideways and walks with a side step.
10. If you want the character to resume forward travel:
    a. Insert an action bead for forward travel later on the path
    b. Create an additional companion waypoint for it.
    c. Click reset.
    d. Play the scenario to observe the resulting behavior.

You may notice that the companion waypoint looks slightly different from other waypoints. Its slope adjuster bar (the long post) does not pass straight through the waypoint. It is angled (in our example, 90 degrees) to reflect the desired change in input and output facing directions.

After you complete the tutorial, try changing the `side_step_L` action to `side_step_R` or `step_Back_L`. Because you have changed the action bead associated with the companion waypoint, you need to update it. On the Action Bead page, click Create/Update. The waypoint will be properly updated, taking into account the change in orientation that will occur at the action bead. You will probably want to move the final waypoint for a completely satisfying overall motion.

6.2. The Action Spreadsheet

The Character Objects:Action Spreadsheet page (Figure 6-4) displays detailed information about the sequence of action beads on the current path. Actions can be added, deleted, and changed, and numerical data can be edited. This page gives access to timing information for the current character and information about the relationship between displacement and action along the path. The Action Spreadsheet is an excellent place to refine the behavior of a character after creating it in the 3D View. It is particularly helpful when working on complex sequences that involve both traveling and stationary actions.

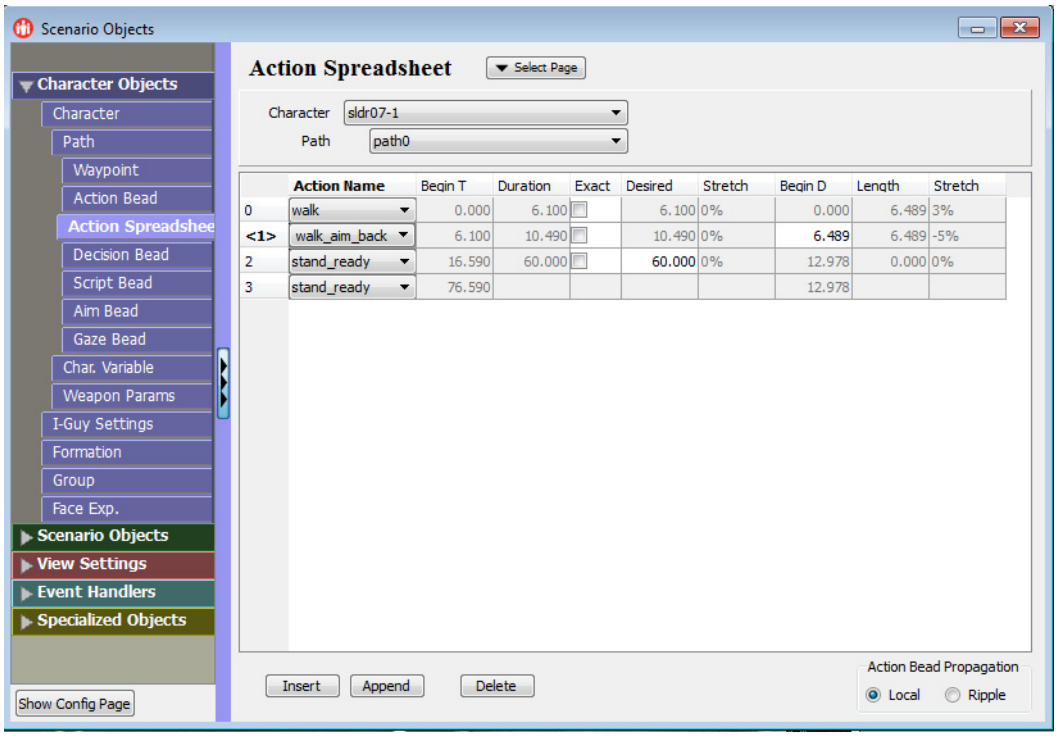


Figure 6-4. Action Spreadsheet

Each row in the spreadsheet contains the data for one action a character will perform on its path. The starting position and displacement (Length) are distances measured along the path, referenced to the first waypoint. The column definitions are as follows:

- Action Name. Name of the action.
- Begin T. The absolute time this action begins.
- Duration. The amount of time the action requires. You set this for stationary actions and when using Exact.
- Exact. For advanced users, allows you to constrain motion both in displacement and duration.

- Desired. The amount of time usually required to perform the action (or repeated loops of the action).
- Stretch. The amount that an action is lengthened to try to fit complete actions into the available time. Stretching actions may result in foot sliding.
- Begin D. The distance from the beginning of the path where the action is located.
- Length. The distance to travel to the next action or waypoint. It should be zero for stationary actions.
- Stretch. Amount the motion was compressed or elongated in travel distance to achieve the length constraint.

6.2.1. Inserting an Action Using the Action Spreadsheet

To insert an action using the Action Spreadsheet:

1. On the Input Mode page, click Action Insert and select the action to be inserted.
2. On the Character Objects:Action Spreadsheet page (Figure 6-4), click the index of the action that is adjacent to where you want to add the new action. (The index number of the selected action is wrapped in angle brackets.)
3. To add the new action above the selected action, click Insert.

   To add the new action below the selected action, click Append.

6.2.2. Editing an Action in the Action Spreadsheet

You can change an action and edit some of its values in the Action Spreadsheet. Values that are editable have a white background. Read-only values have a gray background.

To edit an action in the Action Spreadsheet:

1. On the Character Objects:Action Spreadsheet page (Figure 6-4), click the index of the action you want to edit. (The index number of the selected action is wrapped in angle brackets.)
2. To change the action, select a different action from the Action Name list.
3. Change editable values as desired.
4. Select an option from the Action Bead Propagation group box.

Specifying Exact Timing for an Action

You can specify the exact duration for an action. The duration of each cycle is stretched or shrunk so the overall duration exactly meets your specification. For traveling actions, this procedure slows down or speeds up the travel, without changing the number of steps. The time stretch field of the Action Spreadsheet shows how much the timing was adjusted to accommodate your setting. Remember that the exact duration includes the transition time to get from the previous action to the current one.

To specify an exact duration for an action:

1. On the Character Objects:Action Spreadsheet page (Figure 6-4), click the Exact check box for the action you want to edit. The Duration value becomes editable. Other values might become editable.
2. Specify the duration for the action.

6.2.3. Deleting an Action in the Action Spreadsheet

To delete an action using the Action Spreadsheet:

1. On the Character Objects:Action Spreadsheet page (Figure 6-4), click the index of the action you want to delete. (The index number of the selected action is wrapped in angle brackets.)
2. Click Delete.

6.2.4. Stretch and Shrink

DI-Guy Scenario automatically adjusts behavior to fit the available space. When you have specified that a person walks a certain distance by positioning action beads, DI-Guy Scenario calculates the number of whole steps that will best fit, then stretches or shrinks the whole series by a small amount to make it fit exactly. The last column of the Action Spreadsheet, Stretch, shows how much the DI-Guy Scenario algorithms stretched or shrunk the action to make it fit in the available space.

If you notice excessive feet slipping by a character as it travels, it is probably because the path is excessively stretched or shrunk. Examine the Action Spreadsheet and tune the behavior by adjusting the locations of nearby action beads to reduce the stretch or shrink shown in this field. A stretch or shrink of zero is best.

6.3. Decision Beads

Decision beads are an easy way to make DI-Guy Scenario characters react to events in the scenario that requires no programming. (Although you do not need to understand the syntax of a programming language, you must still understand the logic for If, Then, Else decision making.)

An example of such reactivity is a character that runs until a gun is fired, then goes prone. Or a character that waits at the intersection until no cars are approaching then crosses the street safely.

The Character Objects:Decision Bead page is designed to lead you through specification of the decision logic, so you do not have to remember the operators or do any programming.

Only decision beads of the current path can be edited. If there is no current path, decision beads cannot be created, modified, or deleted.

Decision beads are represented in the 3D View window by red markers on a path. The currently selected decision bead rotates about the path.

6.3.1. Creating a Decision Bead

To create a decision bead:

1. Select the Character Objects:Decision Bead page (Figure 6-5).

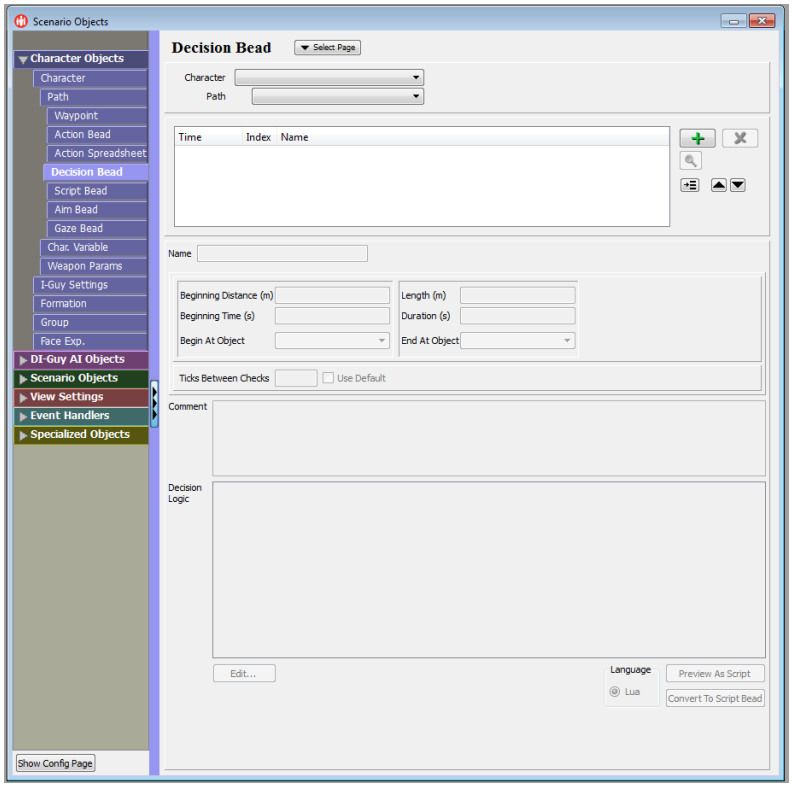


Figure 6-5. Decision Bead page

2. Click the Add button (+). A decision bead is added to the list.
3. Optionally type a name in the Name box.

4. Optionally, set the other parameters, as follows:
   - Beginning Distance. The point along the path where the decision bead is locationed.
   - Beginning Time. The elapsed scenario time at which the bead becomes active.
   - Begin At Object. Begin the decision when the character reaches this object.
   - Length. Specifies the length, in meters, along the path that the decision logic is active.
   - Duration. Specifies how long, in seconds, a decision bead remains active.
   - End At Object. Specifies that the decision stop when the character reaches the selected object.
   - Ticks Between Checks. Specifies how frequently the condition should be checked. Increasing this value may improve performance.

If you leave the Length and Duration values as 0, the decision logic is only checked once – when the scenario starts running, which is probably not what you want. These values are different ways of expressing the same basic behavior. Therefore if you change one, the other changes to a comparable value.

5. Create the decision logic, as described in "Creating Character-Level Decision Logic," on page 6-16.

6.3.2. Creating Character-Level Decision Logic

A new decision bead does not do anything. You must add decision logic to it. Character-level decision logic describes the circumstances under which you want a character to do something and what it should do. Therefore before you can create decision logic you must think about what you want the character to do and the logical process for making a decision to act.

You can also create decision logic at the scenario level. For details, please see Chapter 9, *Events and Event Handlers*.

All decisions use an "If…Then…Else" logic. However, you can shortcut the process by combining the decision bead parameters with the decision logic. For example, if you know you want to activate a particular camera at time t=10, set the decision bead to have a beginning time of 10.0. Then in the logic, just say "Then camera…" and so on. Not only are you not using the Else, you are not even using the If!

The Decision Editor is a front-end to the DI-Guy API, so the boxes refer to the objects, functions, and arguments that API functions require. At each stage of use, the Decision Editor dynamically shows the available objects, operators and arguments.

The following procedure describes a basic If, Then, Else logic. However, you can add multiple rows to each section, which would use AND or OR logic. You can change the order of the statements after you have added them and you can delete rows as well.

For an example of creating decision logic, please see the tutorial "Creating Decision Logic," on page 3-25.

To add decision logic:

1. Select the decision bead you want to edit.
2. Click Edit. The Decision Editor opens (Figure 6-6).

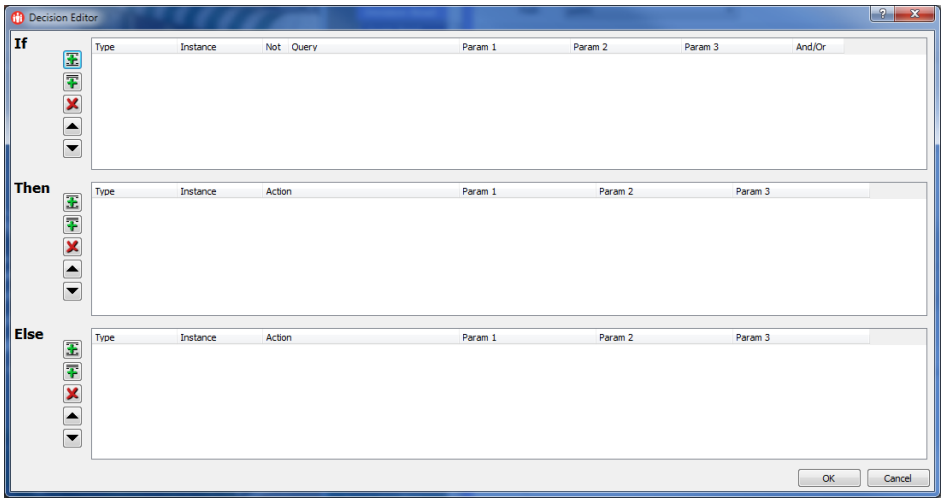


Figure 6-6. Decision Editor

3. In the If section, click the Add button (+). A list gets added to the first row in the window. Select an option from the list. A list gets added to the next column that is appropriate for the selection you made in the first column. (For the first line in the If section, it does not matter which Add button you click. If you add more than one line, the buttons either insert or append new lines.)
4. Build the complete If condition by selecting options from lists as they get added.

---

i As noted in the introduction to this procedure, the If expression is not absolutely required. It is possible to trigger a decision based solely on the time or distance parameters. However, most decision beads will use an If statement.

---

5. In the Then section, click the Add button. A list gets added to the first row in the window. Select an option. A new list gets added to the next appropriate column.
6. Build the complete Then expression by selecting options from the lists as they get added.
7. Optionally, click the Add button in the Else section. Build the Else condition the way you built the If condition.
8. Click OK. The completed decision script is displayed in the Decision Logic window. Figure 6-7 shows a simple decision as created in the Decision Editor and as it is expressed in the Decision Logic window. This decision logic says, "If the current character (this_character) is speaking, then sldr07-1 will nod its head one time for two seconds."

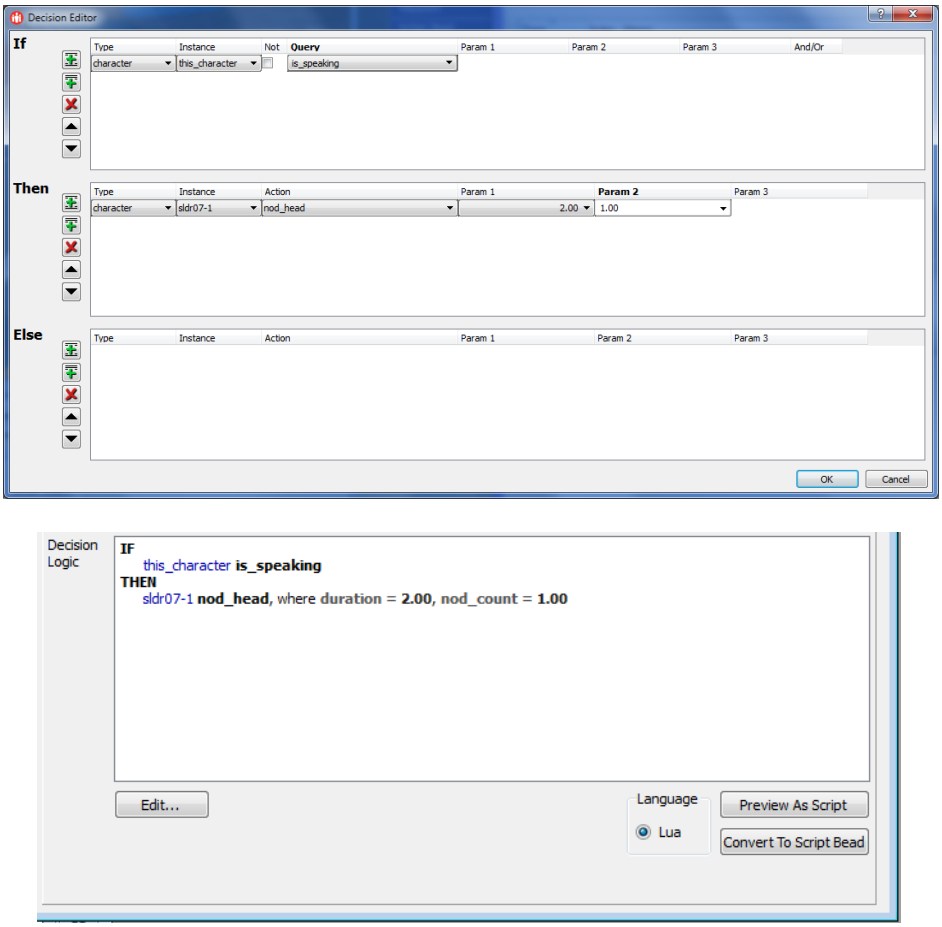


Figure 6-7. Decision Editor and finished decision logic

6.3.3. Converting Decision Beads to Script Beads

Decision beads are relatively easy to construct and the GUI ensures that you do not have to worry about syntax or mismatched parameters and other pitfalls of using a programming language. However, the process also imposes some constraints on the logic that you can write. DI-Guy Scenario also supports a complete Lua scripting environment for scripting character behaviors. It does this with script beads. You can convert decision beads to script beads. For information about script beads please see "Writing Script Beads," on page 6-20.

When you convert a decision bead to a script bead, it is converted, not copied. The decision bead will no longer exist.

To convert a decision bead into a script bead:

1. Select the Character Objects:Decision Bead page (Figure 6-5).
2. Select the decision bead you want to convert.
3. Optionally, click Preview As Script to see the Lua code.
4. Click Convert to Script Bead. The decision bead is converted to a script bead. The script bead is added to the Character Objects:Script Bead page (Figure 6-8).

6.4. Writing Script Beads

DI-Guy Scenario supports scripting using the Lua scripting language. You can access most of the DI-Guy API inside of scripts. You can get at all the other elements in the scenario, including the other characters and scenario objects. The DI-Guy API is described in *DI-Guy SDK Developers Guide* and the DI-Guy SDK class documentation.

Converting a decision bead into a script bead is an easy way to get started with Lua scripting. You will be able to examine the syntax of logic that you already understand.

It is beyond the scope of this manual to teach Lua scripting. There are many good resources online and in print. A good reference on the Lua scripting language is Programming in Lua (http://www.lua.org/pil/). For a brief introduction to the Lua code editor embedded in the script window, please see Appendix D, *Using the Lua Editor*.

To create a script bead:

1. Select the Character Objects:Script Bead page (Figure 6-8).

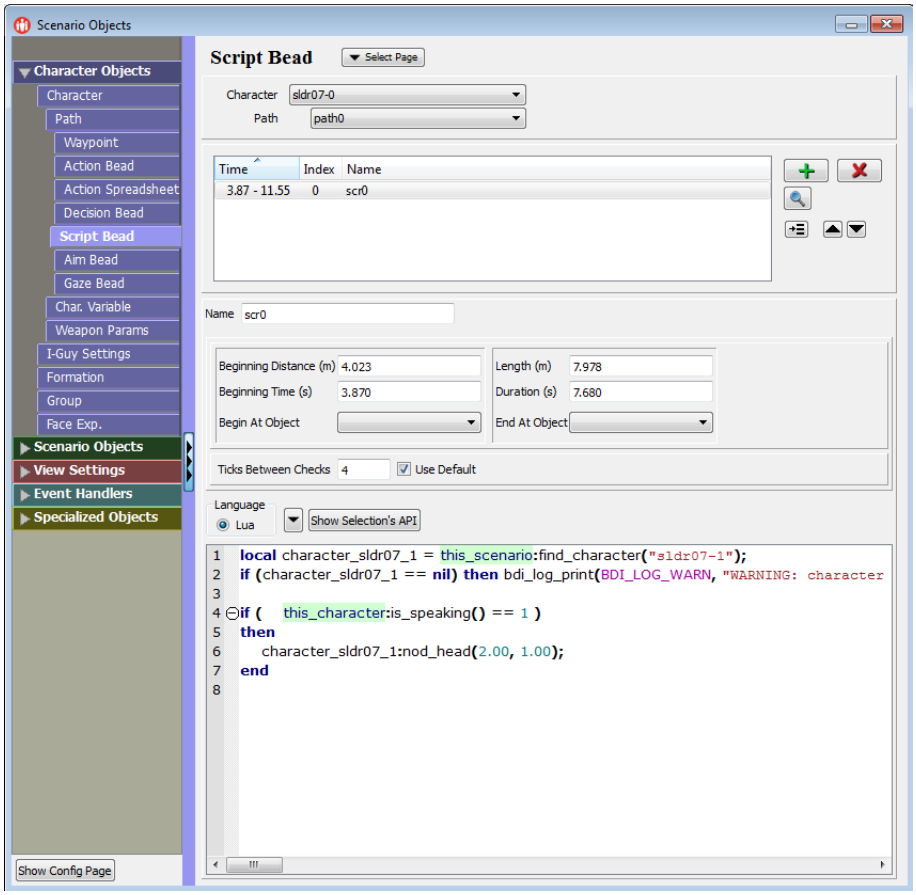


Figure 6-8. Script Bead page

2. Click the Add button (+). A script bead is added to the list.
3. Optionally type a name in the Name box.
4. Optionally, set the other parameters, as follows:
   - Beginning Distance. The point along the path where the script bead is located.
   - Beginning Time. The elapsed scenario time at which the bead becomes active.
   - Begin At Object. Begin the script when the character reaches this object.
   - Length. Specifies the length, in meters, along the path that the script logic is active.
   - Duration. Specifies how long, in seconds, a script bead remains active.
   - End At Object. Specifies that the script stop when the character reaches the selected object.
   - Ticks Between Checks. Specifies how frequently the condition should be checked. Increasing this value may improve performance.

If you leave the Length and Duration values as 0, the script logic is only checked once – when the scenario starts running.

5. Optionally, click the down arrow above the script window to set scripting parameters.
6. In the script window, write the script. If you select a function in the window and click Show Selection's API, API documentation for the function is displayed. If nothing is selected, after a warning prompt, top-level documentation opens.

6.5. Creating Aim Beads

For those actions that support aiming (the action name will contain the word "aim", such as stand_aim), you can use aim beads to set the direction a person is aiming. When the person crosses the aim bead, their aim changes from the previous value to the value specified by the aim bead.

You can specify the following types of aim beads:

- Aim angle mode. The direction of aim is given as an azimuth and elevation. To have a character aim up or to the right, the value is expressed as a negative value. An elevation of -45 causes the point of aim to be upward at a 45 degree angle. The heading and elevation can be expressed in global or local coordinates.
- Aim point mode. The character aims at the point specified. The point can be expressed in global or local coordinates.
- Aim character mode. The character aims at the specified person, vehicle, or prop.
- End aim. The character stops aiming.

The aim bead persists until a new aim bead or an end aim bead is encountered.

To create an aim bead:

1. Select the Character Objects:Aim Bead page (Figure 6-9).

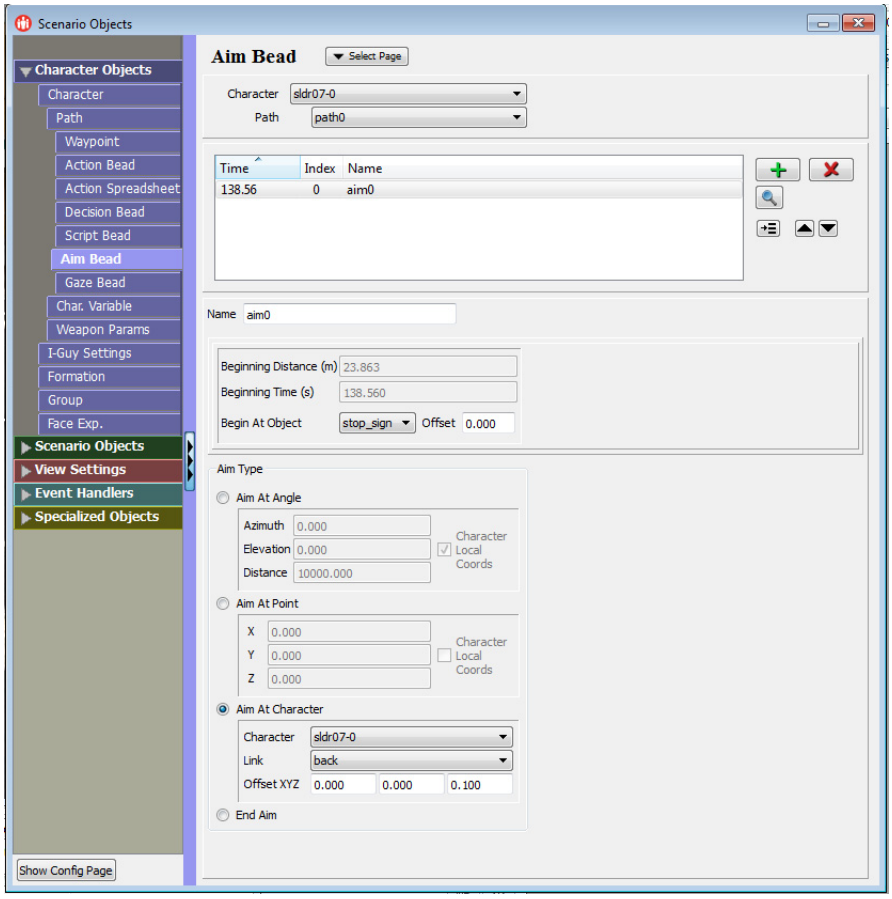


Figure 6-9. Aim Bead page

2. Click the Add button (+). A new aim bead is added to the list.
3. Optionally, type a name for the aim bead.
4. Specify when the aim bead begins, using one of the following:
   - Beginning Distance, in meters.
   - Beginning Time, in seconds.
   - Begin at Object. Select an object from the list.
5. In the Aim Type group box, select the aim type that you want to use.
6. Provide values for the parameters of the aim type that you chose.

6.6. Creating Gaze Beads

A character's gaze is the direction it is looking. You can specify the gaze for a character using gaze beads. When a character reaches a gaze bead, its gaze changes from the previous value to the value specified by the gaze bead.

You can specify the following types of gaze beads:

- Gaze angle mode. The direction of gaze is given as an azimuth and elevation with respect to straight ahead. To have a character gaze up or to the right, the value is expressed as a negative value. An elevation of -45 causes the point of gaze to be upward at a 45 degree angle. The heading and elevation can be expressed in global or local coordinates.
- Gaze point mode. The character gazes at the point specified. The point can be expressed in global or local coordinates.
- Gaze character mode. The character gazes at the specified person, vehicle, or prop.
- End gaze. The character stops gazing.

The gaze bead persists until a new gaze bead or an end gaze bead is encountered.

To create a gaze bead:

1. Select the Character Objects:Gaze Bead page (Figure 6-10).

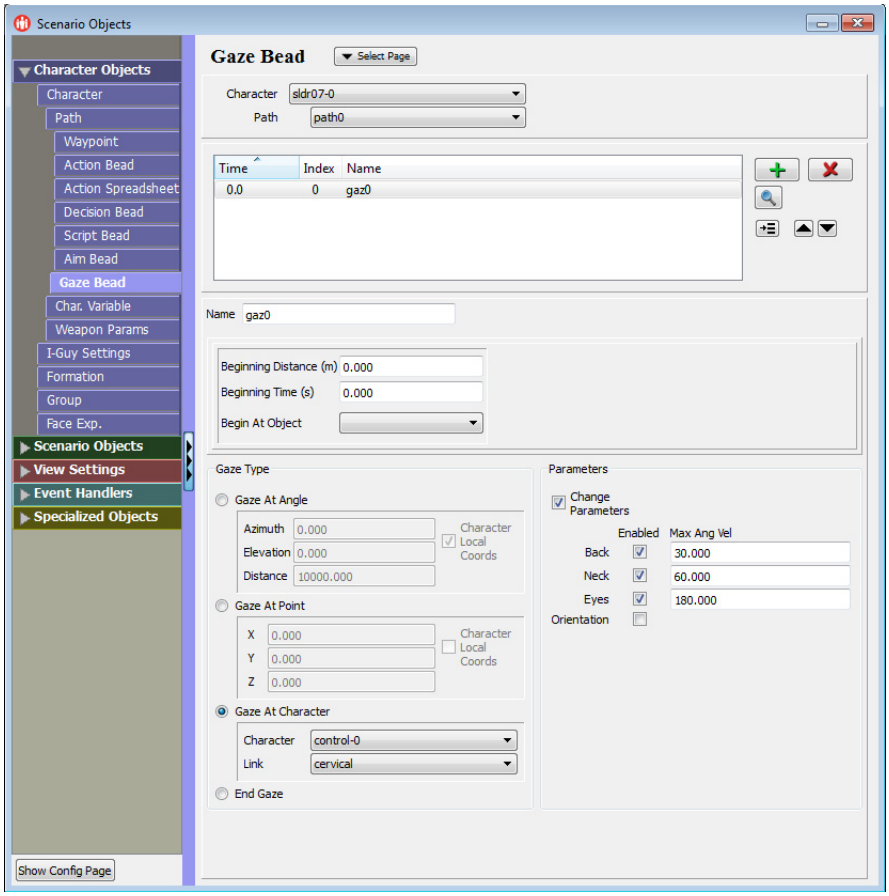


Figure 6-10. Gaze Bead page

2. Click the Add button (+). A new gaze bead is added to the list
3. Optionally, type a name for the gaze bead.
4. Specify when the gaze bead begins, using one of the following:
   - Beginning Distance, in meters.
   - Beginning Time, in seconds.
   - Begin at Object. Select an object from the list.
5. In the Gaze Type group box, select the gaze type that you want to use.
6. Provide values for the parameters of the gaze type that you chose.
7. Optionally, in the Parameters group box, specify how the character will turn the various parts of its body to achieve the gaze goal.

6.7. Gestures

Gestures are motions that involve only a subset of a character's body and that can be added to other actions. Nodding the head yes, shaking the head no, and moving one's hand to emphasize speech are all examples of common gestures. DI-Guy Scenario allows you to specify gestures for characters using decision logic. When a character executes a gesture, the motion 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.

6.7.1. Head Nods and Shakes

Head nodding (yes) and head shaking (no) are gestures. These actions are algorithmically driven, with parameters that control the duration of the action and the number of cycles of head nod or shake. The motions generated for these gestures are superimposed on the other activities you have given the characters to do.

To specify a head nod or head shake gesture:

1. Select the character.
2. Create a decision bead for that character.
3. In the THEN or ELSE field of the decision bead specify the `nod_head` or the `shake_head` action.
4. On the Character Objects:Decision Bead page, specify the duration of the action and the number of repetitions. A half cycle of head nodding brings the head down and back up. A full cycle of head nodding brings the head down, then up higher than its initial orientation, then down to level. Head shaking works similarly.

6.7.2. Generic Gestures

Head nodding and shaking are algorithmically driven gestures. DI-Guy Scenario also supports motion data driven gestures.

To specify one of these more general gestures, specify `execute_gesture` action in your decision logic. The gestures section of the DI-Guy Data reference (*./doc/index.html*) lists the gestures provided with DI-Guy. The online documentation also provides information about advanced features in the DI-Guy gesture mechanism, such as channels and stages, which are available through Lua scripting.

6.7.3. Previewing Gestures

You can preview a character's gestures. The Gesture Previewer takes over the 3D View and focuses on the active character. A list of available gestures is shown, and can be actively auditioned on the character. Use decision beads or script beads to add gestures to the character.

To preview gestures:

1. Select the character whose gestures you want to preview.
2. On the Character Objects:Path page (Figure 5-1), click Gesture Previewer, or in the Input Mode window, click Show Gesture Previewer. The 3D View adds gesture preview controls to the window and focuses on the selected character (Figure 6-11).

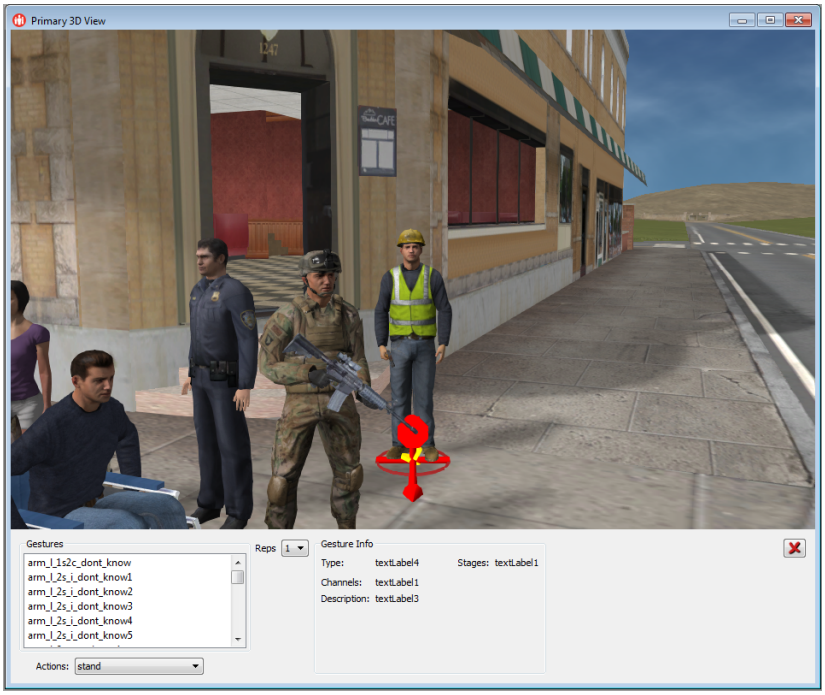


Figure 6-11. Gesture preview mode

3. Select a gesture in the Gestures list to see it.
4. Change the action in the Actions list to see how the chosen gesture works with that action.
5. Select a number in the Reps list to specify how many times to repeat the gesture.
6. Click the Close button (✖) to close the Gesture Previewer and return to normal 3D View mode.

7. Scenario Objects

This chapter describes objects that can affect all of the characters in a scenario.

Scenario-Level Objects 7-2
Guides 7-3
Adding Special Effects (Particle Systems) 7-5
Adding a New Particle System 7-7
Particle System Parameters 7-8
Creating Path Shapes 7-11
Sensor Regions 7-12
Creating Sensor Regions 7-13
Editing Sensor Regions 7-15
Creating Signals 7-16
Tracking Signal Use and Triggering Signals 7-17
Adding Sound Effects 7-18
Setting the Sound Volume 7-19
Creating Scenario Variables 7-20

7.1. Scenario-Level Objects

Many objects in DI-Guy Scenario are tied to characters, such as paths, action beads, and so on. (These objects are described in Chapter 4, *Creating and Editing Characters* and Chapter 5, *Paths and Waypoints*.) Other objects exist at the scenario level and are available to all characters. These include:

- Guides
- Particle systems
- Path shapes
- Sensor Regions
- Signals
- Sounds
- Variables.

DI-Guy Scenario also has scenario-level settings that affect performance. For details about performance settings, please see Chapter 10, *Configuring Performance*.

7.2. Guides

Guides are a means of controlling the travel of characters and vehicles, typically used for formations and incoming network entities. Depending on the rate of network updates and the speed at which entities are moving, there can be significant differences between a character's dead-reckoned[1] location and its actual location. When its position is updated, if a character is moved immediately to its actual location, the movement can be visually disconcerting. Guides help smooth out the transitions between an entity's current location and the desired location. In these cases, input to the guide is in the form of desired position, velocity, and orientation. The guide's control algorithm uses that information to determine the behavior of the character.

Each guide has a name, specifies a control algorithm, and has a set of control parameters.

The control algorithms are:

- Exact. No smoothing. Forces the character to stay precisely in the desired location at all times.
- Drift1. Causes followers to travel at roughly the same speed as the leader.
  - If the follower's position error exceeds **distance_zone_0**, then the character's position is gradually drifted toward the desired position with time constant **position_time_constant**.
  - If the follower's orientation error exceeds **azimuth_zone_0**, the character's orientation is gradually drifted toward the desired orientation with time constant **orientation_time_constant**.
  - If the position error exceeds **distance_zone_1**, the position is immediately adjusted to the desired position.
  - If the orientation error exceeds **azimuth_zone_1**, then the orientation is immediately adjusted to the desired orientation.
- Follow1. Scales travel speed and switches gaits to control the position error. If the character gets too far in front of the desired position (negative position error), the follower switches to standing until the position error returns to positive. The smoothing parameters keep the character from oscillating too much in difficult following situations.
- Follow2. This control algorithm is the same as Drift1, except that the method of reducing error is to speed the character up or slow it down compared to the lead character's speed (rather than by introducing drift). The maximum amount the character's speed of travel is scaled is determined by **max_speed_scale** parameter.
- Missile1. For movement in the air.

1. Dead-reckoning is a method of calculating changes to an entity's position based on its speed, orientation, and heading. It is used to maintain entity movement between network updates.

To create a guide:

1. Select the Scenario Objects:Guide page (Figure 7-1).

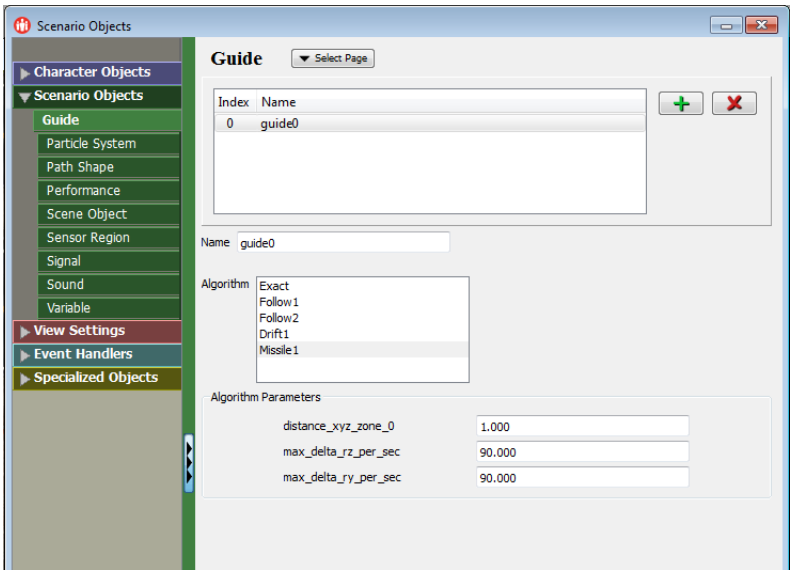


Figure 7-1. Guide page

2. Click the Add button (+). A new guide is added to the list.
3. Optionally, change the name of the guide.
4. Select an algorithm for the guide.
5. Optionally, change the parameter values for the guide.

7.3. Adding Special Effects (Particle Systems)

DI-Guy Scenario supports special effects, such as smoke, fire, explosions, debris, rain, snow, and vomit. These effects are created using a powerful and versatile particle systems engine. DI-Guy Scenario is delivered with predefined special effects. On the Scenario Objects:Particle page you can modify the existing special effects to create an unlimited variety of new ones.

Special effects can be thought of as characters of type “effect”, with an appearance that creates the effect. They are visualized in the 3D View as boxes when the scenario is not running and as the desired effect when it is (Figure 7-2).

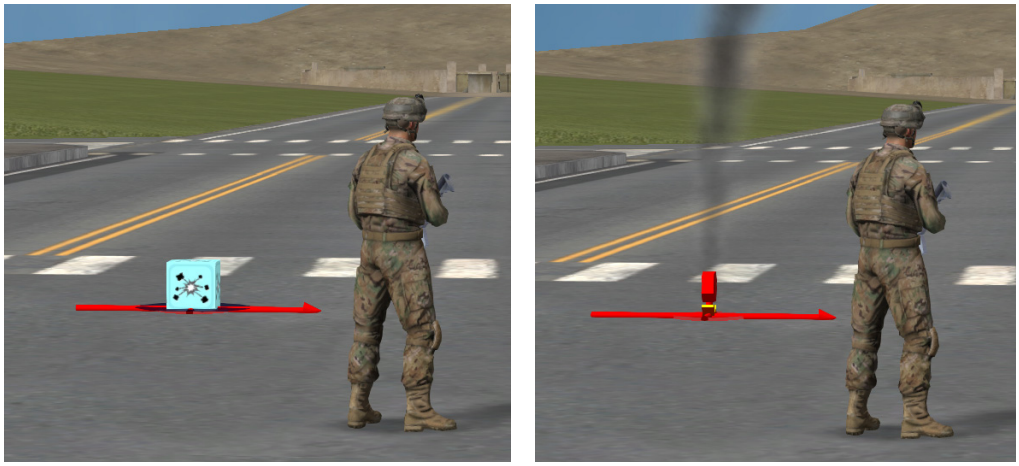


Figure 7-2. Special effect

As long as the character is visible, the particles are emitted. When the object disappears it stops emitting new particles, but the particles already emitted are still be visible in the scenario, for example, as drifting smoke.

The default set of special effects is defined in the configuration file *./config/diguy/particles.cfg*. If you have an old scenario that does not include the effects definitions, but you would like to add them, you can import saved libraries into your scenario.

Selecting Show Collision Planes will visualize all the collision planes in the scene.

You can hide the particle system box on the View Settings:Visibility page.

A large set of parameters determines the look and behavior of the particle systems. They are described in the “Particle System Parameters,” on page 7-8.

To add a special effect to a scenario:

1. In the Input Mode window, click PeopleBlitzer.
2. In the Blitz Char Type list, select effect.
3. In the Blitz Appearance list, select the effect that you want to add, such as `fire_1` or `smoke_1`.
4. Click in the 3D View where you want to place the effect. A waypoint is placed just as for any other character. A blue cube indicates that this is an effect.

7.3.1. Adding a New Particle System

Particle systems have many parameters. You can edit the particle systems provided with DI-Guy Scenario. However, if you want to change a particle system, it is preferable to add a new system based on an existing one and edit the new system. This way you always have the original particle systems available.

To add a new particle system:

1. Select the Scenario Objects:Particle System page (Figure 7-3).

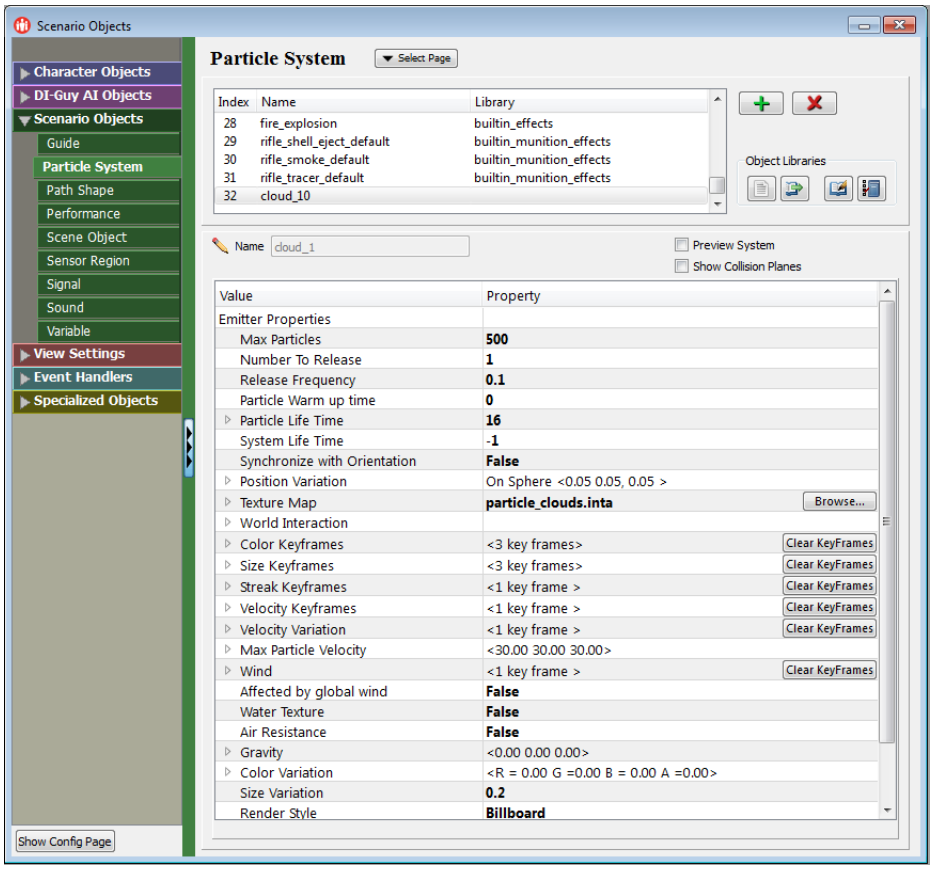


Figure 7-3. Particle System page

2. Select an existing particle system of the type that you want to add.
3. Click the Add button (+). A new particle system is added to the list. It is named based on the system you selected by appending a 0 to the name.
4. Optionally, give the new system a new name.
5. Edit the parameters in the property sheet. For details about the parameters, please see "Particle System Parameters," on page 7-8.

7.3.2. Particle System Parameters

When you select a parameter, a brief description is displayed at the bottom of the Scenario Objects:Particle System page. Table 7-1 describes the parameters.

Table 7-1: Particle system parameters

Parameter	Description
Max Particles	The maximum number of particles that this system creates. Once the maximum is reached, the system waits until particles die before emitting new particles.
Number To Release	The number of new particles to emit when the system is updated. This is bounded by the release interval.
Release Frequency	How frequently the system attempts to emit new particles. (Bounded by update rate.)
Particle Life Time	How long, in seconds, each particle lives.
System Life Time	How long, in seconds, the entire effect should live.
Synchronize with Orientation	If the effect is parented, use the orientation of the parent link.
Position Variation	Where particles should first appear. Noise Type, Min, and Max allow for variation in particle position start.
Texture Map	
Texture Map	The texture map to use on the particles.
Number of X, Y Repeats	This allows the texture map to be divided into smaller sub textures, which is useful in the following cases: ◆ If animation speed is set to zero, a new particle is randomly assigned a sub texture. In this way you can make the same emitter create very different looking particles using the same texture. ◆ If animation speed is set to a number other then zero the particles play through the sub textures like the pages of a flip book, allowing for animated particles.
Rotation Speed	How quickly the particle rotates, in degrees per second.
Blend	Controls particles being blended with the background. (Controls if source and destination parameters work.)
Blend Src Function	OpenGL parameter describing the source transfer method, usually GL_ONE.
Blend Dest Function	OpenGL parameter describing the destination transfer method, usually GL_ONE.
Depth Write	Controls if the particles write to the depth buffer. Turn this on and turn blending off for more solid looking particles.
Alpha Threshold	Controls the clipping of textures based on an alpha threshold. It is useful for controlling the shape of solid particles.

Table 7-1: Particle system parameters

Parameter	Description
World Interactions	
Coordinate Space	Controls if particles move along with the emitter that created them. Particles in local space move along with the emitter as the emitter moves, they can also be scaled. Particles in world space move independently of the emitter and they can be made to collide with world geometry.
Collision Planes	An array of planes that particles can interact with. Each plane has a position and orientation as well as a collision behavior. The behaviors are: bounce, kill, and stick. If bounce is selected the bounce factor is factored in to how much energy the particles retain after hitting the plane.
Local Space Collision Planes	Determines if collision planes track with the object if the object moves.
Collide with World	Enables colliding with the scene objects in the world. Uses an octtree and a spatial partitioning scheme to assess potential collisions. The number of particles colliding with the world should be kept to a reasonable number to keep frame rate up. Moving scene objects will not be properly collided against.
Bounce Factor	The amount to multiply the velocity (usually reduces velocity) as a result of the collision.
Collision Result	What to do when the particle strikes a collision plane. ◆ Bounce. The particle bounces, but it maintains its life. ◆ Bounce and die. The particle bounces; then it dies. ◆ Stick. The particle maintains its position and its life. ◆ Die. The particle is immediately terminated.
Color Keyframes*	The color and alpha value to draw the particle with, colors linearly interpolate between keyframes.
Size Keyframes*	The size of particles. The size linearly interpolates between keyframes.
Streak Keyframes*	The amount the particles should smear in the direction that they are moving.
Velocity Keyframes*	An impulse to apply to a particle as it hits the key frame time.
Velocity Variation	A random velocity to apply to a particle as it hits the keyframe time
Wind	An acceleration factor to apply to the particles. Linearly interpolates between keyframes.
Affected by Global Wind	If true, global wind affects the particle system.
Water Texture	If true, supports footprints on wet surfaces.

Table 7-1: Particle system parameters

Parameter	Description
Air Resistance	When set to true, particles will not go faster than the wind speed.
Gravity	A constant acceleration force applied every frame. <0,0,0> is good for lighter than air particles. <0,0,-9.8> mimics normal earth gravity and is good for solid objects like debris.
Max Particle Velocity	The maximum velocity a particle can have.
Color Variation	Applied to each particle when it is emitted. This can give the particles random variations in color.
Size Variation	Applied to each particle when it is emitted. It is a fudge factor in addition to the size keyframes for varying particle size.
Render Style	Indicates how the graphics are drawn. Usually set to Billboard. ◆ CameraAligned. Aligns the normal of the billboards to the camera. ◆ VelocityAligned. Aligns the normal of the billboards in the direction of the particles velocity. ◆ Sparks. Uses line graphics instead of billboards. ◆ Zup. Z up.
Cast Shadow	If true, the particle system casts a shadow.
Emit Light	If true, the particle system emits light.

*Keyframes are used to vary the parameters of a particle system as a function of time. All keyframes have a time associated with them that is a percentage of the particles' lifetime. Most of the keyframe groupings require at least one keyframe, usually at time 0. Keyframes must also be listed in time order. The user interface will try to maintain this requirement.

7.4. Creating Path Shapes

Path shapes are paths that are not tied to a specific character. You can access them through the DI-Guy API. Waypoints in a path shape behave just like the waypoints used to define character paths.

Because path shapes are not specific to any character, you cannot add character-specific objects to them, such as action beads, decision beads, or script beads.

To create a path shape:

1. Select the Scenario Objects:Path Shape page (Figure 7-4).

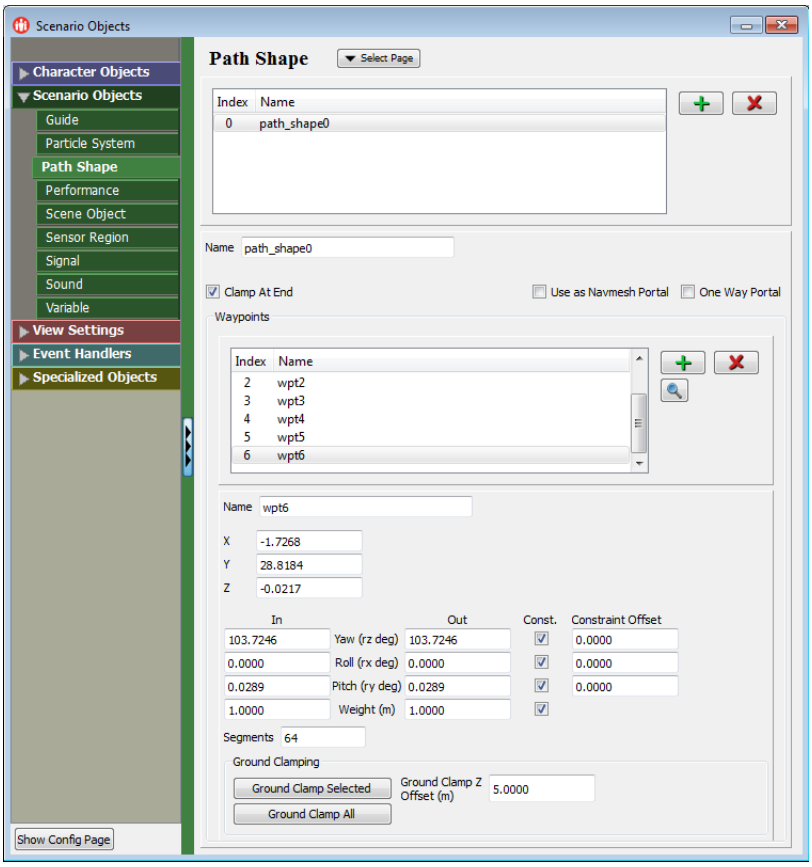


Figure 7-4. Path Shape page

2. Click the Add button (+).
3. In the Input Mode window, click Path Insert.
4. In the 3D View window, click to place the waypoints for the path shape.
5. Optionally, on the Path Shape page, type a name for the path shape.
6. To use this path shape to connect disconnected navmesh regions, select Use As Navmesh Portal.

7. To use this path for a one way connection between navmesh regions, select One Way Portal.
8. To have a character return to the starting point from the end of the path shape, select Clamp At End.
9. Edit the values for the waypoints as desired.

7.5. Sensor Regions

A sensor region is an area in the terrain that can detect when a person or vehicle enters it and makes the information available to the system. The information can be used to control the actions of people in the scenario.

Each sensor region is represented in the 3D View window by a wireframe box (Figure 7-5). The current sensor region is drawn white. All other sensor regions are drawn blue.

Figure 7-5. Sensor region

7.5.1. Creating Sensor Regions

You can create sensor regions from the Input Mode window or from the Scenario Objects:Sensor Region page. Sensor regions are not attached to specific paths or characters, so it does not matter which character and path is selected when editing sensor regions.

When you create a sensor region using Sensor Insert mode, the cursor stays in this mode and you can continue to create additional sensor regions.

To create a sensor region from the Input Mode window:

1. In the Input Mode window, click Sensor Insert.
2. In the 3D View, click to place one vertex of the region and drag to define the entire sensor region.
3. Optionally, edit the sensor region's parameters on the Scenario Objects:Sensor Region page (Figure 7-6).

To create a sensor region from the Sensor Region page:

1. Select the Scenario Objects:Sensor Region page (Figure 7-6).

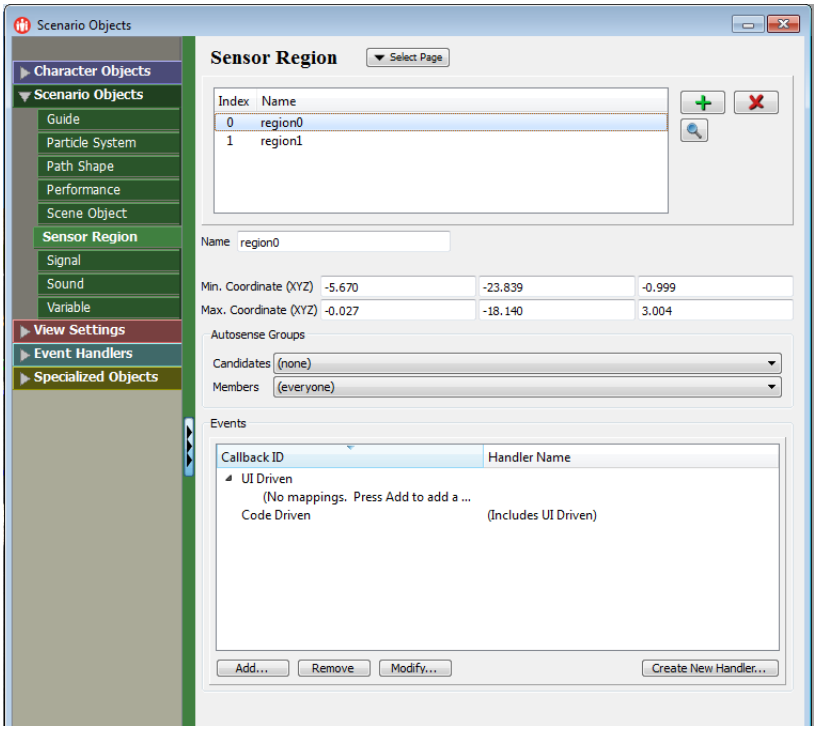


Figure 7-6. Sensor Region page

2. Click the Add button (+). A sensor region is added to the center of the terrain and it is added to the list.
3. Optionally, give the sensor region a name. It must be unique within the scenario.
4. In the Input Mode window, click Sensor Edit.
5. In the 3D View, click one of the legs of the sensor region and drag it to the size that you want.
6. Edit the sensor region's parameters as desired.

7.5.2. Editing Sensor Regions

You can edit a sensor region's coordinates, its autosense groups, and events associated with it.

An autosense group is a group of characters, as defined in the Character Objects:Group page, that a sensor region can sense. Characters that are autosensed can call callback functions that have been registered with the CALLBACK_ID_AUTOSENSE_CHARACTER_ENTERED or CALLBACK_ID_AUTOSENSE_CHARACTER_LEFT IDs.

You can also have an "output" group that is populated by characters in the Candidates group that are currently in the sensor region. Specify the name of this group in the members field.

You can edit sensor regions on the Sensor Region page or using the Sensor Edit input mode.

To edit a sensor region on the Sensor Region page:

1. Select the Scenario Objects:Sensor Region page (Figure 7-6).
2. Select the sensor region that you want to edit.
3. Specify the position by changing the Min. Coordinate and Max Coordinate values. They represent the diagonal corners of the sensor region box, in world coordinates, as follows:

Table 7-2: Sensor region coordinates

Coordinate	Description
X	Min is bottom near left corner X position. Default: –1. Max is top far right corner X position. Default: 1.
Y	Min is bottom near left corner Y position. Default: –1. Max is top far right corner Y position. Default: 1.
Z	Min is bottom near left corner Z position. Default: –1. Max is top far right corner Z position. Default: 1.

4. To have the sensor region sense members of a group, in the Autosense Groups group box, select a group from the Candidates list.
5. To have members of the candidate group that are in the sensor region be assigned to another group, select the output group in the Members list.
6. Add or edit events to associate with this sensor region. For details, please see Chapter 9, *Events and Event Handlers*.

Editing a Sensor Using Sensor Edit Mode

You can edit a sensor region by dragging one of its legs. You cannot move a sensor region as a unit.

To edit a sensor region using sensor edit mode:

1. In the Input Mode window, click Sensor Edit.
2. In the 3D View, click and hold one leg of the region you want to edit.
3. Drag the leg to resize the region.

7.6. Creating Signals

Signals are used to coordinate behavior of people and vehicles, and to allow a character to react to the behavior of other characters. You can use signals to qualify decision logic (decision beads, script beads, and script events) and they can be triggered by decision logic.

You can associate sounds with signals, so the sound is played when the signal is triggered.

To create a signal:

1. Select the Scenario Objects:Signal page (Figure 7-7).

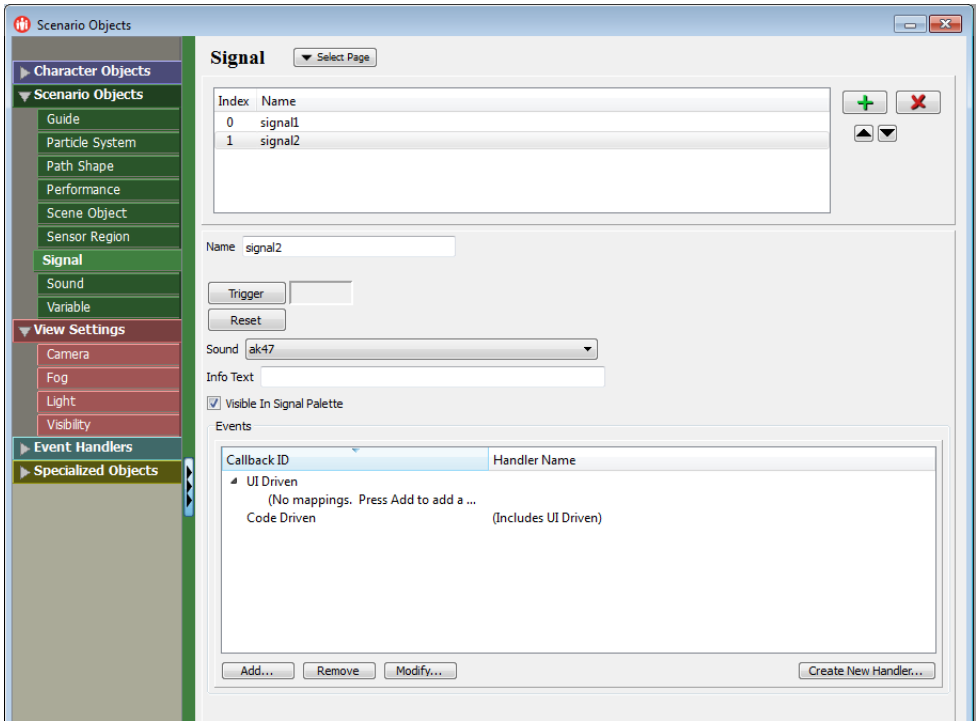


Figure 7-7. Signal page

2. Click the Add button (+). A new signal is added to the list.
3. Optionally, type a name for the signal.
4. Click the Trigger button to set a trigger value.
5. Optionally, select a sound from the Sound list to associate a sound with the trigger. (For details about configuring sounds, please see "Adding Sound Effects," on page 7-18.)
6. Optionally type informative text in the Info Text box.
7. If you want the signal to be listed on the Signals dialog box (Figure 7-8), select the Visible in Signal Palette check box.
8. Associate an event with the signal. For details, please see "Mapping Events to Callbacks and Logic," on page 9-11.

7.6.1. Tracking Signal Use and Triggering Signals

The Signals dialog box (Figure 7-8) displays how frequently a signal has been triggered. It also lets you trigger signals manually.

Figure 7-8. Signals dialog box

To display signal usage:

1. Choose **Window** → **Signals**. The Signals dialog box opens. It lists each signal that has been configured for display. The number of times the signal has been triggered is displayed in the box next to the signal. For details about including signals in this dialog box, please see "Creating Signals," on page 7-16.
2. To trigger a signal, click its button.
3. To reset the counter, click RESET.

7.7. Adding Sound Effects

Sounds effects are an important way to enrich a scenario. DI-Guy Scenario allows you to associate sounds with people, vehicles, and events in the scenario.

DI-Guy Scenario supports use of *.wav* sound files. You can attach a sound to the scenario using decision beads or associate a sound with a signal so that the sound is played when the signal is triggered.

You can also give a sound 3D parameters, which determine how loud it is at particular distances.

To add a sound effect:

1. Select the Scenario Objects:Sound page (Figure 7-9).

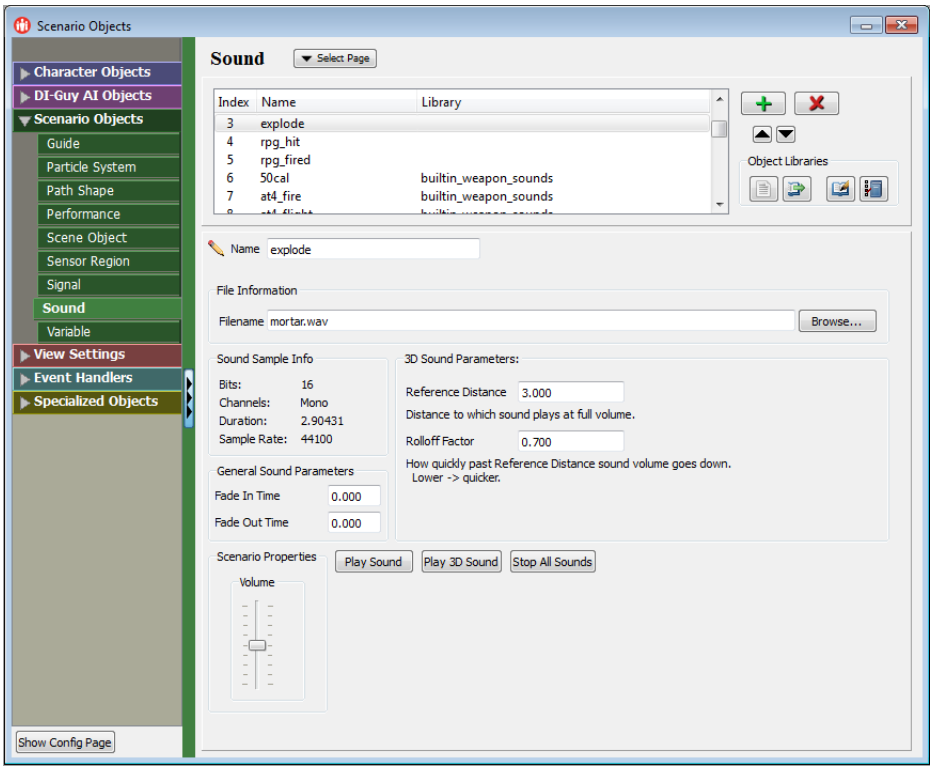


Figure 7-9. Sound page

2. Click the Add button (+).
3. Optionally, type a name for the sound.
4. In the Filename box, type the path to the sound, or click the Browse button and select a sound file.
5. Optionally, change the 3D Sound Parameters, which are described on the page.
6. Optionally, specify the Fade In Time and Fade Out Time.

7. To test the sound, click Play Sound or Play 3D Sound. Play 3D Sound plays the sound at the origin of the terrain. Stop All Sounds stops the test sound.

7.7.1. Setting the Sound Volume

You can adjust the sound volume for all scenario sounds.

➤ To set the sound volume, on the Scenario Objects:Sound page, adjust the Volume slider.

7.8. Creating Scenario Variables

Just as you can create variables associated with each character, you can also create variables associated with the entire scenario.

These variables can be used for a variety of purposes, particularly for logic. Variables can be accessed from other places in DI-Guy Scenario, or by user applications using the DI-Guy SDK that may have loaded the scenario created in DI-Guy Scenario. You can set variables to a value through the GUI, by decision logic, or by script events. Variables can be integers, floats, or strings. Anywhere where there is visibility into the scenario class, the variables can be accessed. An API is available for working with variables. Once a variable is specified, DI-Guy Scenario decision beads and script logic can test its value, and take action according to the result of the test.

To create a variable:

1. Select the Scenario Objects:Variable page (Figure 7-10).

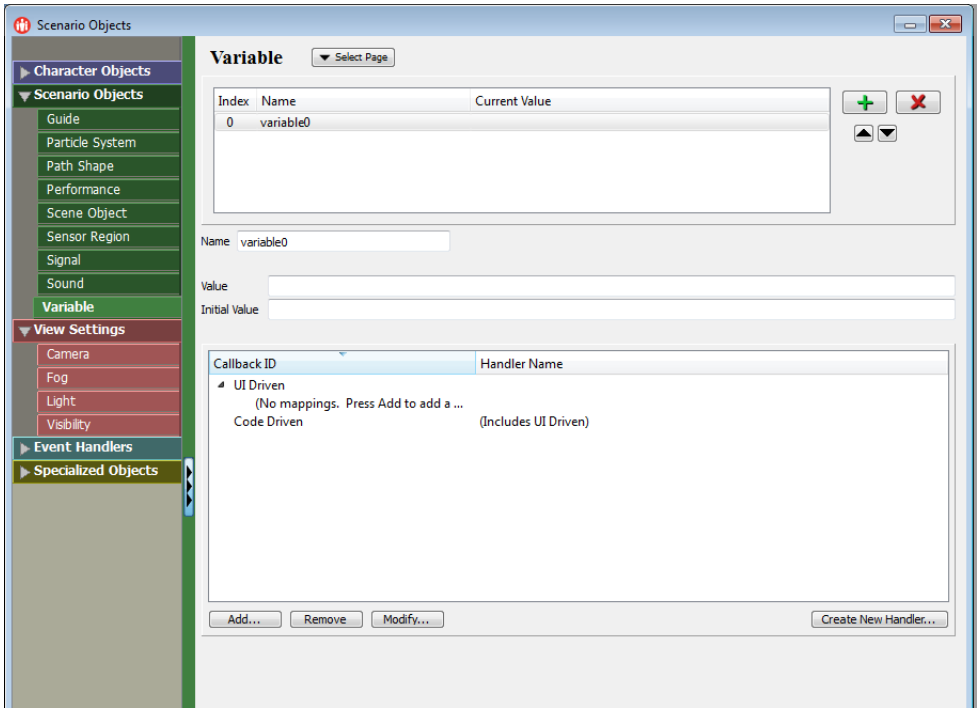


Figure 7-10. Variable page

2. Click the Add button (+). A new variable is added to the list.
3. Optionally, type a name for the variable in the Name box.
4. If you want to change the value of a variable while a scenario is running, set the value in the Value box.
5. In the Initial Value box, enter the value this variable will have at the start of the scenario or when it is reset.

8. Using the Camera

This chapter explains how to configure the camera and save and load settings.

Moving the Camera 8-2
Setting the Direction of Camera Motion 8-2
Changing the Camera's Speed in Mouse Move Mode 8-3
Teleporting the Camera 8-3
Attaching the Camera to a Character 8-4
Tracking a Character 8-5
Setting Up Point of View (POV) Mode 8-5
Saving and Loading Camera Settings 8-5
Saving Camera Settings 8-6
Loading Camera Settings 8-6
Changing the Order of Camera Settings 8-7
Advanced Camera Settings 8-7
Changing the Projection Mode 8-7
Changing the Camera's Speed for Projection Modes 8-7
Setting the Clipping Planes and Field of View 8-8
Specifying Camera Override Preferences 8-9
Viewing a Scenario in Symbolic View 8-9

8.1. Moving the Camera

When DI-Guy Scenario is in Camera input mode, you can move the camera using the mouse. You can move it in opposite directions using the left mouse button and right mouse button and can change the orientation by pressing both buttons at the same time. The specific directional pair – up/down, left/right, forward/backward depends on the current movement mode. For details about how to specify the movement mode, please see "Setting the Direction of Camera Motion," on page 8-2.

To move the camera using the mouse:

1. In the Input Mode window, click Camera.
2. Move the cursor into the 3D View window.
3. Click or hold the right mouse button, left mouse button, or both buttons depending on the type of movement you want.

8.1.1. Setting the Direction of Camera Motion

By default, when you are in Camera mode and use the mouse buttons to move the camera in the 3D View, the camera moves forward and backward. However, you can also set the camera to move left/right and up/down.

To set the direction of camera movement:

- You can steer while traveling forward and backward by moving the mouse left/right, up/down in the 3D View. You can aim the camera without moving it by holding both left and right mouse buttons down at the same time and moving the cursor left/right and up/down in the 3D View.
- Up/down. Press **`d`**, or on the View Settings:Camera page (Figure 8-1), select Up/Down Mouse Move mode.
- Left/right. Press **`s`**, or on the View Settings:Camera page, select Left/Right Mouse Move mode.
- Forward/backward. Press **`f`**, or on the View Settings:Camera page, select Forward/Backward Mouse Move mode.
- Orientation. Move mouse while holding down single directional button or both buttons.

8.1.2. Changing the Camera's Speed in Mouse Move Mode

When the camera is in mouse move mode, you can change its speed of movement in meters per second.

➤ To change the camera's speed, on the View Settings:Camera page (Figure 8-1), in the Mouse Move Mode group box, change the value in the Spd box.

➤ To change the camera's speed dynamically, press and release **+** to increase speed and **–** to decrease speed.

Do not hold down the **+** or **–** key. The change in speed is not continuous. Each press and release changes the speed by one increment.

8.2. Teleporting the Camera

You can make the camera jump instantly from one location to another, rather than flying through the scene.

➤ To teleport the camera, while in Camera mode, point the mouse where you would like to place the camera and **`Shift`**+click. The camera flies to the new location, and spins around to face where it came from.

8.3. Attaching the Camera to a Character

You can attach a camera to any person or vehicle in the scenario. When attached, the camera travels with the person.

To attach the camera to a character:

1. On the View Settings:Camera page (Figure 8-1) select the camera you want to attach.

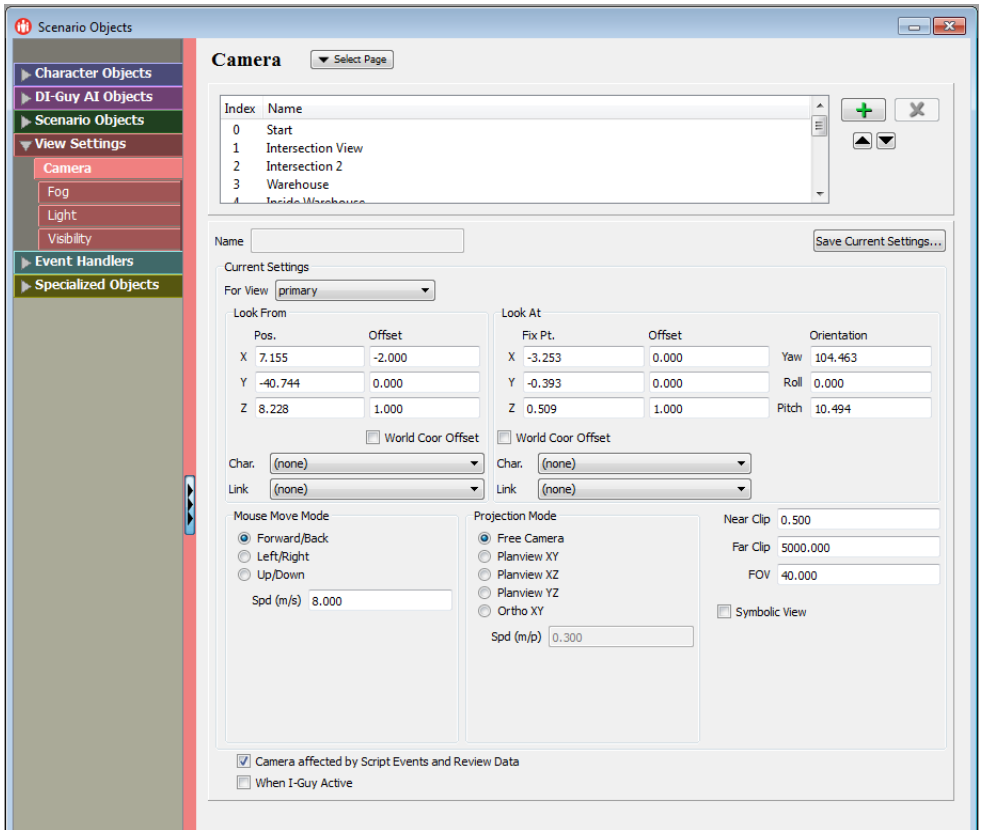


Figure 8-1. Camera page

2. In the Look From group box, select a character from the Char. list.
3. Optionally, specify a link. This is the part of the character's body to which the camera is attached. You can offset the camera to change the actual view. If you do not specify a link, the camera moves with the base or pelvis of the character.
4. Adjust the relative positioning of the camera with respect to the character using the mouse in the 3D View or by setting the values in the Pos. and Offset boxes on the Camera page.

8.4. Tracking a Character

You can specify that the camera track a character. When it tracks a character, it continuously looks at that character. It does so even if it is attached to another character and is moving with it.

To track a character:

1. On the View Settings:Camera page (Figure 8-1) create a new camera or select the camera you want to configure.
2. Specify a location to look at in the Fix Pt, Offset, and Orientation boxes, or in the Look At group box, select a character from the Char. list.
3. Optionally, specify a link. This is the part of the character's body the camera looks at. You can offset the camera to change the actual view. If you do not specify a link, the camera looks one meter above the reference point for the person.

8.5. Setting Up Point of View (POV) Mode

You can set up the camera to support a first person point of view (POV). POV shows the world from the point of view of the character to which the camera is attached. In POV mode, the camera travels with the selected person or vehicle and maintains the specified relative position and orientation.

To set up first person point of view:

1. On the View Settings:Camera page (Figure 8-1) create a new camera or select the camera you want to configure.
2. In the Look From group box, select a character from the Char. list.
3. In the Look At group box, select the same character from the Char. list.
4. Position the camera to get the view you want. You can do that in the 3D View window using the mouse, or using the Position and Fix Point boxes on the Camera page.
5. Optionally, give the camera setting a name and save the setting.

8.6. Saving and Loading Camera Settings

You can define and save an unlimited number of camera settings. You can quickly load any of the first 10 camera settings (as listed on the View Settings:Camera page) into the camera using a hot key.

8.6.1. Saving Camera Settings

You can save camera settings on the View Settings:Camera page or you can save settings to a hot key from the keyboard. When you save camera settings on the Camera page, if the setting is one of the first ten settings (index 0-9), you can load it with the corresponding hot key. If the first ten slots are already filled, the settings are saved and can be loaded from the Camera page.

To save camera settings:

1. In the Input Mode window, select Camera.
2. Using the mouse, position the camera to achieve the view you want.
3. Select the View Settings:Camera page (Figure 8-1).
4. Click Save Current Settings.
5. Optionally, in the Name box, type a name for the setting.

Saving Camera Settings to a Hot Key

If you save camera settings to a hot key, the settings overwrite the settings previously assigned to that key, if any.

To save a camera setting to a hot key:

1. In the Input Mode window, select Camera.
2. Using the mouse, position the camera to achieve the view you want.
3. Type **`Shift`**+*number* where *number* is a digit from 0-9. This saves the camera location and viewing parameters as camera setting *number*.

8.6.2. Loading Camera Settings

You can load camera settings from the View Settings:Camera page or from a hot key. If you need to reset the camera to a default setting, loading a camera setting is the method to use.

To load a camera setting from the Camera page:

1. Select the View Settings:Camera page (Figure 8-1).
2. In the list of settings, select the setting you want to load. The 3D View changes to that setting.

Loading a Camera Setting from a Hot Key

➤ To load a camera setting from the keyboard, press the number key for the saved setting you want to load.

8.6.3. Changing the Order of Camera Settings

As noted previously, you can use hot keys to quickly switch between up to 10 camera settings. If you have saved more than 10 settings and want to change which settings are bound to hot keys, you can do so.

To change the order of camera settings:

1. On the View Settings:Camera page (Figure 8-1), select the setting you want to move up or down in the order.
2. Click the up and down arrows next to the list of settings. To bind a setting to a hot key, move it up in the list until it has an index of 9 or lower.

8.7. Advanced Camera Settings

This section briefly describes the settings on the View Settings:Camera page that have not been discussed in the context of specific procedures.

8.7.1. Changing the Projection Mode

A projection mode defines how the terrain is rendered onto the 3D View. DI-Guy Scenario supports the following projection modes:

- Free camera. The default 3D view. The camera can move in all dimensions.
- Planview XY. A top-down view of the terrain. The camera can move only in the X and Y dimensions along the Z axis.
- Planview XZ. The camera can move in the X and Z planes along the Y axis.
- Planview YZ. The camera can move in the Y and Z planes along the X axis.
- Ortho XY. An orthographic view in the XY plane.

➤ To change the camera's projection mode, on the View Settings:Camera page (Figure 8-1), in the Projection Mode group box, select the projection mode you want.

8.7.2. Changing the Camera's Speed for Projection Modes

For the planview and orthographic projection modes, you can set the camera speed in meters per pixel.

➤ To change the camera's speed, on the View Settings:Camera page (Figure 8-1), in the Projection Mode group box, change the value in the Spd box.

8.7.3. Setting the Clipping Planes and Field of View

Clipping planes determine the bounds within which DI-Guy Scenario displays terrain data. The near clip specifies the distance from the camera at which it starts to display terrain. The far clip specifies the distance from the camera beyond which DI-Guy Scenario does not display any terrain. Limiting the range in which terrain is displayed can improve performance.

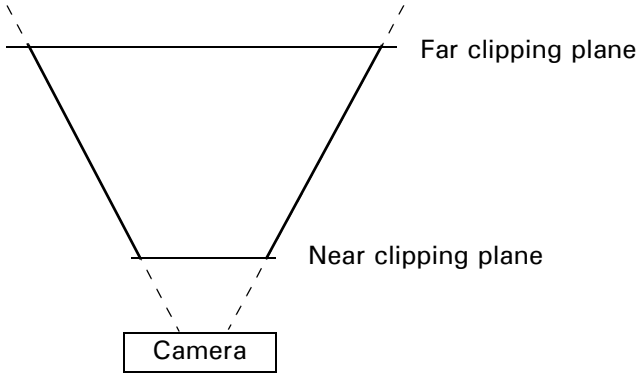


Figure 8-2. LOD zones

The field-of-view (FOV) is the virtual lens angle in degrees. A large value gives DI-Guy Scenario a wide-angle lens. A small value gives the camera a long lens.

➤ To specify clipping planes and FOV, on the View Settings:Camera page (Figure 8-1), change the values in the Near Clip, Far Clip, and FOV boxes.

8.8. Specifying Camera Override Preferences

Camera movement can be controlled by the Look At and Look From settings on the View Settings:Camera page. It can also be controlled by scripts. When such is the case, you cannot control the camera with the mouse. You can specify that if you set the input mode to camera and move the mouse into the 3D View, the mouse overrides other camera control settings.

To specify camera override preferences:

1. Choose **Edit** → **Preferences**. The Preferences dialog box opens.
2. Select the Camera tab (Figure 8-3).

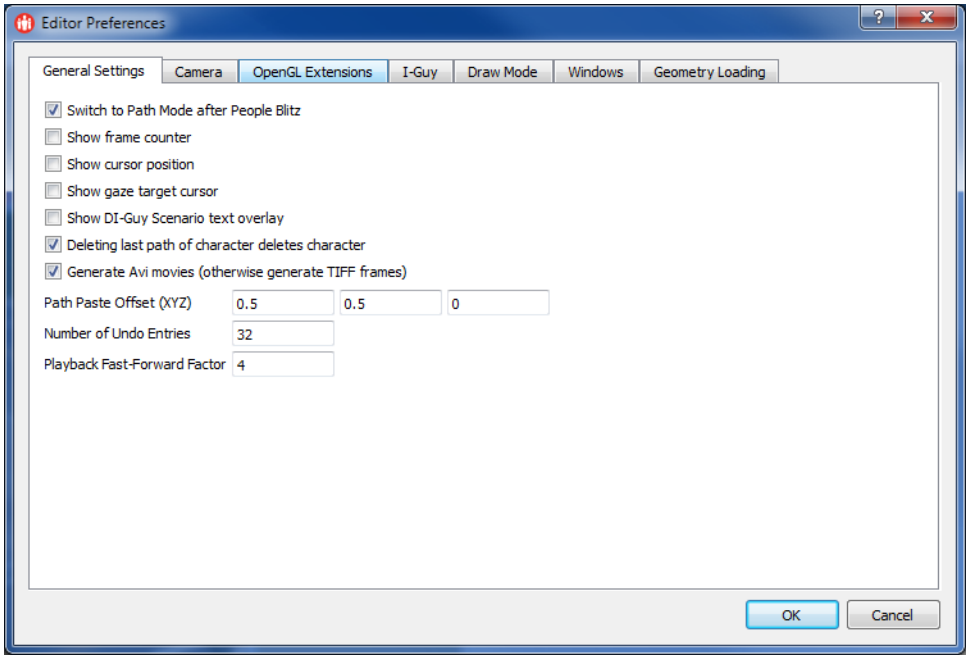


Figure 8-3. Camera tab

3. Change the settings as desired. For descriptions of the settings, please see Table 2-3.

8.9. Viewing a Scenario in Symbolic View

In symbolic view, characters are represented by 2D symbols. This option is available only if DI-Guy AI is supported.

➤ To change to symbolic view, on the View Settings:Camera page (Figure 8-1), select the Symbolic View check box.

9. Events and Event Handlers

This chapter explains how to create scenario-level decision logic and scripts and map them to event handlers.

Introduction to Events and Event Handlers 9-2
Event Types 9-3
Creating Scenario-Level Decision Logic 9-3
Converting Decisions to Scripts 9-6
Writing a Scenario-Level Script 9-7
Adding a Library Function 9-9
Mapping Events to Callbacks and Logic 9-11
Creating Information Popups 9-13
Opening an Info Popup from a Script 9-14
Creating an Interaction Machine 9-15
Example Interaction Machine 9-16
Hot Objects 9-18

9.1. Introduction to Events and Event Handlers

DI-Guy Scenario can respond to various types of scenario events by executing decision logic, running scripts, and calling DI-Guy SDK functions. All three methods draw on the SDK. They represent increasingly sophisticated ways to program the responses. Decision logic is created using a dialog box that guides you through available options based on previous choices. Scripting is done using the Lua programming language. Using SDK functions directly requires an understanding of the SDK.

All three approaches need to connect the decision logic or script to the events that they are supposed to respond to. This is done through a process called event mapping.

Figure 9-1 illustrates the relationships in event mapping. It assumes that you have written some decision logic, scripts, or library functions that can take some kind of action in a scenario. To get that decision logic to run, you must:

1. Select an instance of the type of event for which the decision logic was written.
2. Select the callbacks that you want to be activated when the instance of the event occurs.
3. Map each callback to a script, decision logic, or library function.

Please see "Creating Screen and Button Labels," on page 4-39 for an example of mapping scripts to labels.

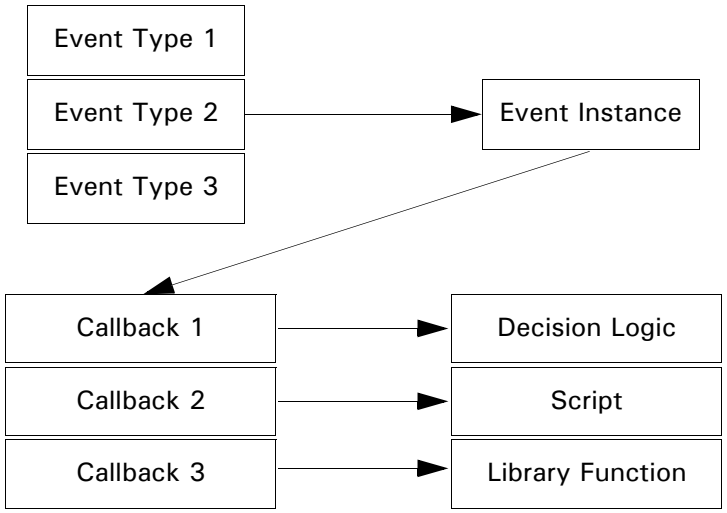


Figure 9-1. Events and event mapping

9.1.1. Event Types

Decisions, scripts, and function calls can execute in response to the following types of events:

- Manual. You trigger the decision logic manually.
- Timed. The decision logic is executed starting at time Trigger T, is checked every Trigger dt (or every timestep if Trigger Every Timestep is selected), and stops executing at Trigger T Out.
- Character or crowd. The decision logic is called every time a character or crowd satisfies a particular condition.
- Label. Clicking a label graphic in the 3D View triggers the decision logic.
- Scenario. The decision logic is called every time the scenario satisfies a particular condition.
- Sensor Region. The decision logic is called every time a particular condition is satisfied involving a sensor region.
- Signal. The decision logic is executed when a signal is received.
- Variable. The decision logic is executed when a variable changes.

9.2. Creating Scenario-Level Decision Logic

The Event Handlers:Decision page (Figure 9-2) has a GUI that lets you create scenario-level decision logic without programming. You can convert decisions to scripts.

All decisions use an “If…Then…Else” logic. However, you can shortcut the process by combining the decision bead parameters with the decision logic. For example, if you know you want to activate a particular camera at time t=10, set the decision bead to have a beginning time of 10.0. Then in the logic, just say “Then camera…” and so on. Not only are you not using the Else, you are not even using the If!

The Decision Editor is a front-end to the DI-Guy API, so the boxes refer to the objects, functions, and arguments that API functions require. At each stage of use, the Decision Editor dynamically shows the available objects, operators and arguments.

The following procedure describes a basic If, Then, Else logic. However, you can add multiple rows to each section, which would use AND or OR logic. You can change the order of the statements after you have added them and you can delete rows as well.

To create scenario-level decision logic:

1. Select the Event Handlers:Decision page (Figure 9-2).

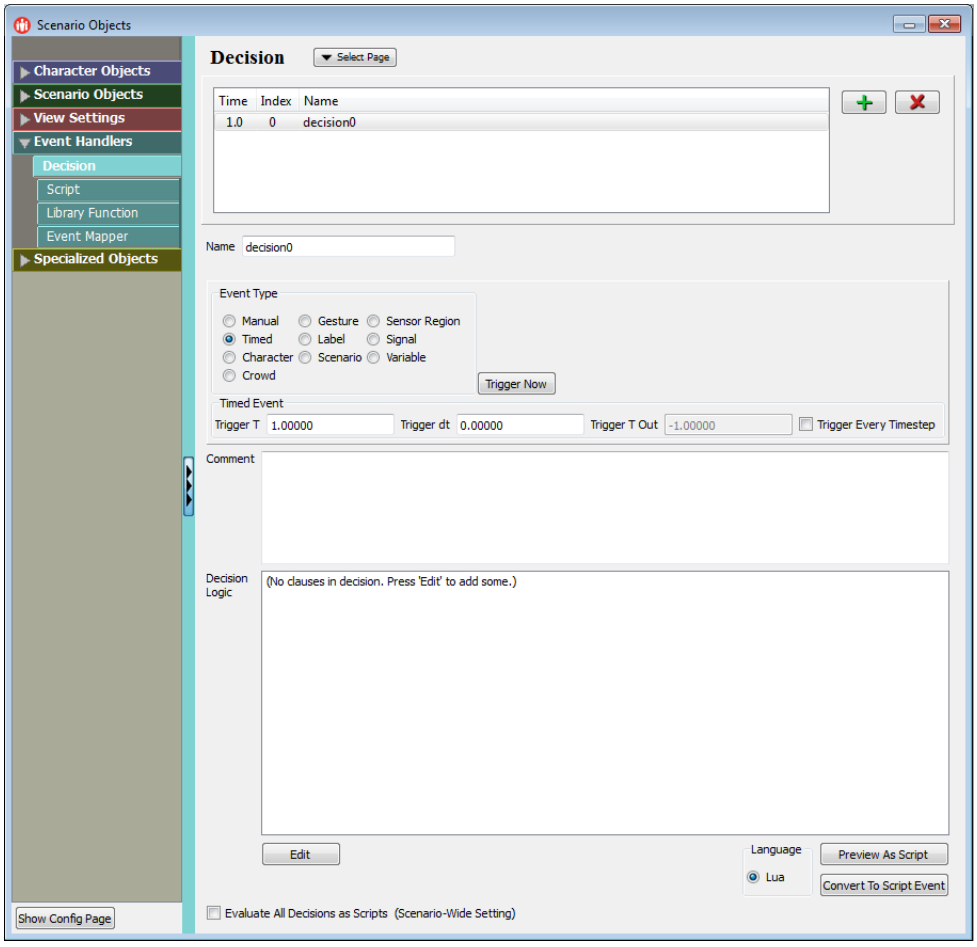


Figure 9-2. Decision page

2. Click the Add button (+). A new decision is added to the list.
3. Optionally, type a new name for the decision.
4. In the Event Type group box, select the event type for this decision.

5. Specify the duration and tick rate of the event with the following parameters:
   - Trigger T. The time at which to start executing the decision. Mutually exclusive with Trigger Every Timestep.
   - Trigger dt. The frequency at which to check for the event. Mutually exclusive with Trigger Every Timestep.
   - Trigger T Out. The time at which to stop executing the decision logic.
   - Trigger Every Timestep. Check every timestep. Mutually exclusive with Trigger T and Trigger dt.
6. Click Edit. The Decision Editor opens (Figure 9-3).

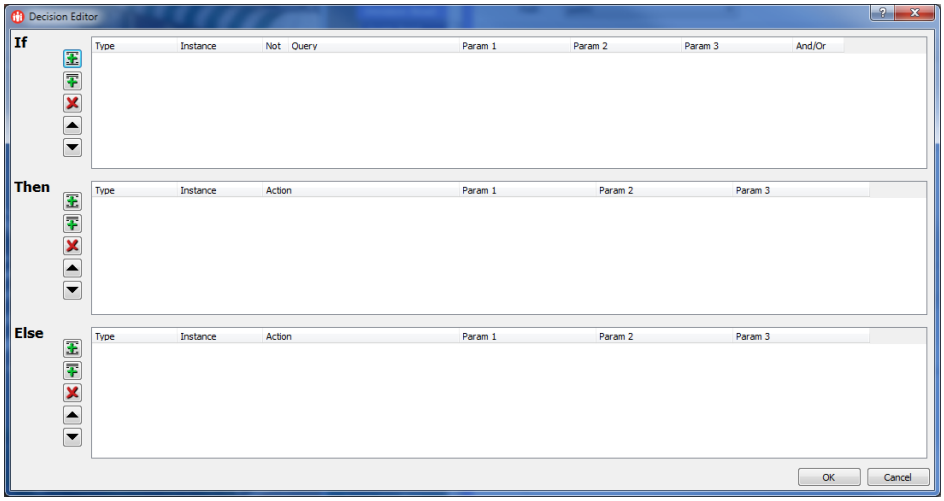


Figure 9-3. Decision Editor

7. In the If section, click the Add button (+). A list gets added to the first row in the window. Select an option from the list. A list gets added to the next column that is appropriate for the selection you made in the first column.
8. Build the complete If condition by selecting options from lists as they get added.

The If expression is not absolutely required. It is possible to trigger a decision based solely on the time or distance parameters. However, most decisions use an If statement.

9. Optionally, add additional If lines using AND or OR logic.
10. In the Then section, click the Add button. A list gets added to the first row in the window. Select an option. A new list gets added to the next appropriate column.
11. Build the complete Then expression by selecting options from the lists as they get added.

12. Optionally, add additional Then lines using AND or OR logic.
13. Optionally, click the Add button in the Else section. Build the Else condition the way you built the If condition.
14. Optionally, add additional Else lines using AND or OR logic.
15. Click OK. The completed decision script is displayed in the Decision Logic window. For an example of a simple decision logic, please see "Creating Character-Level Decision Logic," on page 6-16.

9.2.1. Converting Decisions to Scripts

Decisions are relatively easy to construct and the GUI ensures that you do not have to worry about syntax or mismatched parameters and other pitfalls of using a programming language. However, the process also imposes some constraints on the logic that you can write. DI-Guy Scenario also supports a complete Lua scripting environment for scripting character behaviors. You can convert decision logic to Lua scripts.

When you convert a decision logic to a script, it is converted, not copied. The decision logic will no longer exist.

To convert a decision to a script:

1. Select the Event Handlers:Decision page (Figure 9-2).
2. Select the decision you want to convert.
3. Optionally, click Preview As Script to see the Lua code.
4. Click Convert to Script Event. The decision bead is converted to a script. The script is added to the Event Handlers:Script page (Figure 9-4).

9.3. Writing a Scenario-Level Script

DI-Guy Scenario supports scripting using the Lua scripting language. You can access most of the DI-Guy API inside of scripts. You can get at all the other elements in the scenario, including the other characters and scenario objects. The DI-Guy API is described in *DI-Guy SDK Developers Guide* and the DI-Guy SDK class documentation.

Script events allow you to write complex decision logic for the events and entities of a scenario. Script events use Lua code to specify the logic.

Converting a decision into a script is an easy way to get started with Lua scripting. You will be able to examine the syntax of logic that you already understand.

It is beyond the scope of this manual to teach Lua scripting. There are many good resources online and in print. A good reference on the Lua scripting language is Programming in Lua (http://www.lua.org/pil/). For a brief introduction to the Lua code editor embedded in the script window, please see Appendix D, *Using the Lua Editor*.

To create a scenario-level script:

1. Select the Event Handlers:Script page (Figure 9-4).

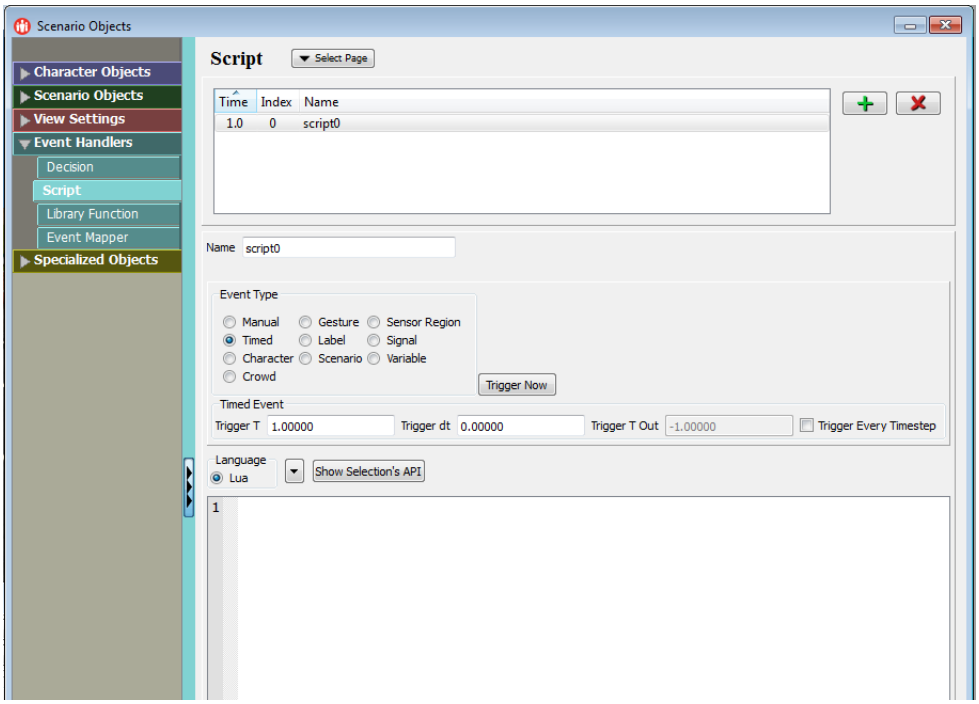


Figure 9-4. Script page

2. Click the Add button (+). A script is added to the list.
3. Optionally type a name in the Name box.

4. In the Event Type group box, select the event type for this script.
5. Specify the duration and tick rate of the event with the following parameters:
   - Trigger T. The time at which to start executing the decision. Mutually exclusive with Trigger Every Timestep.
   - Trigger dt. The frequency at which to check for the event. Mutually exclusive with Trigger Every Timestep.
   - Trigger T Out. The time at which to stop executing the decision logic.
   - Trigger Every Timestep. Check every timestep. Mutually exclusive with Trigger T and Trigger dt.
6. Optionally, click the down arrow above the script window to set scripting parameters.
7. In the script window, write the script. If you select a function in the window and click Show Selection's API, API documentation for the function is displayed. If nothing is selected, after a warning prompt, top-level documentation opens.

9.4. Adding a Library Function

The DI-Guy SDK includes dynamically linked libraries (DLLs) that get loaded when you run DI-Guy Scenario. SDK users can create additional libraries that may be available to you. You can access the functions in these libraries through a library function.

The library functions must match the standard DI-Guy function prototype.

To add a library function:

1. Select the Event Handlers:Library Function page (Figure 9-5).

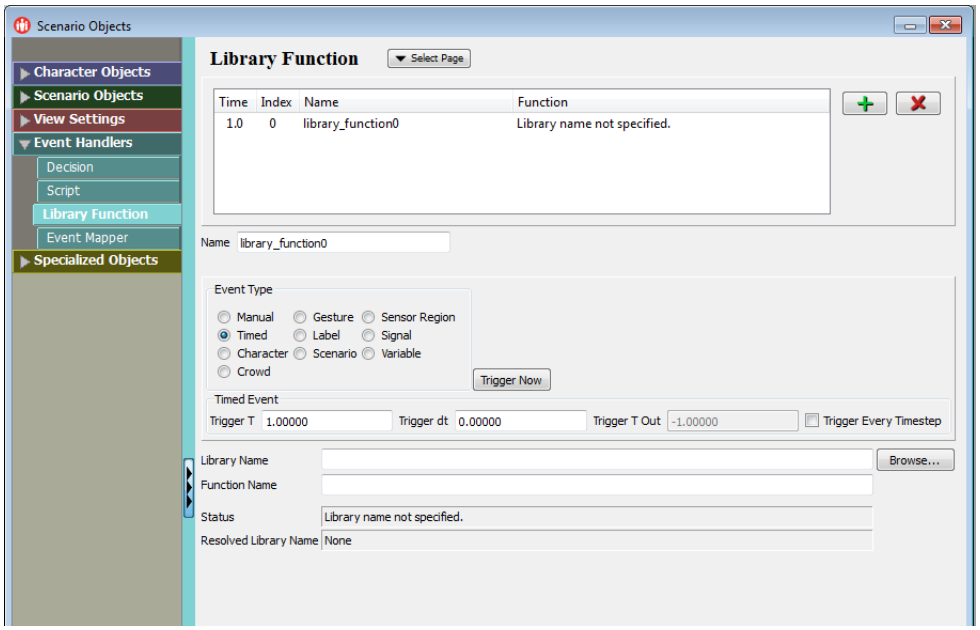


Figure 9-5. Library Function page

2. Click the Add button (+). A new library function is added to the list.
3. Optionally, type a new name for the function.
4. In the Event Type group box, select the event type for this library function.
5. Specify the duration and tick rate of the event with the following parameters:
   - Trigger T. The time at which to start executing the decision. Mutually exclusive with Trigger Every Timestep.
   - Trigger dt. The frequency at which to check for the event. Mutually exclusive with Trigger Every Timestep.
   - Trigger T Out. The time at which to stop executing the decision logic.
   - Trigger Every Timestep. Check every timestep. Mutually exclusive with Trigger T and Trigger dt.

6. In the Library Name box, type the name of the library (DLL), or click Browse and select it from the file browser.
7. Type the name of the function in this library that you want to use when the event occurs.

9.5. Mapping Events to Callbacks and Logic

The Event Mapper maps events generated from characters, scenarios, sensor regions, or signals to decisions, scripts, or library functions. (For background, please see "Introduction to Events and Event Handlers," on page 9-2.) The mapping process itself is relatively simple. The trickier part is understanding which events, callbacks, and logic to map to each other. For a partial list of callbacks, please see Appendix C, *Callback IDs*.

For an example of mapping callbacks to decision logic, please see "Creating Screen and Button Labels," on page 4-39.

To map events to callbacks and logic:

1. Select the Event Handlers:Event Mapper page (Figure 9-6).

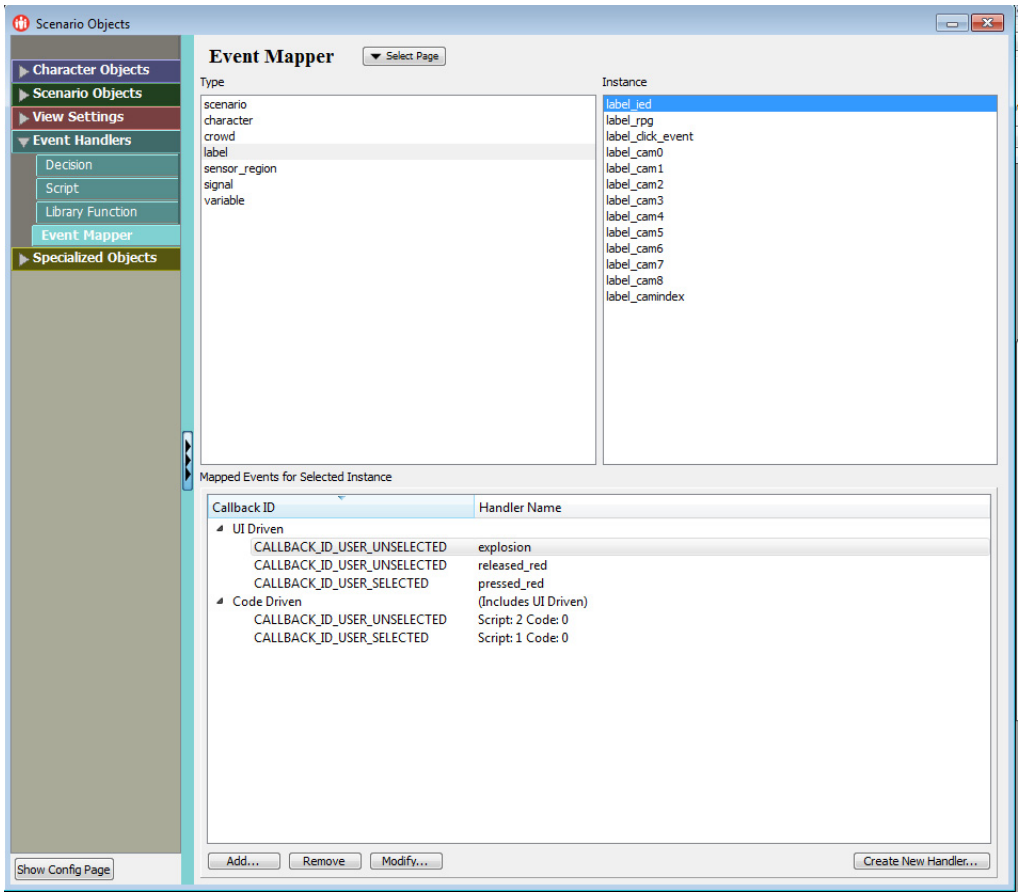


Figure 9-6. Event Mapper page

2. In the Type list, select the type of event you are trying to map (for example, character, scenario, sensor region, or signal). The instance window lists all the available object instances of the type selected. For example, if you select character, it lists all the characters in the scenario.

3. Select the instance of the event type that you want to map.
4. Click Add. The Event Mapping Editor opens (Figure 9-7). It lists the callbacks that are available for the event type you selected and all the decisions, scripts, and library functions that you have designated as event handlers for this particular event type.

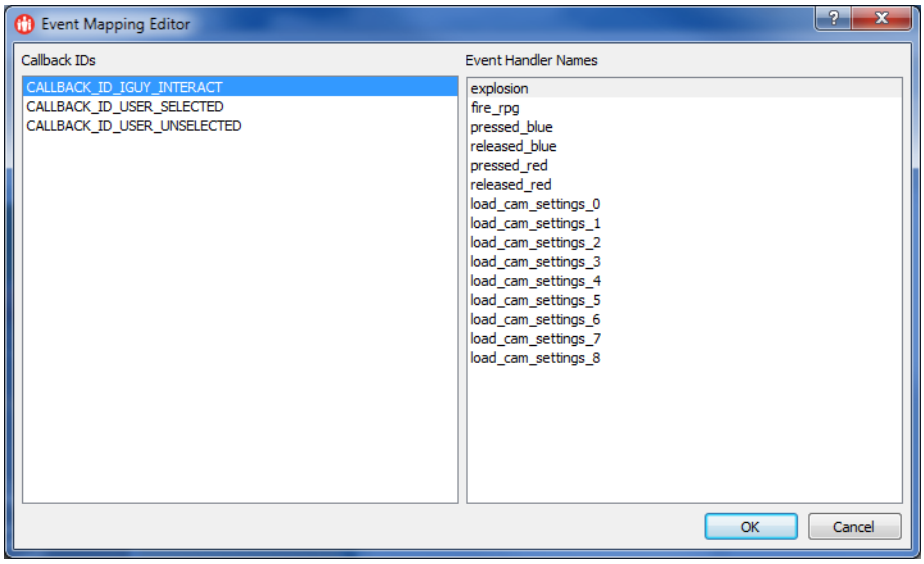


Figure 9-7. Event Mapping Editor

5. Select the Callback ID you are interested in mapping.
6. Select the Event Handler to which the callback is to be mapped.
7. Click OK. The mapping is added to the Event Handler page.
8. Repeat this process for as many callbacks as you need to map.

9.6. Creating Information Popups

Info Popups are windows used to display text and images as messages during a scenario. Info Popup windows can be triggered using decision logic, script beads, or script events. Info Popup windows display a subset of HTML.

Info_Popup_demo.dss is an example scenario that uses Info Popup windows.

To create an info popup:

1. Select the Specialized Objects:Info Popup page (Figure 9-8).

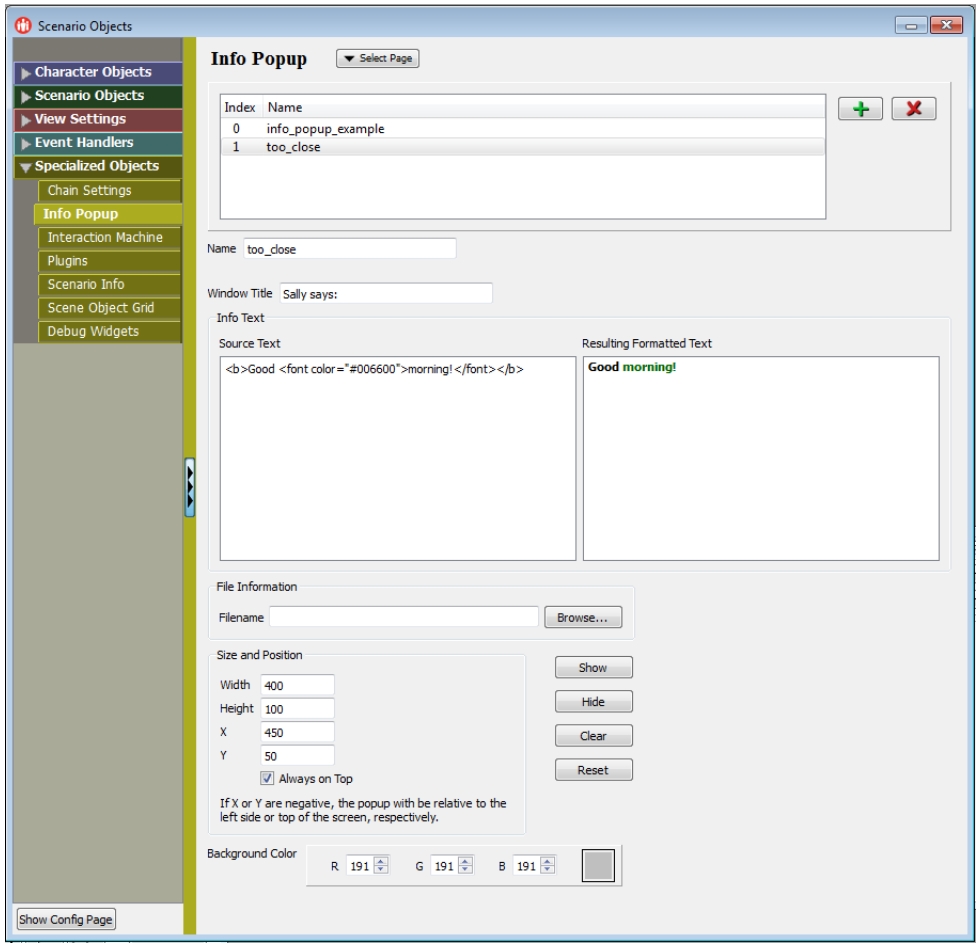


Figure 9-8. Info Popup page

2. Click the Add button (+). A new info popup is added to the list.
3. Optionally type a name for the info popup.
4. In the Window Title box, type the text that you want to be displayed in the title bar of the popup window.

5. In the Source Text window, type HTML code with the text you want to display in the popup window. The Resulting Formatted Text window shows how the text will look in the popup window.
6. Optionally, specify an HTML file to include in the popup window. (The text is not shown in the Info Text windows, it is added to the Source Text.)
7. Optionally, specify the width, height, and location of the popup window.
8. Optionally, change the background color.
9. Click Show to see what the popup window will look like.

9.6.1. Opening an Info Popup from a Script

Once you have defined your info popups and created the HTML, you can cause them to be displayed within a scenario using a decision bead, a script bead, or a script event. Here is the Lua script needed to display an info popup called Hello using a script bead:

```
local info_popup_hello = this_scenario:find_info_popup("Hello");
if (not info_popup_hello) then
  print("info_popup hello not found\n");
  return;
else
info_popup_hello:show();
end
```

9.7. Creating an Interaction Machine

Interaction Machines give scenario developers a way of providing interactive, UI-driven input to a scenario that can change the way a scenario progresses. They are good for use in self-training and avatar-based scenarios by presenting you 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.

Interaction machines are implemented as dialog boxes in DI-Guy Scenario (Figure 9-9). Each one has the following areas:

- Heading. Provides a one-line title for the machine.
- Informational text. Describes the current state. The information text can be coded in plain text or simple HTML-style format.
- Input. Presents the user with options they can click to proceed to a new state.

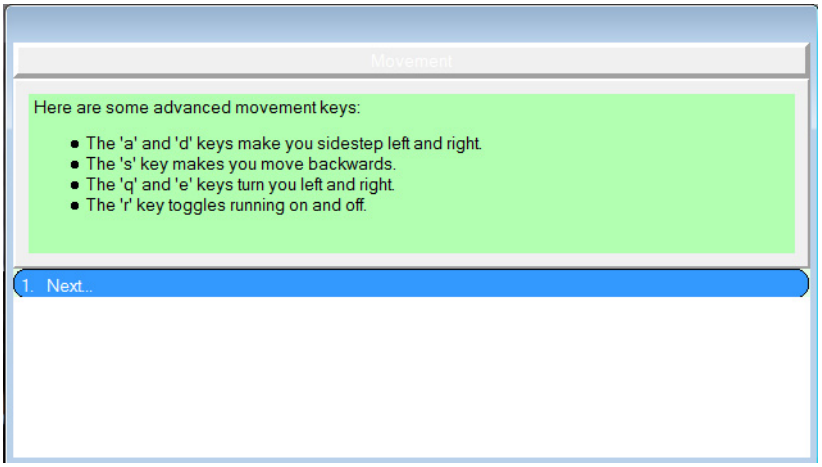


Figure 9-9. Interaction machine dialog box

9.7.1. Example Interaction Machine

DI-Guy Scenario includes a scenario, *Iguy_Interaction_Demo.dss*, that demonstrates how interaction machines work.

The scenario starts with a series of click-through screens that explain how to control the avatar. To understand how to set this up, we will look at the im_tutorial_move interaction machine and script.

The im_tutorial_move interaction machine is defined in the Specialized Objects:Interaction Machines page (Figure 9-10).

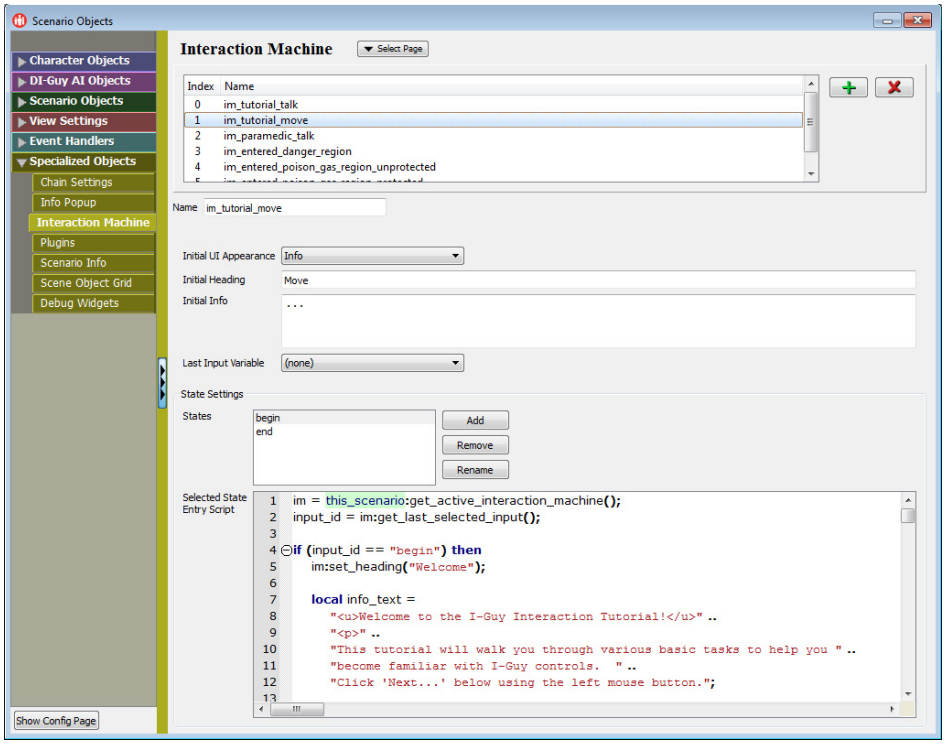


Figure 9-10. Interaction Machine page

The code for the beginning dialog box of the interaction machine is as follows:

```
1     im = this_scenario:get_active_interaction_machine();
2     input_id = im:get_last_selected_input();
3
4     if (input_id eq "begin") then
5       im:set_heading("Welcome");
6
7       my info_text =
8         "<u>Welcome to the I-Guy Interaction Tutorial!</u>" ..
9         "<p>" ..
10        "This tutorial will walk you through various basic tasks to help
        you " ..
11        "become familiar with I-Guy controls. " ..
12        "Click 'Next...' below using the left mouse button.";
```

```
13
14      im:set_info(info_text);
15
16      im:clear_inputs();
17      im:add_input("movement", "Next...");
18
19      im:update_ui();
20    elseif (input_id eq "movement") then
```

Lines 1 and 2 are very common to most interaction machine scripts. They retrieve the current interaction machine and the last input into the machine. If there has been no input yet, the input is returned as “begin”.

The input(s) are user selectable choices, with the first argument being the input_id back into interaction machine, and the second argument being the actual text displayed. The inputs are numbered automatically so you do not need to put a number in when you have multiple choices. When the user selects an input, the interaction machine is called with the new input_id.

The interaction machine is called from a script on the Event Handlers:Script page. This script starts up almost immediately at 0.1 seconds. The script has two key commands. The first is:

```
im = this_scenario:find_interaction_machine ("im_tutorial_move");
```

This line gets a pointer to the im_tutorial_move interaction machine:

```
im:begin_interaction();
```

This line makes the first call into it and brings up the interaction UI.

For more information about the DI-Guy Scenario Interaction Machine, consult the SDK reference manual for the class *diguyInteractionMachine*.

9.7.2. Hot Objects

The IGuy_Interaction_Demo also features clickable “hot objects”.

Figure 9-11 illustrates a label that instructs the user to click the paramedic to talk.

Figure 9-11. Hot object

If we look at the Event Handler:Event Mapper page, and select character in the Event window and Paramedic in the Instance window, you can see that the callback im_paramedic_talk is invoked when the event IGUY_INTERACT happens (Figure 9-12).

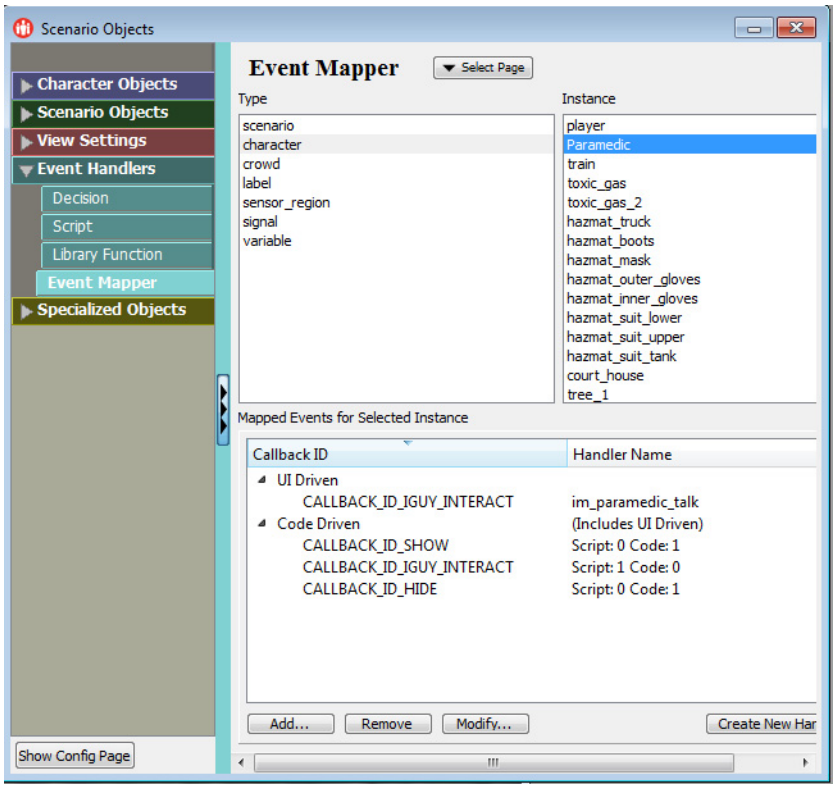


Figure 9-12. Event Mapper page for hot object

In other words, if you are in I-Guy input mode and you click anywhere on the character Paramedic, a callback gets sent to the function im_paramedic_talk.

Therefore, if you want to make a hot object that has some sort of action occur when you click on it, you need only add this event handler to the character.

10. Configuring Performance

This chapter explains how to optimize the performance of DI-Guy Scenario.

Performance Tuning 10-2
Setting the Tick Rate 10-3
Specifying the Default Number of Ticks between Checks 10-4
Skipping Ticks 10-4
LOD Tuning 10-4
LOD Tuning for Terrain 10-5
Graphics Levels of Detail 10-6
Demand Loading for Terrains 10-6
Culling and Draw Management 10-8
Camera Frustum Culling 10-9
Distance Culling 10-10
Load Manager 10-10
Specifying the Drawing Preferences 10-12
Specifying Geometry Loading Preferences 10-14

10.1. Performance Tuning

DI-Guy Scenario's performance depends on several factors, including the speed of your computer, the speed of your 3D graphics card, amount of memory, the size and efficiency of the terrain model (scene object), and the complexity of the scenario. There are several parameters to help you balance load and tune DI-Guy Scenario performance for your application. Some of these parameters only affect use of DI-Guy Scenario, and some affect performance when scenarios are exported to run in an application using an embedded DI-Guy Scenario.

10.2. Setting the Tick Rate

DI-Guy Scenario tries to recalculate behavior (update the scenario) for characters at a frequency defined as **Tick_dt**. Depending on the complexity of the scenario, terrain model, amount of decision logic, and camera views, DI-Guy Scenario may be able to process the scenario in real time at the specified rate, or it may fall behind during portions of the scenario. You can improve the uniformity of performance by selecting **Tick_dt** to be as large as possible while satisfying the needs of your application. Using a **Tick_dt** of less than .03 can result in bad performance, since changes that fast are undetectable by the human eye.

To set the tick rate:

1. Select the Scenario Objects:Performance page (Figure 10-1).

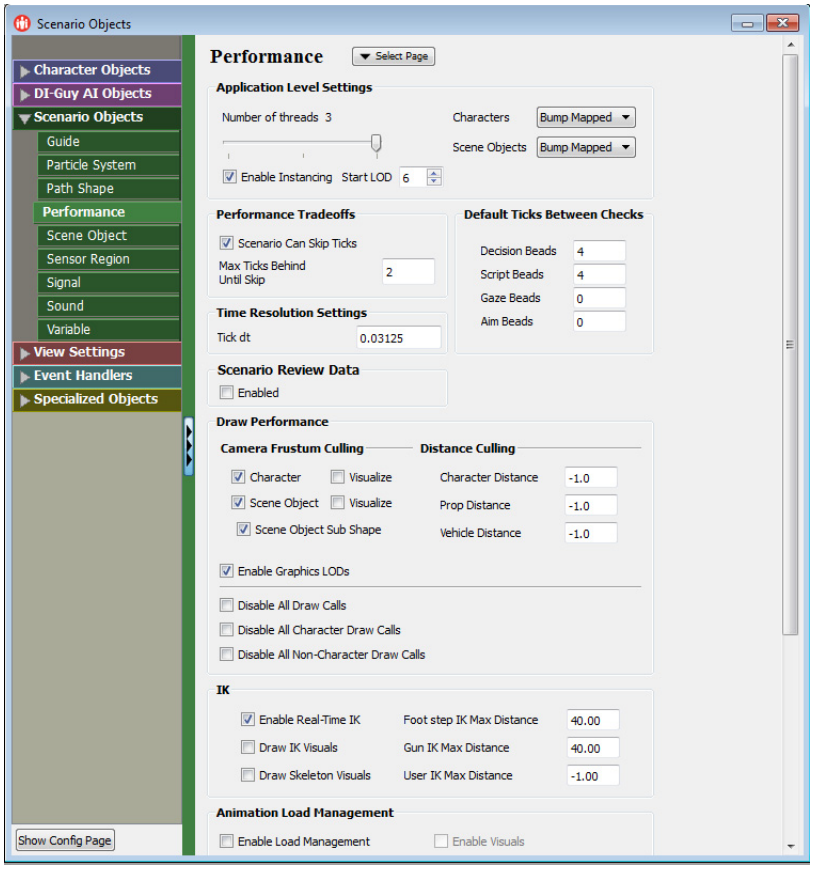


Figure 10-1. Scenario Objects:Performance page

2. In the Time Resolution Settings group box, set the Tick dt value.

10.2.1. Specifying the Default Number of Ticks between Checks

Decision beads, script beads, gaze beads, and aiming beads can reduce DI-Guy Scenario performance. Performance is affected the most when the conditions that trigger these activities span an extended period of time, rather than trigger on a single event. Often, these beads do not need to be checked every tick. You can improve performance by increasing the number of ticks between each check of a condition.

When you create a decision bead, script bead, gaze bead, or aiming bead, its Ticks Between Checks value is set to the default. You can change the ticks between checks on a per-bead basis.

To set the default number of ticks between checks:

1. Select the Scenario Objects:Performance page (Figure 10-1).
2. In the Default Ticks Between Checks group box, specify a number of ticks for each bead type.

10.2.2. Skipping Ticks

If DI-Guy Scenario cannot run at real time, it can skip ticks to try to catch up. This may result in loss of data or discontinuity of remote characters if you are connected over the network. Tick skipping is enabled by default.

To skip ticks:

1. Select the Scenario Objects:Performance page (Figure 10-1).
2. In the Performance Tradeoffs group box, select Scenario Can Skip Ticks.
3. In the Max Ticks Behind Until Skip box, specify the number of ticks that DI-Guy Scenario can fall behind before it skips ticks.

10.3. LOD Tuning

Levels of detail (LODs) is a performance mechanism in which objects in the scene are displayed at higher resolution when they are close to the camera and lower resolution when they are farther away. Viewing objects at lower resolution requires fewer graphic resources, so you can improve performance by reserving high resolution for objects that you care most about – those that are close to the camera.

LOD tuning can be applied to the terrain and to characters.

10.3.1. LOD Tuning for Terrain

The LOD Range Scale Factor parameter allows you to adjust the LOD switching ranges for terrains. If you set the value of this parameter to 0.5, then each LOD switch occurs at half the distance specified in the terrain model. That results in fewer terrain elements being displayed at high resolution and, consequently, better performance.

To set the range scale factor:

1. Select the Specialized Objects:Scene Object Grid page (Figure 10-2).

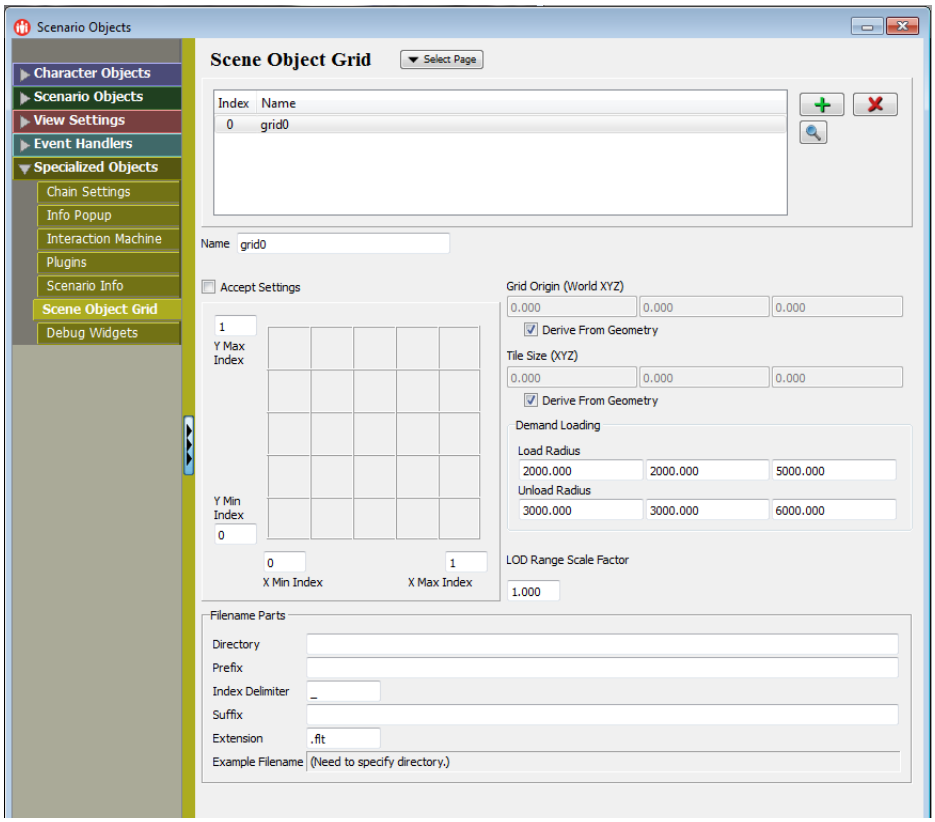


Figure 10-2. Scene Object Grid page

2. Select the scene object that you want to tune.
3. In the LOD Range Scale Factor box, enter a value.

10.3.2. Graphics Levels of Detail

DI-Guy Scenario provides seven LODs for most of its character appearances to help provide high rendering performance. You can enable or disable graphics LODs.

To enable or disable graphics LODs:

1. Select the Scenario Objects:Performance page (Figure 10-1).
2. Select or clear the Enable Graphics LODs check box.

10.4. Demand Loading for Terrains

You can manage large scene objects to improve performance using demand loading. An axis-aligned bounding box is computed for each scene object as it is loaded. The following algorithm is then applied:

- If the camera is further from the bounding box than the unload radius and if the scene object is currently being displayed, DI-Guy Scenario hides the scene object.
- If the camera is closer to the bounding box of the scene object than the load radius and if the scene object is not displayed, then DI-Guy Scenario displays the scene object. DI-Guy Scenario loads the scene object's geometry if it is not already in memory.

The unload radius should be greater than or equal to the load radius. Having separate load and unload radii makes it possible to avoid having a scene object flash on and off as it would if the camera hovered right on the border of a combined load/unload radius.

To configure demand loading:

1. Select the Scenario Objects:Scene Object page (Figure 10-3).

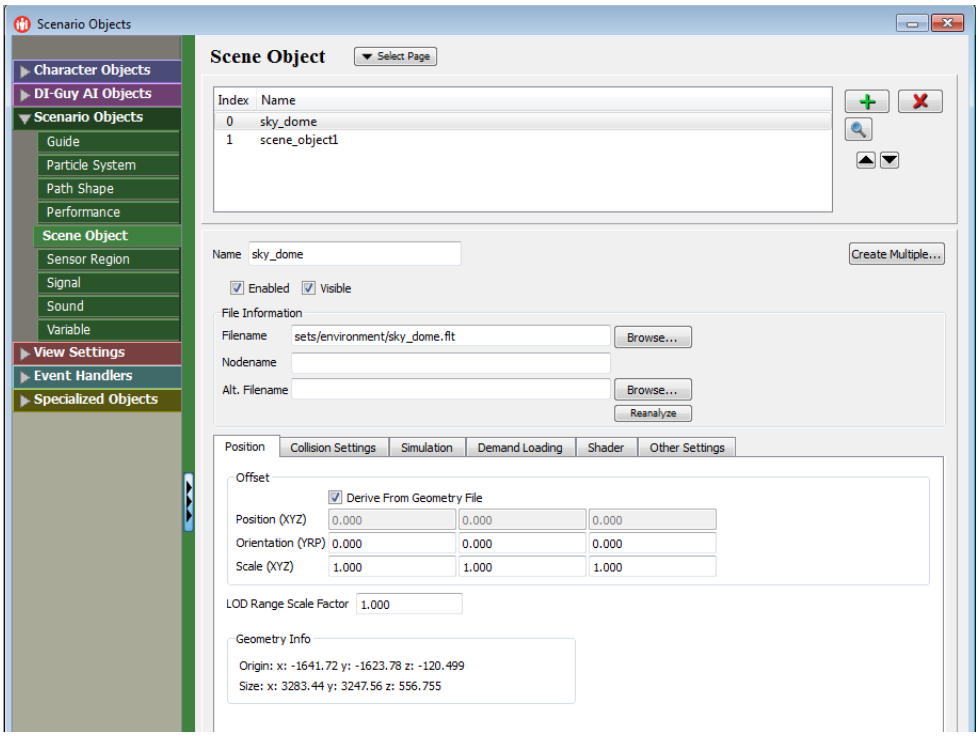


Figure 10-3. Scene Object page

2. Select the scene object that you want to tune.
3. Select the Demand Loading tab (Figure 10-4).

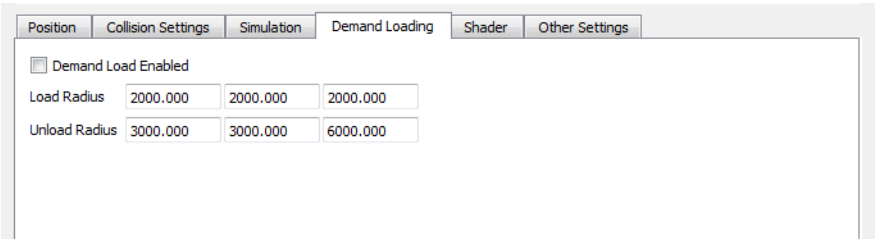


Figure 10-4. Demand Loading tab

4. Select the Demand Load Enabled check box.
5. Specify the Load Radius and Unload Radius. The values are X, Y, and Z, in meters, from the perspective of the camera.

10.5. Culling and Draw Management

You can improve performance by rendering only those objects that are within the view of the camera (frustum culling) or within a certain distance of it (distance culling). This is in addition to any LOD tuning that may be in effect.

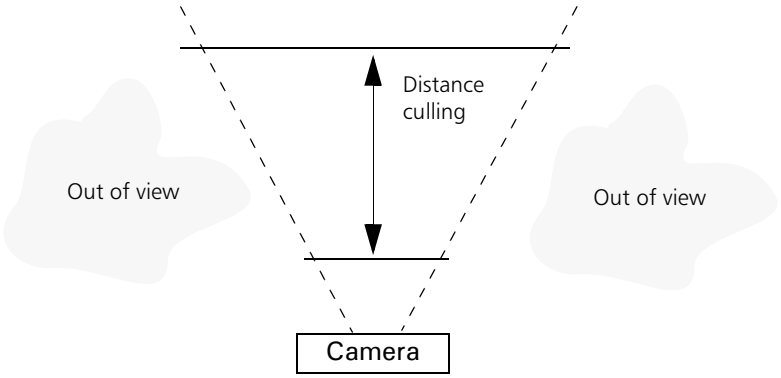


Figure 10-5. LOD zones

You can also enable and disable all drawing, all character drawing, and all non-character drawing.

10.5.1. Camera Frustum Culling

You can specify that characters and scene objects that are not within the camera's view frustum, that is, characters that you cannot see, not be rendered. Since you cannot see them, there is no point in spending any computer resources on rendering them. Frustum culling is enabled by default.

Camera frustum culling is performed based on an extent boundary applied to each character and object. You can visualize the extents for debugging purposes. Figure 10-6 shows several characters with extents visualized. Notice the extent visible to the left of the intersection where a character is not visible.

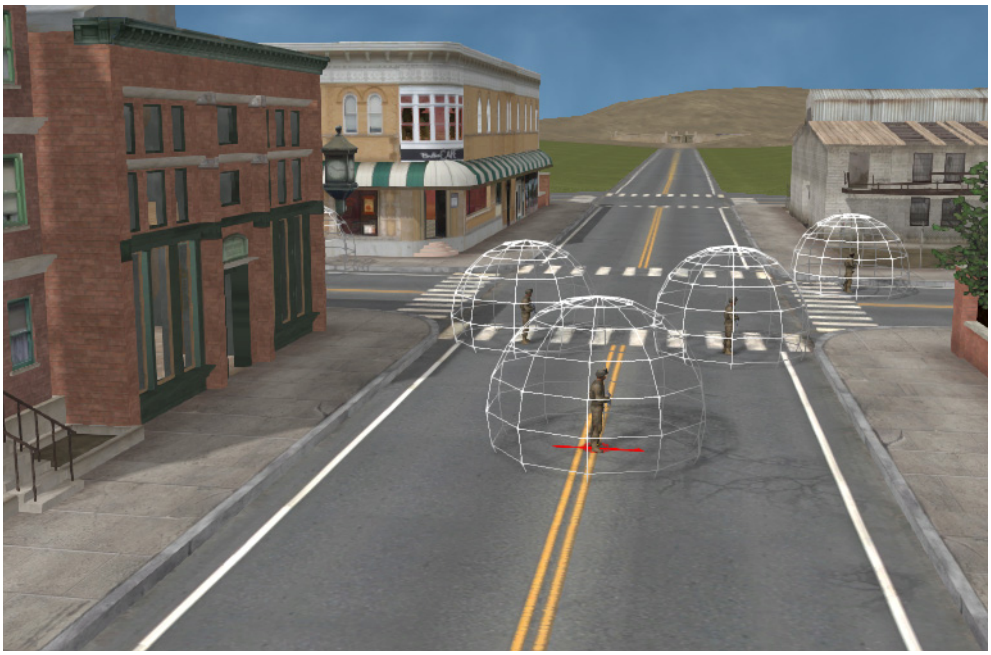


Figure 10-6. Character extents

In addition to culling out entire scene objects, you can cull out object sub shapes. This may be useful if you have very large objects that are unlikely to fall out of the view frustum, but which have sub shapes that could be profitably culled out if they are not being viewed.

To enable or disable frustum culling:

1. Select the Scenario Objects:Performance page (Figure 10-1).
2. In the Draw Performance group box, Camera Frustum Culling column, select or clear the Character, Scene Object, or Scene Object Sub Shape check boxes.
3. Optionally, select the Visualize check box for characters of scene objects to see their extents.

10.5.2. Distance Culling

Distance culling culls out characters, props, and vehicles based on their distance from the camera. The default, -1.0, means there is no culling.

To cull objects by their distance from the camera:

1. Select the Scenario Objects:Performance page (Figure 10-1).
2. In the Draw Performance group box, Distance Culling column, type a value in any of the Character Distance, Prop Distance, and Vehicle Distance boxes.

10.6. Load Manager

The Load Manager lets you make performance improvements when working with scenarios that contain many characters. It only works for single view situations. If you are using multiple views, do not use the Load Manager.

The viewing world is divided into two regions: the visible region and the nonvisible region (Figure 10-7). The visible region (that is, the view frustum) is divided into three zones, zone 0, 1, and 2. (Think of them as near, far, and very far.)

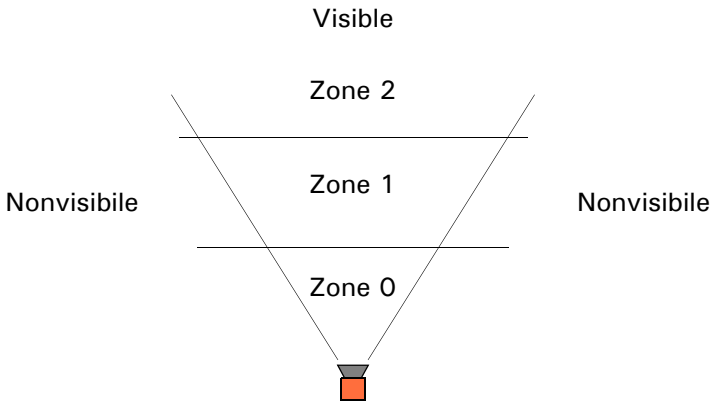


Figure 10-7. Load management zones

Although you do not want to render objects that are not visible, you must still keep track of their location in case they enter the visible zone. However, it may not be necessary to check the location of nonvisible characters as often as you check visible characters.

When load management is enabled, you can specify the extents of the zones and how frequently DI-Guy Scenario updates the characters in those zones and the nonvisible region. Load management is configured with the following parameters:

- Zone Far Extent. Specifies the distance from the camera, in meters, at which Zone 0 and Zone 1 end. There is no need to specify a distance for Zone 2. It is infinite.
- Position Update Rate. Specifies the percentage of frames during which a character's position is updated. In the visible region, this should almost always be 100%.
- Pose Update Rate. Specifies the percentage of frames during which the non-position joints are updated.
- Altitude Update Rate. Specifies the percentage of frames during which a character's altitude is updated.

To configure load management:

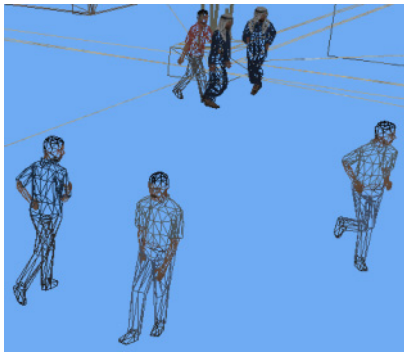
1. Select the Scenario Objects:Performance page (Figure 10-1).
2. In the Animation Load Management group box, select the Enable Load Management check box.
3. Optionally, select Enable Visuals to see the zone boundaries drawn on the screen.
4. Update the Zone Far Extent and update rate values as desired.
5. Optionally, to disable character animations, select the Disable All Character Animations check box.

10.7. Specifying the Drawing Preferences

The Editor Preferences dialog box, Draw Mode tab has settings that affect rendering quality and visual quality. Reducing rendering quality can improve performance, although at the cost of the visual quality of the scenario.

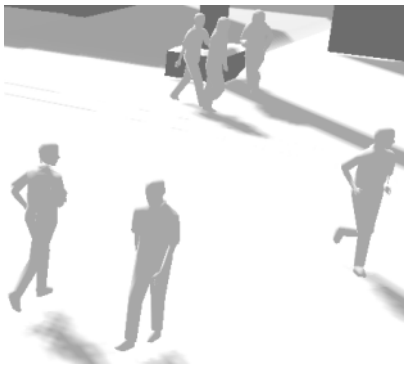
The Polygon Fill Mode settings affect how characters are drawn. Figure 10-8 illustrates the different fill modes along with enabled or disable textures.

Normal draw mode with textures



Outline draw mode

Scribed draw mode



Normal draw mode, no textures

Figure 10-8. Draw modes

To specify drawing mode preferences:

1. Choose **Edit** → **Preferences**. The Editor Preferences dialog box opens.
2. Select the Draw Mode tab (Figure 10-9).

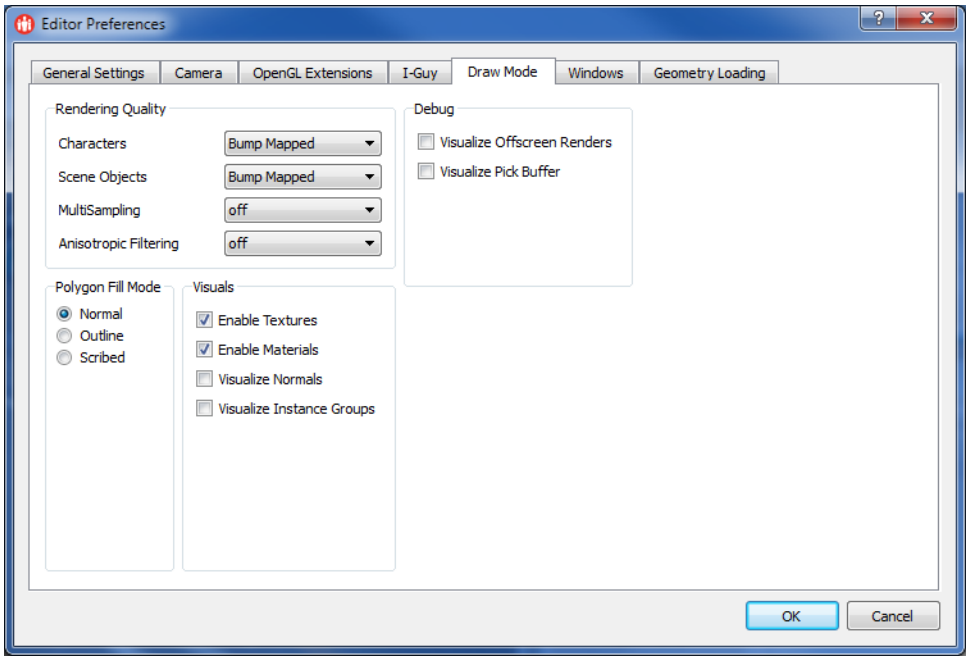


Figure 10-9. Draw Mode tab

3. Change the settings as desired.

10.8. Specifying Geometry Loading Preferences

DI-Guy Scenario has preference settings for optimizing geometry files, using compressed textures, and memory management.

To specify geometry loading preferences:

1. Choose **Edit** → **Preferences**. The Editor Preferences dialog box opens.
2. Select the Geometry Loading tab (Figure 10-10).

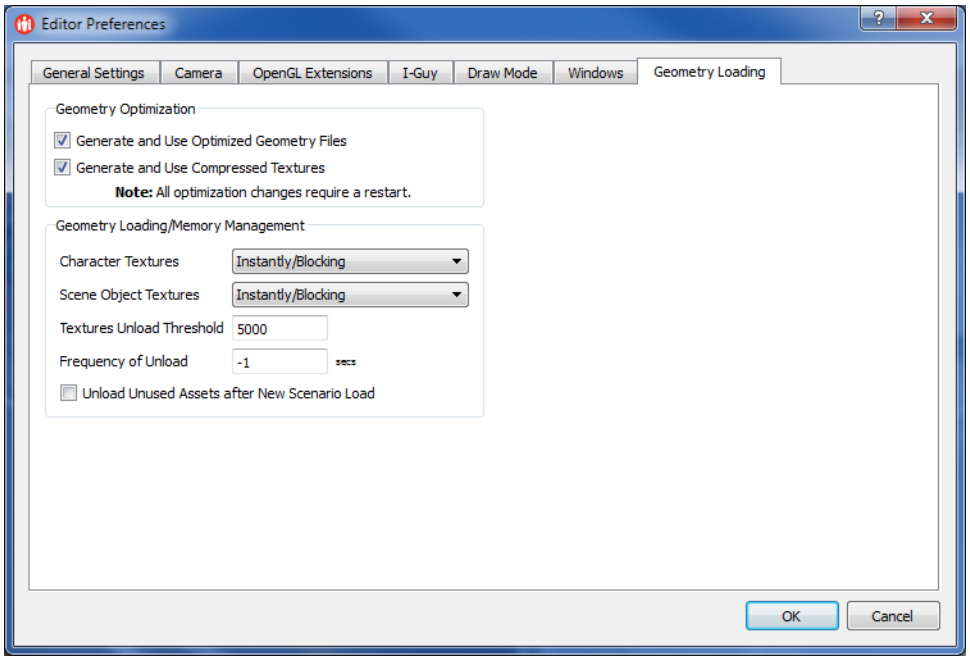


Figure 10-10. Geometry Loading tab

3. Change the settings as desired.

11. Terrains and Environment

This chapter explains how to configure and load terrains and how to set environmental effects.

Terrain Models 11-2
Loading a Terrain 11-4
Positioning the Terrain Database on the Earth's Surface 11-5
UTM and the Exercise Location 11-7
Paging Terrains 11-8
LOD Tuning 11-9
Displaying Terrain Coordinates for an Object 11-10
Configuring Fog 11-11
Adding a Fog Setting 11-12
Editing Fog Settings 11-13
Configuring Lighting Settings 11-13
Adding a Light Setting 11-14
Editing Lighting Settings 11-15

11.1. Terrain Models

Terrain models include the ground, roads, trees, buildings, and furnishings that make up a space. Typically the first thing you do when you create a scenario is to load a terrain. By default, when you create a new scenario, it loads the Stage terrain database. The Stage is a simple model that has a small city intersection and a walled mid-east town (Figure 11-1).

Figure 11-1. DI-Guy default terrain

DI-Guy Scenario includes several terrain models that you can use for demonstrations and to experiment with. They are in *./geometry/sets*. The models are:

- Centennial Hall. A model, developed by the Urban Simulation Team at UCLA, of some buildings on the UCLA campus.
- Convention Center. A simple model including a hotel-style function room.
- Default Stage 11. A few blocks of a small town.
- Default Stage 7. A combination of a park, warehouse, and a street.
- Default Stage. A very simple house on a very simple road.
- Desert.
- Environment sky dome.
- Jailhouse.
- Khost Afghanistan.
- Middle East Town.
- NAWC Rocktown. A model based on the Quantico MOUT site where the US Marines train for urban combat. It was created by the SAST Development Team at NAWCTSD, Orlando.
- Neighborhood.
- Ocean.
- Park.
- Penn Station. A model of the historical Penn Station, which was located in Philadelphia, Pennsylvania, USA. The model was created by San Jose State University, Fakespace Inc, and MultiGen-Paradigm.
- Railroad crossing.
- Sadr City.
- Sunnyvale. A model of Sunnyvale, California, USA, created by MultiGen-Paradigm using MultiGen Creator.
- E-Town Building Library. Four houses from the 100-house library are included in the DI-Guy Scenario prop library. They were developed by CGSD Corp.

Terrain models and tools for creating them are available from a number of sources. We are happy to help create terrains for you, and can also recommend terrain building specialists.

11.1.1. Loading a Terrain

DI-Guy Scenario loads terrain models in OpenFlight format (*.flt*). These terrain models form the surroundings in which the scenario takes place.

To load your terrain model:

1. Select the Scenario Objects:Scene Object page (Figure 11-2).

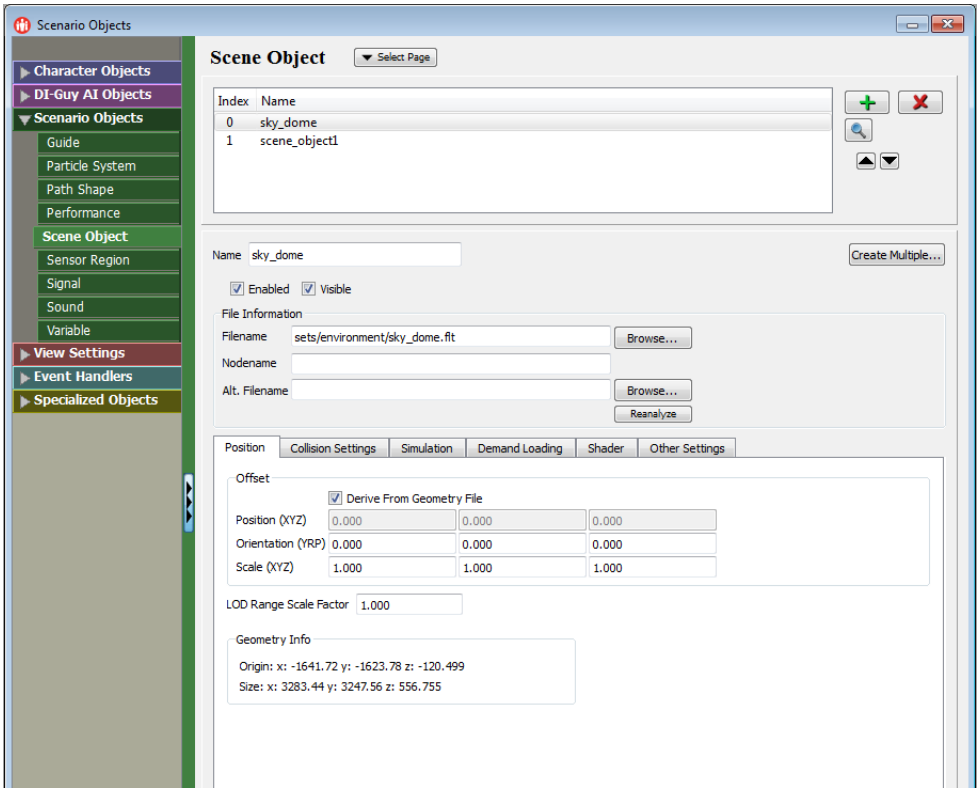


Figure 11-2. Scene Object page

2. In the list, select scene_object1. By default, this is *default_stage11.flt*, as indicated in the Filename box.
3. Click Browse. A file chooser dialog box opens.
4. Select the file you want to load.
5. Click Open.

For your terrain model to load correctly and include all of its textures, you may need to redefine the paths to the texture files. We recommend you use relative paths in the texture file names for your terrain models.

11.2. Positioning the Terrain Database on the Earth's Surface

DI-Guy Scenario uses database coordinates. When it communicates with a simulation using DIS or HLA, the locations of entities are specified to the network in geocentric coordinates. You do not need to deal directly with geocentric coordinates, but you need to know how to situate your database on the surface of the Earth so that when DI-Guy Scenario converts from database coordinates to geocentric, it uses the correct values. DI-Guy Scenario supports UTM, flat earth, and geocentric coordinate systems.

By default, the origin of your database is placed at zero degrees longitude, zero degrees latitude, and at sea-level, according to the WGS84 ellipsoid representation of the earth.

To specify the database location:

1. Choose **Window** → **Network Settings**. The Network Settings dialog box opens.
2. Select the Ex. Location tab (Figure 11-3).

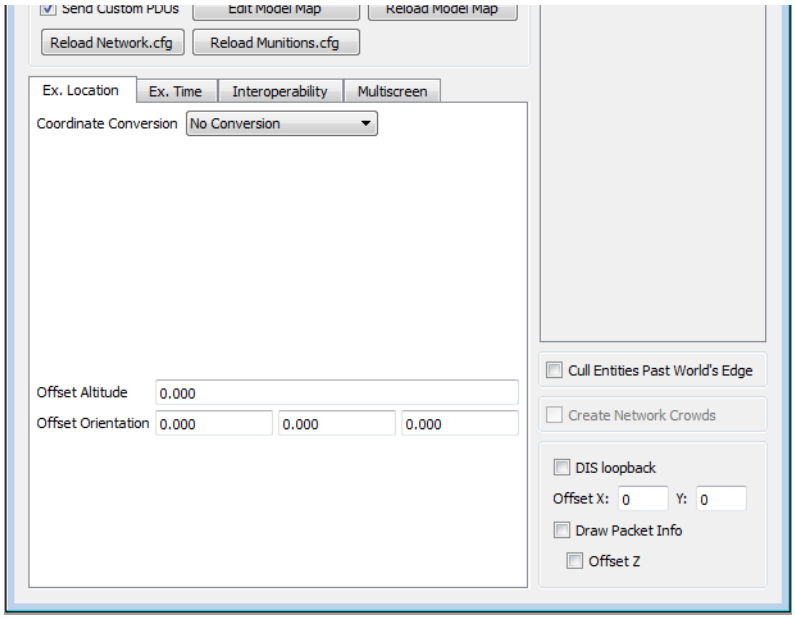


Figure 11-3. Ex. Location tab

3. In the Coordinate Conversion list, select the coordinate system you want to use. The window displays the appropriate coordinate information for the chosen system.

4. For UTM or flat earth, specify the latitude and longitude values for your terrain database. These values can be specified in degrees, minutes, and seconds, or as fractional degrees, and as either signed values or with a N/S, E/W modifier. For example, the latitude 45 degrees, 15 minutes, 45.5 seconds South can be specified in any of the following formats:

   -45 15 45.5

   -45.262639

   45 15 45.5S

   45.262639S

   Longitude is specified similarly, with E and W rather than N and S.

   For more information about using UTM, please see "UTM and the Exercise Location," on page 11-7.

5. For geocentric, specify an offset for the X (east), Y (north), and Z (up) axes, in meters.
6. For any coordinate type, optionally specify an offset altitude, in meters.
7. For any coordinate type, optionally specify an offset orientation, in degrees. The Offset Orientation boxes are in the order positive Z axis, positive X axis, positive Y axis.

11.2.1. UTM and the Exercise Location

For network transmission, your database coordinates are translated into geocentric coordinates. This translation is not linear. Your database, which is taken as flat and rectangular, is curved to match the earth's surface. The origin of the database is mapped to some location within a UTM zone. From this point, database X values correspond to relative easting, database Y values correspond to relative northing, and database Z values correspond to relative altitude.

The value specified in the Longitude box is taken to define the centerline (the 500000 meter line) of a UTM zone, and the first and second Offset Position values are taken as an easting offset and a northing offset from the position identified by the Latitude and Longitude boxes.

UTM zones are numbered from 1 to 60. You can compute the latitude and longitude of the center of a UTM zone n as follows:

```
Longitude = (n * 6) – 183
Latitude = 0
```

Once you have entered these values in the Latitude and Longitude boxes, you must specify the Position Offset of your database origin. The Position Offset tells where your database origin is relative to the center of the UTM zone. The proper Position Offset is a simple function of the UTM coordinates of the database origin:

- X offset = Easting – 500000.
- Y offset = Northing if the database origin is in the northern hemisphere.
- Y offset = Northing – 10000000 if the database origin is in the southern hemisphere.

Enter the values computed above into the first and second Offset Position boxes. Once you have performed these steps the UTM coordinates displayed on screen will be accurate.

11.3. Paging Terrains

DI-Guy Scenario supports very large terrain databases that are composed of tiles arranged in regular rectangular grids. Tiles are loaded and unloaded (paged) as needed to improve graphics performance. This feature has been used to work on terrains that are several hundred kilometers on a side. Typical terrain tiles are five kilometers on a side.

Terrain creation tools such as Terrex's TerraVista can output terrain as a set of OpenFlight terrain tiles appropriate for use as a scene object grid in DI-Guy Scenario.

The Specialized Objects:Scene Object Grid page lets you specify a set of terrain tiles and the parameters needed to control their loading and use within DI-Guy Scenario.

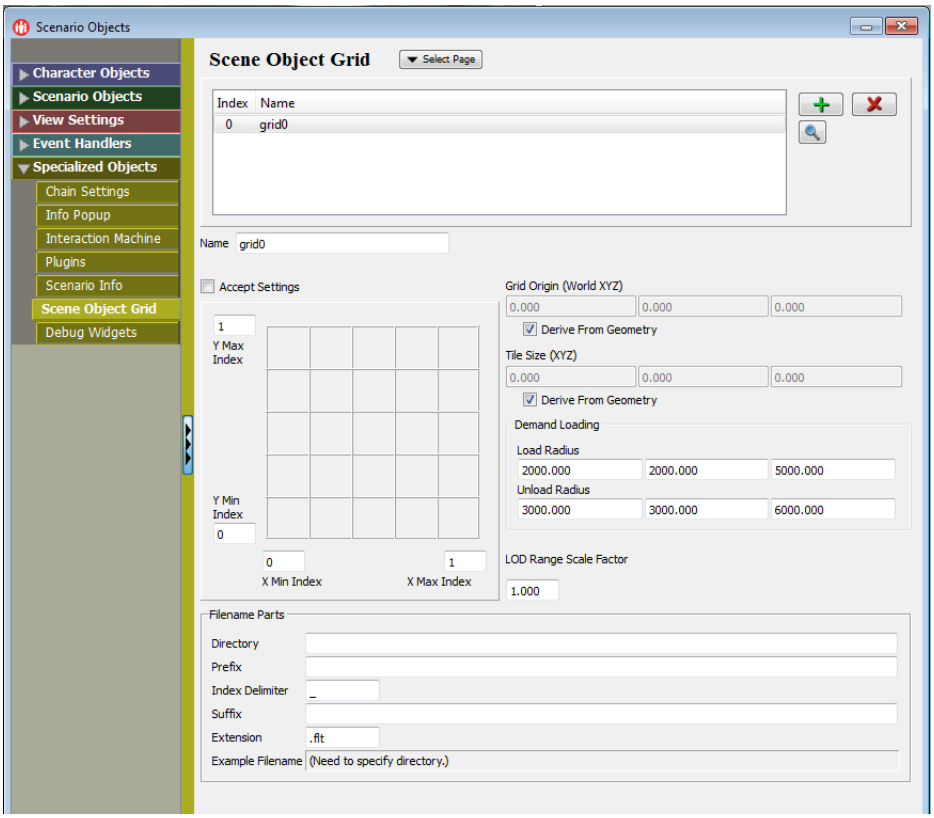


Figure 11-4. Scene Object Grid page

All the tiles that compose a grid must be named in the form <name><xindex><separator><yindex><suffix>.<extension>. For instance the filename for the tile containing the origin of the grid might be *iraq0_0.flt*, where:

- iraq is the name of the terrain.
- 0 is the xindex.
- _ is the index separator.
- 0 is the yindex.
- No suffix is used.
- .flt is the extension.

You can reposition the Scene Object Grid by specifying an XYZ offset in the Grid Origin fields. If Derive From Geometry is selected, then the offset specified in the terrain file is used.

You can also manually specify the size of each tile and the spacing of tiles using the Tile Size fields. If the Derive from Geometry check box is selected, then the size of the tiles is determined automatically from the terrain model.

DI-Guy Scenario pages tiles as needed, focusing on those tiles located near the camera's fixation point. As the fixation point moves through the terrain, tiles load and unload as necessary. Parameters that control the loading and unloading process are Load Radius and Unload Radius. The Load Radius forces tiles to be loaded if they are within the specified distance of the fixation point. The Unload Radius forces tiles to be unloaded if they are further from the fixation point than the specified distance.

DI-Guy Scenario does not try to hide the loading and unloading of tiles, as would be required in an image generator. The purpose here is to allow you to populate a very large space, working in just one region at a time.

11.3.1. LOD Tuning

The LOD Range Scale Factor parameter allows you to adjust the LOD switching ranges to improve performance. If you set the value of this parameter to 0.5, then each LOD switch occurs at half the distance specified in the terrain model. That results in fewer terrain elements being displayed at high resolution and, consequently, better performance.

11.4. Displaying Terrain Coordinates for an Object

You may find it useful to quickly determine the coordinates of a visible object in the terrain database. DI-Guy Scenario can display information about the camera's heading and the cursor's position in a panel that is added to the 3D View (Figure 11-5).

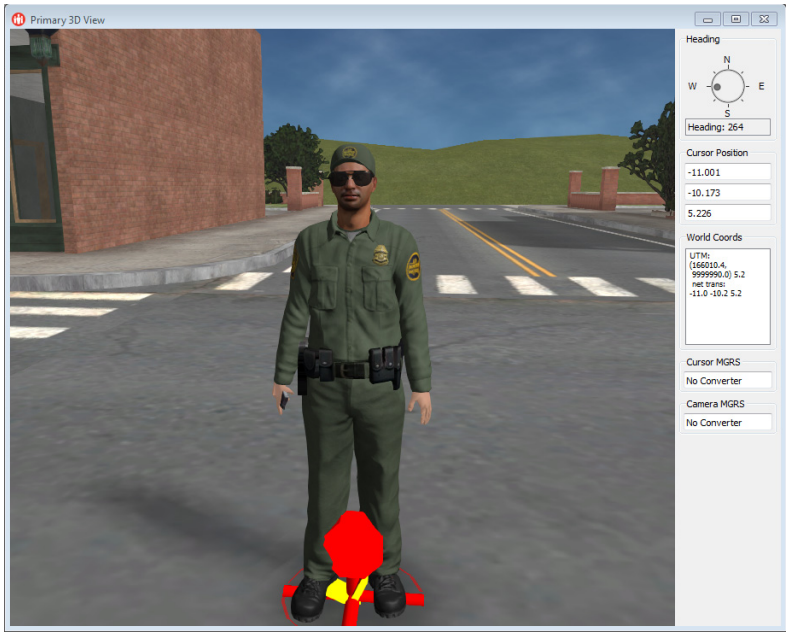


Figure 11-5. Location information in 3D View

The panel displays:

- The camera's heading.
- The cursor position in local coordinates.
- The cursor position in world coordinates.
- The camera position in military grid reference system (MGRS).

You can change the world coordinate format. Formats available are:

- UTM: (599996, 3333342) 0
- LAT/LON: 30:07:38.13N, 46:02:17.12E, 0.120
- Decimal LAT/LON: 30.127259N, 46.038089, 0.120
- Geocentric: 3832700, 3974161, 3182581
- Database: -4.293, 6.404, 0.071

For UTM coordinates to be displayed correctly, the Latitude and Longitude boxes in the Network Settings dialog box, Exercise Location tab (Figure 11-3) must identify the center of the UTM zone.

To turn on display of terrain coordinates:

1. Choose **Edit** → **Preferences**. The Editor Preferences dialog box opens.
2. Select the General Settings tab (Figure 11-6).

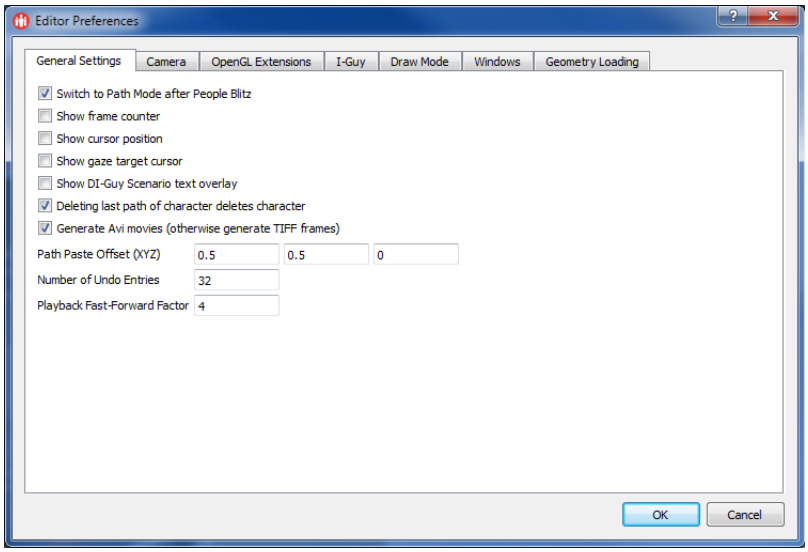


Figure 11-6. Editor Preferences, General Settings

3. Select the Show Cursor Position check box.
4. Click OK. The location information panel is added to the 3D View (Figure 11-5).
5. To change the world coordinates format, press **g**. Keep pressing it to cycle through the supported formats.

11.5. Configuring Fog

You can change the background sky color and add one or more fog effects to a scenario. Each sky and fog combination you create is assigned a name that can be referenced in decision logic.

DI-Guy Scenario supports the OpenGL fog modes Exp, Exp2, and Linear. Exp and Exp2 use different equations to create the fog effect based on the density setting. They do not use the start and end values. You should experiment with these modes to see which look you prefer. Linear mode starts to render fog at the start distance from the camera and increases the density of the fog linearly to the end distance. It does not use the density value.

11.5.1. Adding a Fog Setting

You can add a new setting explicitly, as described in this section, or you can add one by saving current settings to a new setting, as described in "Editing Fog Settings," on page 11-13.

To add a fog setting:

1. Select the View Settings:Fog page (Figure 11-7). The fog settings for the scenario are listed in the window.

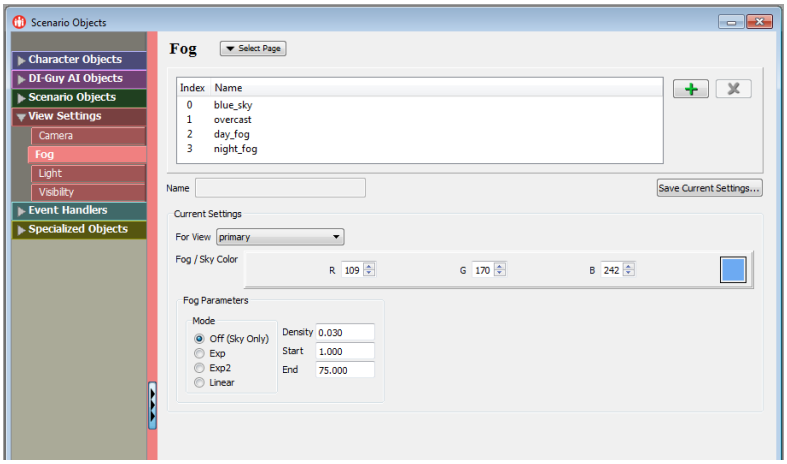


Figure 11-7. Fog tab

2. Click the Add button (+). A new entry is added to the list of fog settings. It has a default name similar to fog0.
3. In the Name box, type a name for the new setting.
4. In the Fog/Sky Color box, enter the RGB values for the sky and fog for this setting or click the color swatch and choose a color from the color picker.
5. In the Fog Parameters group box, specify the mode.
6. For Exp and Exp2 modes, specify the density of the fog.
7. For Linear mode, set the start and end points.
8. Click Save Current Settings. The Select Settings to Overwrite dialog box opens.
9. Select the fog setting to which you want to save the current settings (presumably the one you just added). If you select `(create new)`, a new setting is added to the list.

11.5.2. Editing Fog Settings

To edit fog settings for a scenario:

1. Select the View Settings:Fog page (Figure 11-7). The fog settings for the scenario are listed in the window.
2. Select the fog setting that you want to edit.
3. To change the fog and sky color, change the RGB values, or click the color swatch and choose a new color in the color picker.
4. If you want to save the settings, click Save Current Settings. The Select Settings to Overwrite dialog box opens.
5. Select the fog setting to which you want to save the current settings. You can overwrite an existing setting or apply the current settings to a new setting. If you select `(create new)`, a new setting is added to the list.

11.6. Configuring Lighting Settings

DI-Guy Scenario displays lighting effects in the 3D View. You can create named lighting settings such as daylight, nighttime, and so on. For each setting, you can set the position of the light source, and the ambient, diffuse, and specular components associated with the light source. The lighting components are as follows:

- Ambient. Ambient light is general background light.
- Diffuse. Diffuse light is that which is reflected off of rough surfaces. It is reflected in many directions.
- Specular. Specular light is that which is reflected off of smooth surfaces. It is reflected in one direction.
- Orientation. The direction from which the light source is coming. This affects the direction in which shadows are projected, if they are enabled. If Attached to Camera is selected, the orientation is ignored.
- Attached to Camera. If enabled, the light source moves as the camera moves. The orientation value is ignored.
- Cast Shadows. Enables and disables display of shadows.

11.6.1. Adding a Light Setting

You can add a new light setting explicitly, as described in this section, or you can add one by editing an existing setting and saving current settings to a new setting, as described in "Editing Lighting Settings," on page 11-15

To add a light setting:

1. Select the View Settings:Light page (Figure 11-8). The light settings for the scenario are listed in the window.

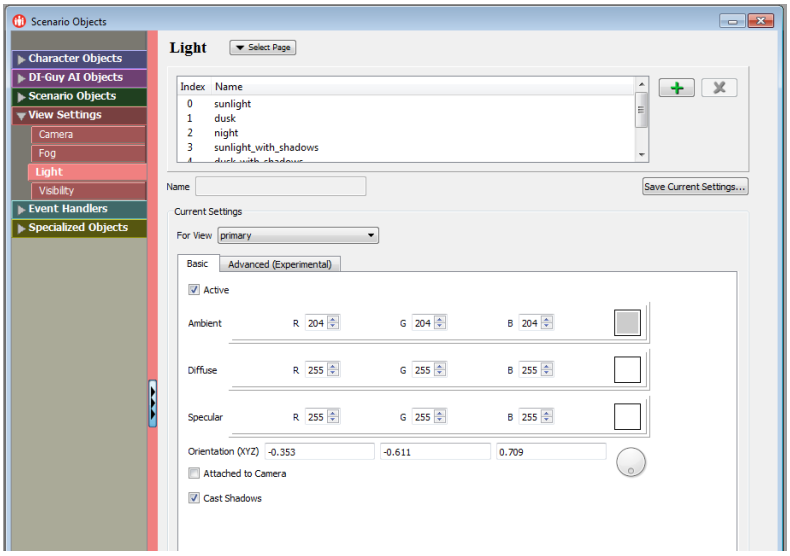


Figure 11-8. Light tab

2. Click the Add button (+). A new item is added to the window with a generic name, such as light0.
3. In the Name box, type a name for the light setting.
4. To enable or disable lighting for this setting, select or clear the Active check box.
5. To change the ambient, diffuse, or specular lighting color, change the RGB values or click the color swatch and choose a new color in the color picker.
6. To set the orientation of the light source, change the Orientation XYZ values, or spin the orientation control to a new orientation. (If Attached to Camera is selected, this control is disabled.)
7. To attach the light source to the camera instead of using the specified orientation, select the Attached To Camera check box. Clear the check box to use the specified orientation.
8. To cast shadows, select the Cast Shadows check box.
9. If you want to save the settings, click Save Current Settings. The Select Light Settings to Use dialog box opens.

10. Select the light setting to which you want to save the current settings – presumably the one you have just added.

11.6.2. Editing Lighting Settings

To edit a lighting setting:

1. Select the View Settings:Light page (Figure 11-8). The light settings for the scenario are listed in the window.
2. Select the light setting that you want to edit.
3. Change the settings as desired.
4. If you want to save the settings, click Save Current Settings. The Select Light Settings to Use dialog box opens.
5. Select the light setting to which you want to save the current settings. You can overwrite an existing setting or apply the current settings to a new setting. If you select `(create new)`, a new setting is added to the list.

12. Networking

This chapter explains how to connect DI-Guy Scenario to an exercise over the network.

Networking in DI-Guy Scenario 12-2
Joining a DIS Exercise 12-2
DI-Guy Custom PDUs 12-4
DI-Guy Scenario Entity Mapping 12-5
Adding a Model Mapping 12-6
Deleting a Model Mapping 12-7
Reloading Model Mappings 12-7
HLA Support 12-7
RPR-FOM Elements and DI-Guy Scenario FOM Extensions 12-8

12.1. Networking in DI-Guy Scenario

Networking is an extra-cost option for DI-Guy Scenario. With the networking option, DI-Guy Scenario can use DIS or HLA to interoperate with other computers running DI-Guy Scenario and with third party applications such as OneSAF, VR-Forces, VBS2, and so on.

Please see Chapter 5, *Networking with DI-Guy*, in *DI-Guy SDK Developers Guide* for technical details of DI-Guy Scenario networking, custom PDUs, FOMS, and so on.

12.2. Joining a DIS Exercise

Distributed Interactive Simulation (DIS) is a network protocol that is commonly used by real time simulations. All participants in an exercise send status updates at a regular interval called a heartbeat. The protocol defines a set of protocol data units (PDUs) for sending updates that all participants understand. Applications may also send custom PDUs. There is no guarantee that another application will understand custom PDUs. DIS does not require use of any mediating application and is relatively simple to use. (By contrast, HLA requires installation of an RTI application.) It is not well suited for exercises that have a high number of entities due to the amount of network traffic that they generate.

To join an exercise using DIS:

1. Choose **Window** → **Network Settings**. The DIS Networking Settings dialog box opens (Figure 12-1).

DI-Guy Scenario uses DIS by default. Please see "HLA Support," on page 12-7 for details about starting DI-Guy Scenario using HLA.

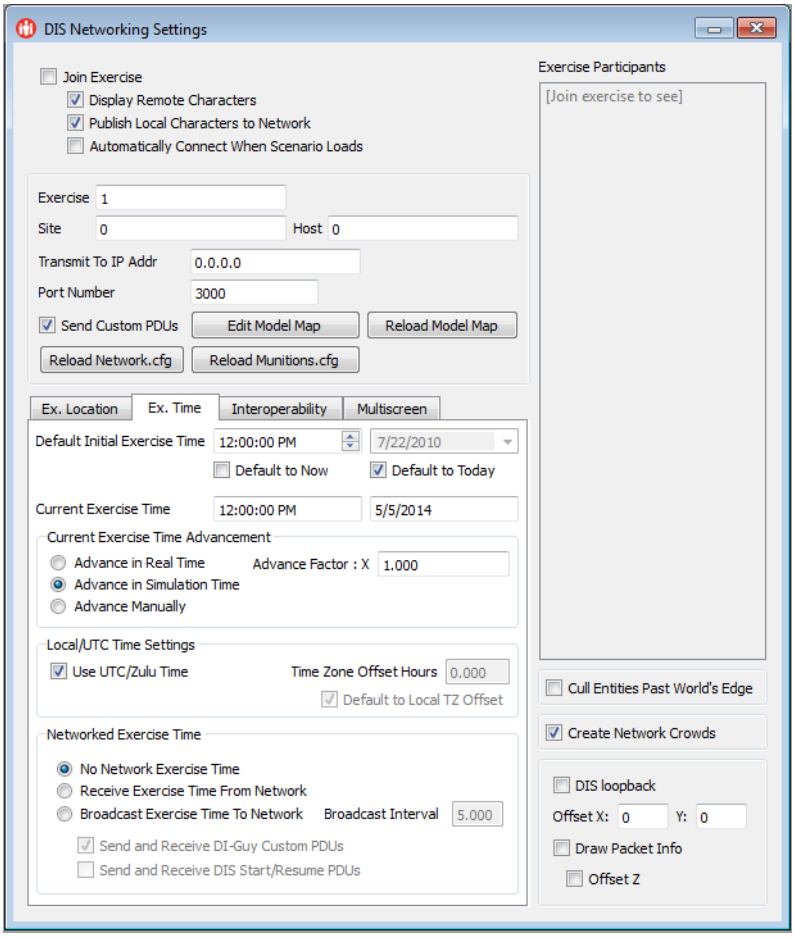


Figure 12-1. DIS Networking Settings dialog box

2. In the Exercise box, type the number of the exercise you want to join. Exercise 0 will connect you to all DIS traffic on the network. Exercise 1 or above will connect you to exercises that use the same exercise number.
3. If there is more than one instance of DI-Guy Scenario connected to the network at your site and you want to interact with them, use the same Port Number. If you want to isolate yourself from other exercises, change the Port Number.
4. Select the Join Exercise check box.
5. If you want DI-Guy Scenario to display remotely simulated entities, select the Display Remote Characters check box.
6. If you want DI-Guy Scenario to broadcast its own simulated entities, select the Publish Local Characters to Network check box.
7. If you want to connect automatically the next time the scenario loads, select the Automatically Connect When Scenario Loads check box.

DI-Guy Scenario saves the connection settings when it exits, and reads those saved connection settings the next time it starts up. Exercise location is stored in the scenario file, and restored when you load the scenario. Entity update rates and other information is stored in *./config/diguy/scenario/network.cfg*. Please see *DI-Guy SDK Developers Guide* for more information.

12.2.1. DI-Guy Custom PDUs

The DI-Guy SDK supports custom PDUs. Applications can use DI-Guy custom PDUs to control DI-Guy Scenario. For example, they can cause DI-Guy Scenario to load a scenario, start, stop, play a script, and so on. Please consult *DI-Guy SDK Developers Guide* for detailed information on how DI-Guy Scenario custom PDUs work and how they can be constructed or read.

12.3. DI-Guy Scenario Entity Mapping

DIS and the HLA RPR FOM use an array of seven integer values, called entity types or entity type enumerations, to describe entities. When DI-Guy Scenario receives information about an entity over the network, it has to decide how to represent the entity in the 3D View. It does this by mapping entity type enumerations to DI-Guy characters.

DIS enumerations are standardized in the DIS Enumeration Document (*Enumeration and Bit Encoded Values for Use with Protocols for DIS Applications*). It is available at the IEEE web site: http://shop.ieee.org/store/ and at the SISO web site: http://www.sisostds.org/.

The DI-Guy Scenario network model map describes how DI-Guy Scenario character types and appearances are mapped to and from DIS entity types. The Network Model Map Editor (Figure 12-2) lets you configure these mappings. The default settings are in *./config/diguy/network_model_map.cfg*. Changes are saved to *./custom/config/diguy/network_model_map.cfg*, which overrides the defaults.

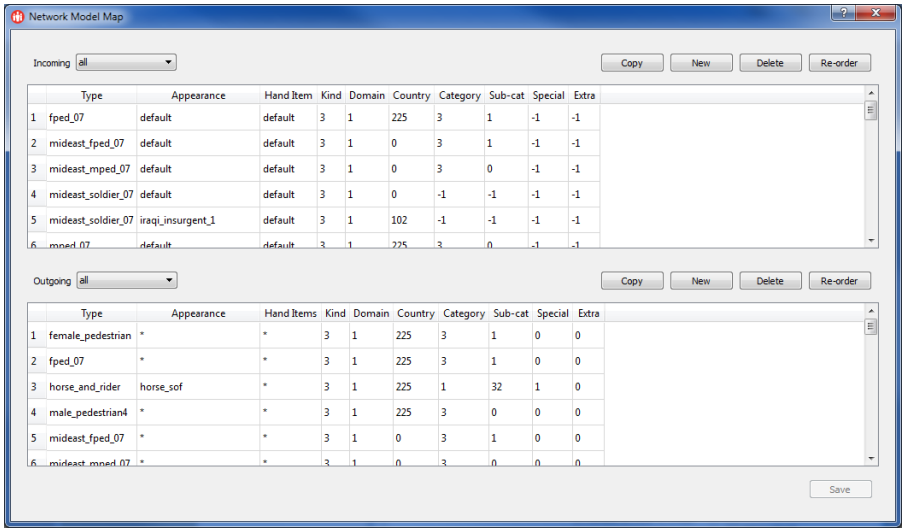


Figure 12-2. Network Model Map Editor

In the map of remotely simulated entity types to DI-Guy Scenario character types and appearances, -1 is a wildcard, meaning any number. In the map of locally simulated entities to entity types, an asterisk (*) is a wildcard meaning any string. Wildcards cannot precede non-wildcards in either mapping. All lines are considered during mapping, and the most specific match (the one with the fewest wildcards) is used. A line with all wildcards to the left of the second equals sign denotes a default mapping for when no more specific mapping exists. If there is no match in the table, DI-Guy Scenario uses an American soldier.

12.3.1. Adding a Model Mapping

You can add new model mappings by adding a completely new mapping or copying an existing mapping. Copying a mapping is quicker if you just want to change part of an existing mapping.

To add a new model mapping:

1. Choose **Window** → **Network Settings**. The DIS Networking Settings dialog box opens (Figure 12-1).
2. Click Edit Model Map. The Network Model Map window opens (Figure 12-2).
3. Optionally, filter the entity type mappings shown in the window by choosing an option from the Incoming or Outgoing list.
4. In the Outgoing section or the Incoming section, click New. A new mapping is added to the end of the list. If you filtered the list, the new mapping uses the character type for which you filtered the list. If you are not filtering the list, the new mapping uses the first character in the list of characters.
5. In the Type column, choose a character type for the new mapping.
6. In the Appearance column, choose an appearance for the character.
7. Optionally, in the Hand Item column, choose a hand item.
8. In the seven columns for the enumeration, type the enumeration values for this character.
9. Click Save.

To add a new model mapping by copying a mapping:

1. Choose **Window** → **Network Settings**. The DIS Networking Settings dialog box opens (Figure 12-1).
2. Click Edit Model Map. The Network Model Map window opens (Figure 12-2).
3. Optionally, filter the entity type mappings shown in the window by choosing an option from the Incoming or Outgoing list.
4. Select the mapping you want to copy. You can click on the Type, Appearance, or Hand Item column.
5. In the Outgoing section or the Incoming section, click Copy. A copy of the mapping is added below the selected mapping.
6. Make your changes to the Type, Appearance, Hand Item, or enumeration values to create the new mapping.
7. Click Save.

12.3.2. Deleting a Model Mapping

To delete a model mapping:

1. Select the model mapping you want to delete.
2. Click Delete.
3. Click Save.

12.3.3. Reloading Model Mappings

If you change model mappings while you are running a scenario, you can reload the model map file to capture the changes.

To reload the model map:

1. Choose **Window** → **Network Settings**. The DIS Networking Settings dialog box opens (Figure 12-1).
2. Click the Reload Model Map button.

12.4. HLA Support

The High Level Architecture (HLA) is simulation networking architecture that was developed to try to resolve some of the problems with DIS, in particular extensibility and the ability to handle high entity counts. Instead of sending complete updates at a specified interval like DIS, HLA sends update messages only when an entity's state changes. Therefore an exercise can have large numbers of entities with infrequent state changes without flooding the network with update messages.

DI-Guy Scenario supports the HLA 1.3 specification, the HLA 1516 SISO DLC specification, and HLA Evolved.

To use HLA, you must install a runtime infrastructure (RTI) application, such as the MÄK RTI.

In HLA, the objects and interactions that an exercise supports are defined in a federation object model (FOM). DI-Guy Scenario supports the RPR FOM, which has a similar set of objects and interactions as DIS. DI-Guy has a set of FOM extensions to support DI-Guy capabilities.

For information about configuring DI-Guy for HLA, please consult Chapter 5, *Networking with DI-Guy*, in *DI-Guy SDK Developers Guide*.

By default, DI-Guy Scenario networking uses DIS. If you want to use HLA, you must start DI-Guy Scenario with the `+module net_hla` command-line argument.

12.4.1. RPR-FOM Elements and DI-Guy Scenario FOM Extensions

DI-Guy provides a FOM for use with HLA 1.3 and HLA 1516 and one for use with HLA Evolved. Please consult the *DI-Guy SDK Developers Guide* for information about attributes and elements used by DI-Guy Scenario, as well as information about the DI-Guy FOM Extensions for HLA.

A. Troubleshooting

This appendix describes some common problems that DI-Guy Scenario users run into.

Troubleshooting FAQ A-2

A.1. Troubleshooting FAQ

After moving a group of waypoints, why are some waypoints not selectable?

When moving a path the X Y Z position of each waypoint is calculated relative to the currently selected waypoint. If the ground is uneven, other waypoints in the group may penetrate the ground when moved, and become hard to see and select. Use the Character Elements:Waypoint page to adjust any waypoints that are not properly set on the ground.

Why are the character's feet sliding along the floor?

The greater the difference between the natural length of a motion cycle and the value in the Length edit box, the more the feet of the character will appear to slide along the ground.

Why did I lose all my action beads?

Changing the character type of a path will delete action beads, as available actions are character specific.

Why can't I create a waypoint or action bead?

The character might not have a path. If there is no current path, you cannot create or edit waypoints or action beads. Make sure there is a current path on the Character Objects:Path page.

Why does my character disappear when it reaches the end of its path?

Check the dead space for the current path on the Character Objects:Path page. If the dead space value is negative, the character will jump back to the beginning of the path to finish motions. If the beginning of the path is off the screen, the character seems to disappear.

Why does my character jump to the next path?

Check the dead space for the current path on the Character Objects:Path page. If the dead space value is positive, the character will jump to the beginning of the next path.

Check to make sure the first waypoint of the next path is in the same position as the last waypoint of the preceding path.

B. Keyboard Shortcuts

This appendix lists keyboard shortcuts and hot keys.

Keyboard Shortcuts and Hot Keys B-2
Hot Keys B-2

B.1. Keyboard Shortcuts and Hot Keys

DI-Guy Scenario has keyboard shortcuts and hot keys that let you quickly access its features and change modes. Two hot keys are particularly useful:

- Pressing **`Alt`** temporarily switches you into Camera Mode, no matter which mode is selected in the Input Mode window. When you release **`Alt`**, the system reverts to the previous mode. This hot key makes it easier to reposition the camera while crafting paths and action sequences.
- When you are in Path Edit, Action Edit, or Sensor Edit modes, pressing **`Ctrl`** shifts you into Path Insert, Action Insert, or Sensor Insert modes, respectively. This allows you to work on positioning a set of waypoints, action beads or sensor regions, then temporarily switch to an insert mode to add a new element, and then revert back to tweaking positions, orientation, and so on.

B.1.1. Hot Keys

Table B-1 lists the hot key functions.

Table B-1: Hot keys

Hot key	Function
Modes	
c	Switch to Camera mode.
f	Switch camera to Forward/Back mode.
s	Switch camera to Left/Right mode.
d	Switch camera to Up/Down mode.
x	Switch to Path Edit mode.
z	Switch to Action Edit mode.
v	Switch PeopleBlitzer mode.
`Alt`	Temporarily switch to Camera mode.
`Ctrl`	Temporarily switch to Insert mode.
`Shift`	Temporarily switch to ‘Go There’ camera mode.
Camera	
f	Switch camera to Forward/Back mode.
s	Switch camera to Left/Right mode.
d	Switch camera to Up/Down mode.
+	Double camera motion speed.
-	Halve camera motion speed.

Table B-1: Hot keys

Hot key	Function
a	Toggle script control of camera.
1 thru 0	Load camera settings.
SHIFT + [0 .. 9]	Save camera to camera settings slot.
Windows and Dialog Boxes	
q	Toggle display of Character Elements page.
w	Toggle display of Scenario Objects page.
e	Toggle display of Camera View page.
r	Toggle display of Signal page.
t	Toggle display of Action Spreadsheet.
l	Toggle display of Log window.
i	Toggle display of Input Mode window.
Miscellaneous	
g	Cycle through units displayed by pointing the cursor.
Ctrl + f	Toggle show frame counter preference.
Ctrl + h	Toggle show overlay text preference.
Ctrl + **SPACE**	Manually push a whole-scenario undo.
>	Next path becomes current path.
<	Previous path becomes current path.
.	Next character becomes current character.
,	Previous character becomes current character.
j	Toggle current character is I-Guy.
p	Show/hide paths.
Standard Hot keys	
Ctrl + x	Cut.
Ctrl + c	Copy.
Ctrl + v	Paste.
Ctrl + z	Undo.
Ctrl + q	Quit.
Ctrl + s	File save.
Ctrl + a	File save as.

Table B-1: Hot keys

Hot key	Function
Ctrl + o	File open.
Ctrl + w	File close.

C. Callback IDs

This appendix lists some of the common callback IDs that you may want to use in scripts.

Callback IDs C-2
Character Callbacks C-2
Scenario Callbacks C-4
Sensor Region Callbacks C-4
Signal Callbacks C-5
Crowd Callbacks (Requires DI-Guy AI) C-5

C.1. Callback IDs

This appendix lists some of the available callback IDs, classified by Event Type. This is not an exhaustive list and you are encouraged to consult the include files such as *diguyCharacter.h*, *diguyCrowd.h*, *diguyCharacterGesture.h*, *diguyScenario.h*, *diguySensorRegion.h*, and so on, for the latest information regarding callbacks.

C.1.1. Character Callbacks

CALLBACK_ID_CREATE. Called when a new character is created.

CALLBACK_ID_CURRENT_ACTION_CHANGED. Called when a character's current action is changed.

CALLBACK_ID_CURRENT_APPEARANCE_CHANGED. Called when a character's current appearance is changed.

CALLBACK_ID_CURRENT_HEAD_APPEARANCE_CHANGED. Called when a character's current head appearance is changed.

CALLBACK_ID_CURRENT_TOUT_REACHED. Called when time equals the character's T-out value.

CALLBACK_ID_DESIRED_ACTION_CHANGED. Called when a character's desired action is changed.

CALLBACK_ID_DESIRED_ACTION_REACHED. Called when a character's desired action is reached.

CALLBACK_ID_DESTROY. Called when a character is destroyed.

CALLBACK_ID_DONE_SPEAKING. Called when the character has finished speaking the contents of a speak() function call.

CALLBACK_ID_END_OF_PATH_REACHED. Called when a character reaches the end of its current path.

CALLBACK_ID_FIRE_WEAPON_SUCCESS. Called when the character has fired his weapon and hit a target. Information is then available on who was hit and where.

CALLBACK_ID_GAZE_STATUS. Called after the character's gaze has experienced a status change.

CALLBACK_ID_MANUALLY_INVOKED. Called when an event handler is manually invoked.

CALLBACK_ID_GUIDE_ORIENTATION_ACQUIRED. Called when the character has reached its desired orientation as set by set_desired_orientation().

CALLBACK_ID_GUIDE_ORIENTATION_UNACQUIRED. Called if the character turns too far away from its desired orientation after it has been previously reached.

CALLBACK_ID_GUIDE_POSITION_ACQUIRED. Called when the character has reached its desired position as set by set_desired_position().

CALLBACK_ID_GUIDE_POSITION_UNACQUIRED. Called if the character moves too far away from its desired position after it has been previously reached.

CALLBACK_ID_HIDE. Called when the character is being hidden for any reason.

CALLBACK_ID_IGUY_INTERACT. Called when the user clicks on another character in I-Guy mode

CALLBACK_ID_IGUY_INTERACT_WITH_SUBJECT. Called when the user clicks on another character in I-Guy mode

CALLBACK_ID_IMPACT. Called after 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.

CALLBACK_ID_LPOINT_STATUS. Called after the character's lpoint (left arm pointing) has experienced a status change.

CALLBACK_ID_POST_CREATE. Called after a character's full creation.

CALLBACK_ID_POST_CREATE_GEOMETRY. Called after a character's geometry is created.

CALLBACK_ID_POST_DIE. Called when the character has been told to die, after a die action has been selected and initiated.

CALLBACK_ID_POST_FIRE_WEAPON. Called when the character has been told to fire its weapon, after a final decision has been made to fire.

CALLBACK_ID_POST_UPDATE. Called after the character is updated.

CALLBACK_ID_PRE_DESTROY. Called before a character's full destruction.

CALLBACK_ID_PRE_DESTROY_GEOMETRY. Called before a character's geometry is destroyed.

CALLBACK_ID_PRE_DIE. Called when the character has been told to die, before a die action has been selected and initiated.

CALLBACK_ID_PRE_FIRE_WEAPON. Called when the character has been told to fire its weapon, before a final decision has been made to fire.

CALLBACK_ID_PRE_UPDATE. Called before the character is updated.

CALLBACK_ID_SHOW. Called when the character is being shown for any reason.

CALLBACK_ID_USER_SELECTED. Called when the character is selected in DI-Guy Scenario.

CALLBACK_ID_USER_UNSELECTED. Called on a currently selected character when a different character is selected in DI-Guy Scenario.

C.1.2. Scenario Callbacks

CALLBACK_ID_CREATE. Called when a new scenario is created.

CALLBACK_ID_DESTROY. Called when a scenario is destroyed.

CALLBACK_ID_LOAD. Called when a scenario is loaded.

CALLBACK_ID_LOAD_SCENARIO_FILE. Called just before a scenario loads a new file.

CALLBACK_ID_RESET. Called when a scenario is reset.

CALLBACK_ID_SAVE. Called when a scenario is saved.

CALLBACK_ID_SAVE_SCENARO_FILE. Called just after a scenario saves a new file, with a char * pointer to the file name.

CALLBACK_ID_SCENE_OBJECT_IMPACT. Called when a weapon was fired and a scene object was struck.

CALLBACK_ID_WAIT_CURSOR_HIDE. Called when a DI-Guy Scenario operation that caused a wait cursor to be shown has completed, allowing an application to hide the wait cursor.

CALLBACK_ID_WAIT_CURSOR_SHOW. Called when a DI-Guy Scenario operation causes a wait cursor to be shown.

CALLBACK_ID_POST_DRAW. Called just after a scenario finishes its draw commands. The *diguyViewPainter* class can be used to issue 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);
```

C.1.3. Sensor Region Callbacks

CALLBACK_ID_CREATE. Called when a new sensor region is created.

CALLBACK_ID_DESTROY. Called when a sensor region is destroyed.

CALLBACK_ID_DETONATION. Called when a detonation occurs within the sensor region.

CALLBACK_ID_AUTOSENSE_CHARACTER_ENTERED. Called when a candidate character has entered a sensor region.

CALLBACK_ID_AUTOSENSE_CHARACTER_LEFT. Called when a candidate character has left a sensor region.

C.1.4. Signal Callbacks

CALLBACK_ID_PRE_TRIGGER. Called whenever the trigger() function is called, whether internally by DI-Guy or explicitly in C++ or in a script. It is called before the default handler of the function.

CALLBACK_ID_POST_TRIGGER. Called whenever the trigger() function is called, whether internally by DI-Guy or explicitly in C++ or in a script. It is called after the default handler of the function.

C.1.5. Crowd Callbacks (Requires DI-Guy AI)

Most of these callbacks have additional data that is stored in the crowd, this data can be retrieved by calling either of the following functions inside the callback handler:

```
diguyCharacter* diguyCrowd::get_callback_character();
diguyImpact* get_callback_impact();
```

CALLBACK_ID_CREATE. Called when a new crowd is created.

CALLBACK_ID_DESTROY. Called when a crowd is destroyed.

CALLBACK_ID_CROWD_MEMBER_KILLED. 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. Called when a member of the crowd is hit by a detonation. If this callback is not present, the impacted character automatically dies. 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. 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. 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.

D. Using the Lua Editor

This appendix describes the features of the Lua editor embedded in the Script pages.

The DI-Guy Scenario Lua Code Editor D-2
- Simple Auto Complete Based on Document Inspection D-2
- Autocomplete for DI-Guy Constants D-2
- Autocomplete for DI-Guy Object Functions D-2
- Calltips D-3
- Full Documentation in the API Viewer D-3
- Querying Lua for Related Objects and Fields D-4

D.1. The DI-Guy Scenario Lua Code Editor

DI-Guy Scenario includes a Lua script code editing system that supports code coloring, syntax highlighting, autocomplete, and automatic display of DI-Guy function calls.

D.1.1. Simple Auto Complete Based on Document Inspection

Whenever you begin to enter code in the Code Editor, it tries to match other words in the code to the current string being entered. If a potential match is found, the Code Editor brings up a multiple choice box showing those autocomplete words.

D.1.2. Autocomplete for DI-Guy Constants

The auto complete index has also been primed with all of the script accessible DIGUY_* constants. Just type DIGUY to show the list. Callback ID's are also included in the list.

D.1.3. Autocomplete for DI-Guy Object Functions

When the member operator ":" is typed in Lua, a simple syntax analyzer is triggered that attempts to derive the object type on the left side of the member operator and then shows what script-accessible functions an object has.

- API Based on Known Objects. If an object name is one that is explicitly set by our script engine, for example: "this_scenario", "this_character", "this_app" or "callback_object", the list of functions that the object has should pop up. The callback_object type is derived from the type of callback that is being edited. In this case there is a high level of confidence in the object type that is being picked.
- API Based on Rules. Lua is a dynamically typed language, making it difficult to determine an exact object type except during code execution. We have implemented a simple rule system for determining what type an object is based on the name of the object.

The list below shows the conventions the Code Editor recognizes. Following the code conventions will greatly assist with debugging and rapid prototyping. Let the Code Editor help you. If you have a *diguyCharacter*, call it "character" or "characterBob", and so on.

Table D-1: Code conventions

Object name starts with:	Assumed to be:
agent or params	diguyAgentParams
camera	diguyViewCamera
camera_settings	diguyViewCameraSettings
character	diguyCharacter
crowd	diguyCrowd
fog	diguyViewFog
gesture	diguyCharacterGesture
group	diguyCharacterGroup
guide	diguyCharacterGuide
impact	diguyImpact
info_popup	diguyInfoPopup
label	diguyViewLabel
light	diguyViewLight
path	diguyCharacterPath
profile	diguyCrowdProfile
sensor_region	diguySensorRegion
region	diguyRegion
signal	diguySignal
sound	diguySound
view	diguyView
variable	diguyVariable
waypoint	diguyWaypoint

D.1.4. Calltips

Whenever you enter "(", the code editor shows calltips for the function in question. The calltip information is in *./doc/diguy_sdk_reference/api*. Press **`Esc`** to hide the calltip.

D.1.5. Full Documentation in the API Viewer

The Code Editor includes a simple DI-Guy API documentation viewer. The documentation is accessible directly using **Help** → **API Docs** or indirectly using the Show Selection's API button.

For example, if you select character:add_guide and click Show Selection's API, the DI-Guy API viewer opens and scrolls to the object/function entry. This will only work with objects that match the DI-Guy Lua naming convention (Table D-1).

D.1.6. Querying Lua for Related Objects and Fields

The Lua language has introspection functionality, allowing you to query what objects and fields a given object might have. If it is not possible to guess an object's type, the Lua interpreter is queried for an object's existence.

For example, if you type "luaCharacter:", a function (located in *luacore.lua*) recursively iterates over the *luaCharacter* object's functions and adds them to a list which is provided to autocomplete. A similar thing happens if you type "luaCharacter.". The autocomplete list includes all fields, including strings, booleans, numbers, tables, and the functions that the object has.

The parser tries to identify nested luaObjects as well, for example, luaSmartObjects.registry.socialize_men

Nested luaObject autocomplete also works on any built-in Lua library. Therefore, if you type "math.", "string.", or "table.", the editor brings up these built-in library contents. Calltips for built-in Lua libraries also get displayed. Local variables cannot be analyzed in this fashion.

E. Switches and DOF in Scene Objects

This appendix explains how to activate switches and degrees of freedom (DOF) in OpenFlight models.

Activating Switches and DOF in Scene Objects E-2
Scene Object Behavior File Keywords E-2
Specifying Switches E-2
Specifying DOF E-3
Example Scene Object Behavior File E-4

E.1. Activating Switches and DOF in Scene Objects

DI-Guy Scenario supports scene object behavior files that operate switch nodes and DOF that are built into your OpenFlight terrain models. This section tells you how to specify a behavior control file for execution, and gives the format for data in the file.

To assign a scene object behavior file to a scenario:

1. Open the scenario file in a text editor.
2. In the Scene Objects section, find the scene object to which you want to add a behavior file.
3. Add the following lines under the scene object:

```
behavior_enabled = 1
behavior_filename = my_behavior_filename
```

E.1.1. Scene Object Behavior File Keywords

Keywords in the scene object behavior file determine the interpretation of subsequent lines of data in the file. At any point in the file, you can use these keywords at the beginning of a line to change the expected data format of subsequent lines. Any line not in a valid format is discarded. The keywords are:

```
switch
dof
dof_flt
```

E.1.2. Specifying Switches

The switch data format is:

```
time switch_name active_state
```

where:

- *time* is a floating point number representing the time in seconds when the data are applied to the scene object. As the program runs, the switch enters the state *active_state* until superseded by another state at a later time.
- *switch_name* is the name of the switch as specified in the OpenFlight (*.flt*) file.
- *active_state* is an integer that specifies which branch of the switch will be active after the given time. For instance, if a switch has two states, say on and off, then the states are 0 and 1, respectively.

The following example specifies a switch named s1 that is turned off at 7 seconds. We first change the data format to switch mode, and then specify a switch change.

```
switch
7.0 s1 0
```

E.1.3. Specifying DOF

The DOF data format is as follows:

```
time dof_name [x:my_x_displacement] [y:my_y_displacement]
   [z:my_z_displacement] [yaw:my_yaw] [roll:my_roll]
   [pitch:my_pitch]
```

where:

- *time* is a floating point number representing the time in seconds when the data are applied to the scene object.
- *dof_name* is the name of the DOF as specified in the OpenFlight (*.flt*) file.
- `x:`, `y:`, and `z:` are optional. The values must be prefaced by the keyword.
- `yaw:`, `roll:`, and `pitch:` are optional. The values must be prefaced by the keyword.

The following example specifies a DOF displaced 3 units in the Y direction and rotated 30 degrees around the yaw-axis (at 7 seconds).

```
dof
7.0 my_dof y:3.0 yaw:30.0
```

Euler Angle Rotation Order

Using the keyword `dof` indicates that the data following will be understood as:

- Yaw represents rotation around the Z-axis.
- Roll represents rotation around the X-axis.
- Pitch represents rotation around the Y-axis.

All rotations of the DOF are done in ZXY order.

Using the keyword `dof_flt` indicates that the data following will be understood as:

- Yaw represents rotation around the Z-axis.
- Roll represents rotation around the Y-axis.
- Pitch represents rotation around the X-axis.

All rotations of the DOF will be done in XYZ order.

Interpolation

The keywords `dof` and `dof_flt` may optionally be followed by a colon and a string denoting the type of interpolation to be done between successive states of the DOF. For instance, you can write `dof:HalfSine` to specify that the DOF state between two successive states should change according to a smooth HalfSine interpolation function. Available interpolation functions are Linear, HalfSine, and QuarterSine.

E.1.4. Example Scene Object Behavior File

The following example code uses the switch nodes and DOF options described in this appendix:

```
dof:HalfSine
0.0 d1
1.0 d2 yaw:30.0 x:2.0
2.0 d2 yaw:30.0 pitch:50.0 y:2.0
switch 1.0 s1 0
2.0 s1 1
2.0 t1 3
3.0 s1 2
4.0 t1 2
dof_flt
5.0 d3 roll:20.0 x:4 y:3 z:2
switch
7.0 s1 0
```

DI-Guy Glossary

This glossary defines terms as they are used in DI-Guy.

action bead

An action bead appears as a yellow crosshair located on a path. Each action bead identifies an action that a person will perform when the person reaches the position of the action bead on the path.

actor

The flesh-and-blood actor whose performance was captured using motion capture. Motion files reference the original actor. The DI-Guy motion engine often performs mathematics to scale motions to fit the end dimensions of the virtual character's appearance.

aim bead

An aim bead sets the azimuth and elevation of the weapon of a person who is aiming.

appearance

The specific geometry and textures that describe the visual look of the character. DI-Guy appearances can be skinned (deformable mesh) or segmented (rigid interpenetrated polyhedral). Many skinned appearances include equipment that may be segmented.

bead

A bead is a graphical symbol marker associated with a person and a path. Beads can be positioned anywhere along a path, but cannot leave the path. DI-Guy Scenario has action beads, aim beads, decision beads, gaze beads, motion beads, and signal beads.

camera

Cameras control the 3D view of the scenario. Each virtual camera has a position and an orientation just as physical cameras do for a motion picture.

chain

A chain is a set of continuous joint-link DOFs that do not branch. Thus, the set of the three DOFs of the left thigh of the character could be considered a single chain, while the set of DOFs for both thighs would have to be considered two chains. Chains are used by Inverse Kinematics

channel

A Motion Track consists of three channels, the A, B, and T channels. The A and B channels can have motions, poses, and trajectories loaded into them while the T channel can have transitions.

character

A human entity, vehicle, or prop in the simulated scenario. Each character has an appearance, a repertoire of behavior, and at least one path.

character_type

The combination of the set of actions a character can perform with the set of visual appearances the character can use.

decision bead

A decision bead is a place on a path where a person's next action is contingent on signals in the scenario or signals initiated by the user.

degree-of-freedom (DOF)

Joints in the model of the character can be described by the number and type of DOFs of which they are composed. For example, a pin joint contains a single degree-of-freedom joint where the joint can rotate about one axis. The base joint is a 6DOF joint; it describes a position that requires 6 degrees-of-freedom to specify it (X, Y, Z, YAW, ROLL, and PITCH)

entity

A person, vehicle, or prop in the simulated scenario.

gaze bead

A gaze bead sets the azimuth and elevation of a person's gaze.

joint

Describes the way two rigid links are joined together. Popular joints include pin, frozen, 6DOF (degrees-of-freedom), ball, 3DOF, and slider.

joint group asset (JGA)

Every Motion Track contains a JGA. The JGA features a picture of the model with a Boolean value for every DOF in the model. These are used to indicate which DOFs are to be affected by the Motion Track

link

A link is a joint plus an offset. Often shapes are associated with links. Links can be traversed and drawn. Sometimes links are referred to as “bones”.

motion

A motion is a set of one or more frames of data. Each frame of data specifies the value for every DOF (Degree of Freedom) in the model. Motions containing two or more frames of data also have a DT value that specifies the amount of time between two frames of consecutive data. A motion has a time duration of (N-1) x DT seconds, where N is the number of frames in the motion. A motion that has only one frame of data is also called a pose.

motion track

A motion track is a horizontal bar in the Motion Editor Timeline consisting of a head and a body into which motions, poses, and transitions can be loaded. Each motion track has a joint group asset associated with it. Tracks can be combined easily in the Motion Editor to create complex motions.

path

People travel along paths to move through scenarios. Paths are spline-shaped functions defined by waypoints.

person

A person is a human entity in the simulated scenario. Each person has an appearance, a repertoire of behavior, and at least one path.

pose

A pose is a motion that has only one frame of data is called a pose. A pose has duration of zero seconds.

project

A project is the highest level of data organization in the Motion Editor. It describes the complete state of the Timeline and thus preserves all the information needed to fully describe the new motion being created. A project file (*.mep* file) is the text file that describes a project.

runtime pose

A runtime pose is a pose that does not have an independent external file representing it. Runtime pose information is stored in memory when a project is loaded or if a new project is being edited. They are stored directly in the Motion Editor Project File (*.mep*) when a project is saved. A runtime pose can be exported to create a pose asset.

script bead

A script bead is a place on a path where execution of a Lua script is triggered.

sensor region

A 3D region that is sensitive to the presence of a person or vehicle. Each time a person enters or leaves a sensor region, an event is generated. Events are available as signals to other entities.

shape

A mesh or polyhedral associated with a link. Unlike the link, which is conceptual in nature, shapes are actual drawables.

shapeset

A collection of one or more shapes.

signal

A signal is a communication event that can influence the behavior of people in a scenario. For example, one team might provide a signal when it has completed a task so another team can begin a second task at the right time. Signals can be triggered by characters in a scenario or manually by a user.

trajectory

An asset that can be constructed on the timeline that holds a series of trajectory atoms.

trajectory atom

One of the repeated components of a trajectory. The six values [X, Y, Z, Yaw, Roll, and Pitch] that completely specify the position and orientation of an object in space are held in each atom. The sequence of atoms determines the path of the end-effector of the link through time.

variable

An object used to communicate with outside processes through the DI-Guy API.

waypoint

Waypoints are graphical elements that define paths. Each waypoint has a position, an orientation, and weighting factors that determine how much the waypoint affects the shape of the path.

Index

-C command-line option 2-6
-D command-line option 2-5
-f command-line option 2-5
-fullscreen command-line option 2-5
-fullscreen2 command-line option 2-5
-hide_menu command-line option 2-5
-hide_status command-line option 2-5
-L command-line option 2-5
-module command-line option 2-6
-N command-line option 2-5
-net command-line option 2-5
-p command-line option 2-5
-plugin command-line option 2-6
-pmu command-line option 2-6
-V command-line option 2-6
+module command-line option 2-6
+plugin command-line option 2-6, 2-7

Numerics

2D symbol 8-9
3D
  window, multiple 2-20
3D View, window 8-2
3D View window 3-3

A

action 3-11, 3-18, 6-11
  adding 3-19, 6-10
  affected by scale 4-13
  bead 5-2
  changing 3-21
  cutting, copying, pasting 4-19
  deleting 6-11
action (continued)
  duration, exact 6-11
  editing 6-10
  non-forward 6-7
  pointing camera at 6-7
  stationary 3-20
  traveling 3-20
action bead 3-18, 6-2
  attributes 6-6
  creating 6-2
  dragging 3-20
  editing 6-5
  loss of A-2
  moving 3-20, 6-6
  propagation of 6-5
  transitions 6-6
Action Edit 3-20, 6-2
Action Insert 3-19, 6-2
Action Spreadsheet 6-2, 6-9
active object 9-17
actor 1-1
adding
  action 6-10
  actions 3-19
  character 3-11
  decision bead 6-12
  fog setting 11-12
  light setting 11-14
  model mapping 12-5, 12-6
  object 2-21
  particle system 7-7
  special effect 7-6
  waypoints 3-16
AI_Crowd_Blitz_Demo, example scenario 4-39

aim
  bead 5-2
  ending 6-19
aim angle mode 6-19
aim bead, creating 6-19
aim character mode 6-19
aim point mode 6-19
aiming
  character 6-19
  using joystick 4-25
altitude, waypoint 5-10
Altitude Update Rate 10-11
ambient light 11-13
AND 6-14, 9-3
angle, aiming 6-19
Animation Load Management 10-11
API, label functions 4-38
appearance 3-10, 1-1
  changing 3-12
  character, changing 4-11
  choosing 3-10
Appearance Wizard 4-6
Apply Actor Scale to Action Bead Travel 4-13
argument, command line 2-4
attaching, camera to character 3-30, 8-4
attribute, action bead 6-6
autocomplete, Lua D-2
AVI 2-18
  recording scenario 2-14

B

bead 5-2
  decision 6-11
behavior 3-18
Blitz Action 4-4
boundary, extent 10-9
branched, path 5-10
button
  joystick, mappings 4-25
  label 4-34, 4-39
  mouse 8-2

C

callback 4-41, 9-2
  character 4-17, C-2
  crowd C-5
  mapping to events 9-11
  scenario C-4
callback (continued)
  sensor region C-4
  signal C-5
callback ID C-2
CALLBACK_ID_AUTOSENSE_CHARACTER
    _ENTERED 7-15
CALLBACK_ID_AUTOSENSE_CHARACTER
    _LEFT 7-15
CALLBACK_ID_USER_SELECTED 4-41
CALLBACK_ID_USER_UNSELECTED 4-41
calltip D-3
Camera
  input mode 8-2
  page 8-6
camera 3-28
  attached to lighting 11-13
  attaching to character 8-4
  control of 2-18
  frustum culling 10-9
  heading 11-9
  jumping 3-30
  jumping to location 8-3
  motion, direction 8-2
  moving 3-5, 8-2
    up down left right forward and backward 8-2
  orientation 8-2
  overriding scripts or tracking settings 8-8
  point of view 8-5
  pointing at action 6-7
  POV 3-32
  projection mode 8-7
  setting
    changing order 8-7
    loading 8-6
    saving and loading 8-5
  settings
    hot key 8-6
    loading from hot key 8-6
    saving 8-6
  speed
    Mouse Move mode 8-2
    projection mode 8-7
  teleporting 3-30, 8-3
    to character 2-22
  tethering to character 3-30
  tracking character 3-31, 8-5
Camera page 3-29, 3-30, 8-2, 8-6, 8-9
camera setting, saving 3-29
Camera tab, Preferences DI-Guy 2-18
Centennial Hall, terrain database 11-3

chain 4-44
Chain Settings, page 4-44
changing 8-7
  action 3-21
  appearance 3-12
  character appearance 4-11
  character head 4-11
  character name 4-9
  character scale 4-12
  character type 4-10
  environment 3-34
  object index 2-22
character
  adding 3-11
  adding to scenario 3-6
  aiming direction 6-19
  appearance, changing 4-11
  attaching camera to 8-4
  callbacks C-2
  chains and ropes 4-44
  choosing 4-6
  creating 4-3
  creating with path 4-5
  culling 10-8, 10-9
  cutting, copying, pasting 4-19
  decision logic 3-22
  deleting 2-18, 4-9
  direction of gaze 6-21
  editing 4-10
  effect 3-12, 7-5
  enabling and disabling 4-9
  entity type enumeration 12-4
  event handlers and callbacks 4-17
  generic procedures 2-20
  group membership 4-14
  head, changing 4-11
  hiding 4-10
  I-Guy 4-22
  label 4-18, 4-34
    enabling and disabling 4-35
  motion texture 4-14
  movement 5-2
  name, changing 4-9
  network settings 4-14
  parent 4-14
  parenting to character 4-21
  paths 5-2
  performance 4-19
  prop 3-12
  review data 4-15
character (continued)
  scale
    affecting action 4-13
    changing 4-12
  scene object 4-13
  selecting 3-13, 4-8
  shader 4-15
  teleporting camera to 2-22
  tethering camera to 3-30
  time in and time out 4-13
  tracking 3-31
  tracking with camera 8-5
  type 1-2
    changing 4-10
  variables 4-20
Character Objects, section 4-3
Character Objects tab 3-13
Character page 3-14
  Events tab 4-17
  General tab 4-14
  Label tab 4-18
  Performance Settings tab 4-19
  Review Data tab 4-15
  Shader tab 4-15
  Simulation tab 4-17
  tabs 4-13
character type
  choosing 3-10
  vehicle 3-12
choosing
  appearance 3-10
  character 4-6
  character type 3-10
class, *diguyViewLabel* 4-38
clickable object 9-17
clipping plane 8-8
codec 2-14
coil 4-44
collision detection 4-24
color, sky 11-11
combining, scenarios 2-14
command line
  argument 2-4
  starting DI-Guy Scenario 2-4
command-line option
  +module 2-6
  +plugin 2-6, 2-7
  -C 2-6
  -D 2-5
  -f 2-5

command-line option (continued)
-fullscreen 2-5
-fullscreen2 2-5
-hide_menu 2-5
-hide_status 2-5
-L 2-5
-module 2-6
-N 2-5
-net 2-5
-p 2-5
-plugin 2-6
-pmu 2-6
-V 2-6
companion, waypoints 6-7
companion waypoint
tutorial 6-8
updating 6-8
compressing, textures 10-13
configuration file
network 2-6
startup 2-6
configuring, fog 11-11
constant D-2
content, customizing 1-7
control, common 2-20
Convention Center, terrain database 11-3
converting, decision bead to script bead 6-17
converting to script bead 6-17
coordinate
database 11-5
flat earth 11-5
geocentric 11-5
mouse 2-18
object 11-9
UTM 11-5
copying
characters 4-19
object into scenario 2-24
objects 2-22
creating
action beads 6-2
aim bead 6-19
character 4-3
character with path 4-5
decision logic 6-13
formation 4-27
gaze bead 6-21
group 4-32
library 2-26
object 2-21
creating (continued)
path 4-5, 5-2
paths 3-14
scenario 1-4, 2-10
sensor region 7-13
signal 7-17
variable, scenario 7-21
waypoint 5-6
waypoints 5-5
in 3D View 5-6
crowd, callbacks C-5
culling
camera frustum 10-9
distance 10-10
objects 10-8
current time 2-11
cursor, position 2-18, 11-9
curvature, path 3-15
custom
content 1-7
PDUs 12-4
cutting, characters and paths 4-19

D

database, coordinates 11-5
dead space 5-4, A-2
dead-reckoning 7-3
debris 7-5
debugging
rendering 10-12
scenario 2-19
Decision, page 9-4
decision
bead 5-2
converting to script 9-6
decision bead 3-22, 4-21, 6-11, 6-17
adding 6-12
logic for 6-13
Decision Bead page 3-23, 6-12
Decision Editor 3-25, 6-14, 9-3
decision logic 3-21, 9-2
character 3-22
creating 6-13
mapping 9-11
scenario 9-3
Decision page 9-6
deleting
action 6-11
character 2-18, 4-9

deleting (continued)
model mapping 12-6
objects 2-22
Demand Loading, tab 10-7
demand loading, terrain 10-6
demo, mode 2-5
Derive Desired Duration 6-6
Derive Duration 6-6
Derive Length 6-6
Derive Reps 6-6
description, scenario 2-13
Desert, terrain database 11-3
dialog box
Editor Preferences 11-10
Network Settings 11-5
Object Library Selector 2-27
Signals 7-18
diffuse light 11-13
DI-Guy, installing 2-3
DI-Guy AI 1-4, 1-5, 8-9, C-5
DI-Guy API 9-7
DI-Guy FOM Extensions 12-7
DI-Guy Motion Editor 1-5, 1-7
DI-Guy Scenario
starting 2-3, 3-3
command line 2-4
DI-Guy SDK 9-2, 9-7, 9-9
label functions 4-38
diguyViewLabel class 4-38
Direct X 1-4
DIS 1-5
Enumeration Document 12-4
HLA 11-5, 12-2
joining exercise 12-2
disabling
characters 4-9
label, character 4-35
displaying
frame rate 2-17
object text 2-23
objects 2-28
distance, culling 10-10
DLL 9-9
DOF, OpenFlight E-2, E-3
dragging, action beads 3-20
Draw Mode tab, Preferences dialog box 2-19
draw style, path 5-4
DSS file extension 2-3
dual screen, mode 2-4
duration
action, exact 6-11
dust 7-5

E

editing
action 6-10
action bead 6-5
action beads 6-5
characters 4-10
fog setting 11-13
label 4-36
lighting settings 11-15
paths 3-15
sensor region 7-15
waypoints 5-8
editor, Lua D-2
Editor Preferences dialog box 11-10
General Settings tab 11-10
effect
character 3-12, 7-5
special 7-5
effects, lighting 11-13
Else 6-14, 9-3
embarkation, character 4-21
enabling
characters 4-9
label, character 4-35
ending
aiming behavior 6-19
gazing 6-21
entity type 12-4
enumeration 12-4
enumeration, entity type 12-4
Enumeration and Bit Encoded Values for Use with Protocols for DIS Applications 12-4
environment, changing 3-34
E-Town Building Library, terrain database 11-3
euler angle E-3
event 9-2
mapping to callback 9-11
timing 9-10
trigger 9-10
types 9-3
event handler 4-42, 9-2
character 4-17
Event Mapper 9-11
page 9-11
Event Mapper page 4-42, 9-18

Event Mapping Editor 9-12
Events tab, Character page 4-17
Ex. Location tab 11-5
exact duration 6-11
exercise
  network, joining 12-2
exercise location 11-6
Exercise Time, enabling or disabling 2-19
exface, module 2-6
explosion 7-5
Expressive Faces 1-5, 1-7, 4-33
extending, paths 3-16
extent, boundary 10-9
eyepoint. See camera

F

Face Expressions page 4-33
FaceFX 1-7
FAQ A-2
far clip 8-8
fast forward 2-11
  speed 2-18
fast reverse 2-11
federation object model 12-7
feet
  sliding A-2
  slipping 6-11
field of view 8-8
file
  extension, DSS 2-3
fill mode, polygon 10-12
fire 7-5
firing, using joystick 4-25
first person POV 4-25, 8-5
fixation point, gaze 2-18
flat earth, coordinates 11-5
floating, window 2-19
fog
  adding setting 11-12
  configuring 11-11
  setting, editing 11-13
fog mode, OpenGL 11-11
Fog page 3-35, 11-12, 11-13
FOM 12-7
format, time 2-19
formation
  creating 4-27
  role 4-31
  subgroup 4-30
FOV 8-8
frame, stepping through scenario 2-11
frame rate, displaying 2-17
free camera, projection mode 8-7
friendly, group 4-32
frustum 8-8
  camera, culling 10-9
fullscreen, mode 2-5
function
  library 9-9
  SDK 9-2

G

gain, joystick 4-26
gaze
  bead 5-2
  character 6-21
  ending 6-21
  fixation point 2-18
gaze angle mode 6-21
gaze bead, creating 6-21
gaze character mode 6-21
gaze point mode 6-21
General Settings tab, Editor Preferences dialog box 11-10
General tab
  Character page 4-14
  Preferences dialog box 2-17
generic gestures 6-23
geocentric, coordinates 11-5
geometry
  loading 2-19, 10-13
  optimizing 2-19, 10-13
Geometry Loading tab, Preferences dialog box 2-19
gesture 6-23
  generic 6-23
  head 6-23
  previewing 6-24
Gesture Previewer 6-24
graphics, LOD 10-6
grid, terrain 11-7
ground clamping, waypoints 5-10
group
  creating 4-32
  membership 4-14
GUI, common procedures 2-20
guide 7-3
Guide page 7-4

H

hand item 3-11
head
  character, changing 4-11
  nodding and shaking 6-23
heading, camera 11-9
hiding
  characters 4-9, 4-10
  menu bar 2-5
  objects 2-28
  paths 5-4
  status bar 2-5
High Level Architecture, HLA 12-7
HLA 1-5
  RPR FOM 12-4
HLA 1.3 12-7
HLA 1516 12-7
HLA Evolved 12-7
hostile, group 4-32
hot key B-2
  camera setting 8-7
  camera settings 8-6
  loading camera setting 8-6
hot object 9-17

I

ID, callback C-2
If 6-14, 9-3
I-Guy
  character 4-22
  keyboard and mouse control 4-26
  mode 4-22
I-Guy Settings page 4-24
I-Guy tab, Preferences dialog box 2-18
importing, library into scenario 2-27
index
  object, changing 2-22
  path 5-2
Info Popup, page 9-13
info popup, opening from script 9-14
information
  logging 2-16
  popup window 9-13
Initial Motion Texture 4-14
initial orientation 4-12
initial path 4-11
initial play mode 2-5
initial position 4-12
initialization, script 2-7, 2-8
Input Mode, window 8-2
input mode, Camera 8-2
Input Mode window 3-3, 4-3
inserting
  waypoint 5-6
  waypoints 3-17
installing, DI-Guy Scenario 2-3
interaction machine 9-14, 9-15
Interaction Machines page 9-15
Is Scene Object 4-13

J

Jailhouse, terrain database 11-3
joining, exercise 12-2
joint 1-3
joystick 4-22
  aiming and firing
    first person POV 4-25
    third person POV 4-25
  button, mapping 4-25
  gain 4-26
  settings 2-19
  using 4-23
jumping, camera 3-30

K

keyboard 4-22
  I-Guy mode 4-26
  shortcuts B-2
keyframe 7-9, 7-10
keyword, switch E-2
Khost Afghanistan, terrain database 11-3

L

label
  button 4-34, 4-39
  character 4-18, 4-34
    enabling and disabling 4-35
  editing 4-36
  function, SDK 4-38
  property sheet 4-37
  screen 4-34, 4-39
Label tab, Character page 4-18
level of detail, tuning 10-4

library
adding objects to 2-26
creating 2-26
function 9-9
importing into scenario 2-27
managing 2-26
object 2-23
Library Function, page 9-9
light
setting, adding 11-14
Light page 3-34, 11-14, 11-15
lighting
effects 11-13
fog 3-34
setting, editing 11-15
link 1-3
Linux 1-4
load, radius 10-6
load management 10-11
Load Manager 4-19, 10-10
Load Radius 10-7
loading
camera setting 8-6
camera settings 8-5
geometry 10-13
module, startup 2-6
plug-in 2-6
plug-ins 2-7
scenario 2-10
terrain 3-32
terrains 11-3
local path 5-4
location, terrain 11-5
LOD
graphics 10-6
tuning 10-4, 11-9
tuning for terrain 10-5
LOD Range Scale Factor 10-5, 11-9
LOD tuning, terrain 11-9
Log, window 2-16
log, sending to file 2-5
log file, notification level 2-5
Log window, notification level 2-5
logic
decision 6-12
creating 6-13
Look At 8-5
overriding 8-8
Look From 3-30
overriding 8-8
looking
character, direction 6-21
Lua 6-17, 6-18, 9-2, 9-6, 9-7
autocomplete D-2
editor D-2

M

machine
interaction 9-14
state 9-14
managing, library 2-26
material, enabling and disabling 10-12
Max Ticks Behind Until Skip 10-4
memory management 2-19, 10-13
menu bar, hiding 2-5
merging, scenarios 2-14
Microsoft Sidewinder Precision 2 4-23
Middle East Town, terrain database 11-3
mode
demo 2-5
dual screen 2-4
fullscreen 2-5
I-Guy 4-22
Mouse Move 8-2
Path Edit 3-6
projection 8-7
model, terrain 3-4, 11-2
model mapping
adding 12-5, 12-6
deleting 12-6
reloading 12-6
module
exface 2-6
loading and unloading at startup 2-6
net_dis 2-6
net_hla 2-6
particles 2-6
script_lua 2-6
sound_openal 2-6
Motex Gain 4-14
motion, camera direction 8-2
motion texture, character 4-14
mouse 4-22
button 8-2
coordinates 2-18
I-Guy mode 4-26
Mouse Move mode 8-2
camera speed 8-2

movie
AVI or TIFF 2-18
recording scenario to 2-14
moving
action bead 3-20, 6-6
camera 3-5, 8-2
waypoints 3-15, 5-9
multiple 3D windows 2-20

N

name
character, changing 4-9
NAWC Rocktown, terrain database 11-3
near clip 8-8
Neighborhood, terrain database 11-3
net_dis, module 2-6
net_hla, module 2-6
network
configuration file 2-6
connecting to at startup 2-5
Network Model Map 12-5
Network Model Map Editor 12-5
network settings, character 4-14
Network Settings dialog box 11-5
network updates 7-3
networking 12-2
DIS and HLA 1-5
neutral, group 4-32
nodding, head 6-23
non-forward action 6-7
normals, visualizing 10-12
notification level 2-16
log file 2-5
Log window 2-5

O

object
adding 2-21
coordinate 11-9
copying 2-22
copying into scenario 2-24
culling 10-8
deleting 2-22
displaying 2-28
generic procedures 2-20
hiding 2-28
hot 9-17
index, changing 2-22
object (continued)
libraries 2-23
sub shape 10-9
text, displaying 2-23
Object Library Manager 2-26
Object Library Selector dialog box 2-27
Ocean, terrain database 11-3
OpenFlight
DOF E-3
DOF and switches E-2
OpenGL 1-4
fog mode 11-11
opening, scenario 2-10
OpenSceneGraph 1-4
optimizing
geometry 2-19
geometry files 10-13
option, command-line 2-4
OR 6-14, 9-3
orientation
camera 8-2
lighting 11-13
Ortho XY projection mode 8-7
orthographic projection mode 8-7
overlay, text 2-18

P

page
Camera 3-29, 3-30, 8-2, 8-6, 8-9
Chain Settings 4-44
Character 3-14
Decision 9-4, 9-6
Decision Bead 3-23, 6-12
Event Mapper 4-42, 9-11, 9-18
Face Expressions 4-33
Fog 3-35, 11-12, 11-13
Guide 7-4
I-Guy Settings 4-24
Info Popup 9-13
Interaction Machines 9-15
Library Function 9-9
Light 3-34, 11-14, 11-15
Path 5-2
Scenario Info 2-13
Scene Object 3-33, 10-7, 11-4
Scene Object Grid 11-7
Script 2-8, 2-9, 9-6, 9-7
Scripts 9-17
Sensor Region 7-15

page (continued)
Signal 7-17
Variable 7-21
Visibility 7-6
Waypoint 5-6
Weapon Parameters 4-21
paging, terrain 11-7
parameter
particle system 7-8
weapon 4-21
parent, character 4-14
parenting, character to character 4-21
Park, terrain database 11-3
particle system 3-12, 7-5
adding 7-7
box, showing and hiding 7-6
parameters 7-8
particles, module 2-6
pasting, characters 4-19
path
branched 5-10
character 5-2
creating 3-14, 4-5, 5-2
curvature 3-15
cutting, copying, and pasting 4-19
draw style 5-4
editing 3-15
extending 3-16
hiding 5-4
index 5-2
regular and local 5-4
selecting 3-14, 5-4
shape 7-11
waypoint 5-2
waypoints 5-5
Path Edit mode 3-6
Path Insert 5-3
Path page 5-2
path shape 5-2
PDU, custom 12-4
Penn Station, terrain database 11-3
people, adding to scenario 3-6
PeopleBlitzer 3-7, 4-3
performance
character 4-19
culling 10-8
demand loading 10-6
Load Manager 10-10
LOD tuning 10-4
rendering quality 2-19
performance (continued)
tick rate 10-3
tuning 10-2
Performance Settings tab, Character page 4-19
Planview XY projection mode 8-7
Planview XZ projection mode 8-7
Planview YZ projection mode 8-7
platforms, supported 1-4
play mode, initial 2-5
playback, starting and stopping 2-11
playing, scenario 2-11, 3-8
plug-in
loading 2-7
loading and unloading at startup 2-6
plug-in architecture 1-5
point of view, camera 3-32, 8-5
polygon, fill mode 10-12
polygon fill mode 2-19
popup window, information 9-13
Pose Update Rate 10-11
position, cursor 2-18, 11-9
Position Update Rate 10-11
POV 3-32
camera 8-5
preferences, setting 2-17
Preferences dialog box
Draw Mode tab 2-19
General tab 2-17
Geometry Loading tab 2-19
I-Guy tab 2-18
Windows tab 2-19
Preferences DI-Guy, Camera tab 2-18
previewing, gesture 6-24
Primary 3D View window 3-3
procedure, common 2-20
projection, mode 8-7
projection mode, camera speed 8-7
prop, character 3-12, 4-44
property sheet, label 4-37
propogating, action beads 6-5
protocol data unit, PDU 12-2

Q

quality
rendering 10-12
visual 10-12

R

radius
load 10-6
unload 10-6
Railroad crossing, terrain database 11-3
Ramp Editor 4-16
read-only, scenario 2-13
recording, scenario 2-14
region
sensor 3-21, 7-12
callback C-4
editing 7-15
regular path 5-4
reloading, model mapping 12-6
rendering
debugging 10-12
quality 10-12
rendering quality, performance 2-19
reps attribute 6-6
resetting, scenario 2-11
resetting number of 7-18
resolution, managing for performance 10-4
reverse, playback 2-11
review data, character 4-15
Review Data tab, Character page 4-15
rewinding, scenario 2-11
role, formation 4-31
rope 4-44
RPR FOM 12-7
extensions 12-7
HLA 12-4
RTI 12-2, 12-7
running, scenario 2-11
runtime infrastructure 12-7

S

Sadr City, terrain database 11-3
saving
camera setting 3-29
camera settings 8-5, 8-6
scenario 2-10, 3-9
scale
affecting action 4-13
character, changing 4-12
scenario
adding people 3-6
callbacks C-4
copying object into 2-24
scenario (continued)
creating 1-4, 2-10
debugging 2-19
decision logic 9-3
description 2-13
end time 2-11
file, startup 2-6
importing library into 2-27
introduction 2-9
merging 2-14
opening 2-10
playing 3-8
read-only 2-13
recording 2-14
resetting 2-11
rewinding 2-11
running 2-11
saving 2-10, 3-9
script 9-7
sound effects 7-19
start time 2-11
variable 7-21
Scenario Can Skip Ticks 10-4
Scenario Info, page 2-13
Scenario Objects window 3-13
scene object
character 4-13
culling 10-9
Scene Object check box 10-9
Scene Object Grid page 11-7
Scene Object page 3-33, 10-7, 11-4
Scene Object Sub Shape 10-9
screen, label 4-34, 4-39
Script, page 2-8, 2-9, 9-7
script 9-2
bead 5-2
converting decision to 9-6
initialization 2-7, 2-8
mapping 9-11
opening info popup 9-14
overriding 8-8
scenario 9-7
startup 2-8
script bead
creating from decision bead 6-17
writing 6-18
Script page 9-6
script_lua, module 2-6
Scripts page 9-17

SDK 9-2
  functions 9-2
  label functions 4-38
section, Character Objects 4-3
selecting
  character 3-13, 4-8
  path 5-4
  paths 3-14
  waypoints 5-7
Sensor Insert 3-21, 7-13
sensor region 3-21, 7-12
  callbacks C-4
  creating 7-13
  editing 7-15
Sensor Region page 7-15
setting
  camera
    changing order 8-7
    hot key 8-6
    loading 8-6
    saving 8-6
    saving and loading 8-5
  joystick 2-19
  preferences 2-17
shader, character 4-15
Shader tab, Character page 4-15
shadows, casting 11-13
shaking, head 6-23
shape 1-4
  path 7-11
shapeset 1-4
shortcut, keyboard B-2
shrinking, action 6-11
signal 7-18
  callbacks C-5
  creating 7-17
  sound 7-17
  tracking use of 7-18
Signal page 7-17
Signals dialog box 7-18
Simulation tab, Character page 4-17
skipping, ticks 10-4
sky, color 11-11
sliding, feet A-2
slipping, feet 6-11
slope adjuster 3-15, 5-5
smoke 7-5
smoothing 7-3
smoothness, waypoint 5-9
sound
  adding to scenario 7-19
  signal 7-17
  volume 7-20
sound_openal, module 2-6
special effect, adding 7-6
specify effect 7-5
specular light 11-13
speed
  camera, projection mode 8-7
  camera movement, Mouse Move mode 8-2
  fast forward 2-18
Stage
  terrain 3-4
  terrain database 11-2
starting, DI-Guy Scenario 2-3, 3-3
startup
  configuration file 2-6
  connecting to network 2-5
  loading and unloading module 2-6
  loading and unloading plug-ins 2-6
  play mode 2-5
  scenario files 2-6
  script 2-8
state, machine 9-14
stationary action 3-20
status bar, hiding 2-5
stopping, playback 2-11
stretching, action 6-11
stretching and shrinking 6-11
sub shape, object 10-9
subgroup, formation 4-30
Sunnyvale, terrain database 11-3
supported platforms 1-4
switch, keyword E-2
switch node, OpenFlight E-2
symbol, 2D 8-9
symbolic view 8-9

T

T Out, scenario 2-11
tab
  Character Objects 3-13
  Character page 4-13
  Demand Loading 10-7
  Ex. Location 11-5
teleporting
  camera 3-30, 8-3
    to object 2-22

terrain 3-4
  demand loading 10-6
  grid 11-7
  loading 3-32, 11-3
  location 11-5
  LOD tuning 10-5, 11-9
  models 11-2
  paging 11-7
  positioning on earth 11-5
  scene object 4-13
  Stage 3-4
  tile 11-7
terrain database
  Centennial Hall 11-3
  Convention Center 11-3
  Desert 11-3
  E-Town Building Library 11-3
  Jailhouse 11-3
  Khost Afghanistan 11-3
  Middle East Town 11-3
  NAWC Rocktown 11-3
  Neighborhood 11-3
  Ocean 11-3
  Park 11-3
  Penn Station 11-3
  Railroad crossing 11-3
  Sadr City 11-3
  Stage 11-2
  Sunnyvale 11-3
text
  object, displaying 2-23
  overlay 2-18
texture
  compressing 10-13
  enabling and disabling 2-19, 10-12
Then 6-14, 9-3
third person POV 4-25
tick
  rate 10-3
  setting time between 10-4
  skipping 10-4
Tick_dt 10-3
Ticks Between Checks 10-4
TIFF frame generator 2-18
tile, terrain 11-7
time
  between ticks 10-4
  format 2-19
Time Control, window 2-11
Time Control window 3-3, 3-8
time in, character 4-13
time out, character 4-13
tracking, character with camera 3-31, 8-5
Transition After 6-6
Transition Before 6-6
traveling action 3-20
trigger, event 9-10
Trigger dt 9-10
Trigger T 9-10
Trigger T Out 9-10
troubleshooting A-2
tuning
  LOD 10-4, 11-9
  performance 10-2
tutorial, companion waypoint 6-8
type
  character 1-2
    changing 4-10

U

undo 4-20
  levels of 2-18
unload, radius 10-6
Unload Radius 10-7
unloading
  module, startup 2-6
  plug-in 2-6
updating, companion waypoint 6-8
using, joystick 4-23
UTM
  coordinates 11-5
  exercise location 11-6

V

variable
  character 4-20
  scenario 7-21
    creating 7-21
Variable page 7-21
Vega Prime 1-4
vehicle, character type 3-12
view, symbolic 8-9
view frustum 8-8
Visibility page 2-28, 7-6
visual, quality 10-12
visual quality 7-3
volume, sound 7-20

W

waypoint
adding 3-16
altitude 5-10
companion 6-7
creating 5-5, 5-6
in 3D View 5-6
cutting, copying, pasting 4-19
editing 5-8
ground clamping 5-10
inserting 3-17, 5-6
moving 3-15, 5-9
path 5-2
selecting 5-7
problems A-2
smoothness 5-9
Waypoint page 5-6
weapon, parameters 4-21
Weapon Parameters page 4-21
WGS84 11-5
window
3D, multiple 2-20
3D View 3-3, 8-2
floating 2-19
information popup 9-13
Input Mode 3-3, 4-3, 8-2
Log 2-16
Primary 3D View 3-3
Scenario Objects 3-13
Time Control 2-11, 3-3, 3-8
Windows 1-4
Windows tab, Preferences dialog box 2-19
writing, script bead 6-18

X-Y-Z

Zone Far Extent 10-11

Link - Simulate - Visualize

DIG-13.0-6-140523

150 CAMBRIDGE PARK DRIVE, 3RD FLOOR CAMBRIDGE, MA 02140 617.876.8085 www.mak.com