DI-Guy

HUMAN SIMULATION SOFTWARE

DI-Guy Scenario 12.5

USER GUIDE

DI-Guy Scenario 12.5
User Guide

Boston Dynamics
78 Fourth Avenue
Waltham, MA 02451 USA

For questions regarding *DI-Guy Scenario*,
send email to diguy@BostonDynamics.com .

Part No. DI-Guy Scenario 12.5.0-User Guide

DI-Guy Software License Agreement

This is a legally binding agreement between you ("LICENSEE") and Boston Dynamics, with its principal place of business at 78 Fourth Avenue Waltham, MA USA ("Boston Dynamics"). By breaking the seal on the package containing the DI-Guy products (DI-Guy SDK, DI-Guy Scenario, DI-Guy AI and DI-Guy Motion Editor), and/or by installing and/or using the enclosed software, data, models, documentation, or related recorded information, regardless of its form or the method of the recording (the "SOFTWARE"), LICENSEE is agreeing to be bound by the terms and conditions of this agreement, including the software license and disclaimer of software warranty below. Please read this document carefully before opening the package and using the SOFTWARE. If LICENSEE does not agree with the terms and conditions of this agreement, LICENSEE should promptly return the unopened package and the SOFTWARE to the place where it was obtained, and LICENSEE will be given a full refund of any license fee that LICENSEE paid.

1. Grant of License: Subject to the terms and conditions of this Agreement, Boston Dynamics hereby grants to LICENSEE a non-transferable and non-exclusive right to use and execute the SOFTWARE on a single computer system at one time. LICENSEE agrees that it will not reverse engineer, decompile or disassemble any portion of the SOFTWARE, nor extract data of any kind from the SOFTWARE, including but not limited to motion data, 3D models, textures, texture movies, image sequences, terrain databases and the like. If LICENSEE disposes of any media or apparatus containing SOFTWARE, LICENSEE will ensure that it has completely erased or otherwise destroyed any SOFTWARE contained on such media or stored in such apparatus. LICENSEE may not distribute, lease, transfer, export, loan or otherwise convey the SOFTWARE or any portion thereof to anyone.

2. Copying Restrictions: In order to effect LICENSEE's license rights hereunder, it may install the SOFTWARE by copying it onto the hard disk drive or into the CPU memory of a computer system for use thereon, and may make full or partial copies of the SOFTWARE, but only as necessary for backup or archival purposes. LICENSEE agrees that (i) LICENSEE's use and possession of such copies shall be solely under the terms and conditions of this Agreement, and (ii) LICENSEE shall place the same proprietary and copyright notices and legends on all such copies as included by Boston Dynamics on the media containing the authorized copy of the SOFTWARE originally provided by Boston Dynamics.

3. Ownership: LICENSEE agrees and acknowledges that Boston Dynamics transfers no ownership interest in the SOFTWARE, in the intellectual property in SOFTWARE, nor in any SOFTWARE copy to LICENSEE under this Agreement or otherwise, and that Boston Dynamics and its licensors reserve all rights not expressly granted hereunder. After LICENSEE pays any applicable license fees, LICENSEE will own the media on which the SOFTWARE was originally provided hereunder and on which LICENSEE subsequently copies the SOFTWARE, but Boston Dynamics and its licensors shall retain ownership of all SOFTWARE and copies of the SOFTWARE or portions thereof embodied in or on any media.

4. Transfer Restrictions: LICENSEE may not sublicense, transfer, or assign this Agreement or any rights or obligations under this Agreement, in whole or in part.

5. Export Restrictions: LICENSEE agrees not to export or re-export, directly or indirectly, from the United States through any country or to any country of ultimate destination defined by the United States Government or any agency thereof as an "Excluded Territory". LICENSEE will not export, directly or indirectly, the Licensed Programs or Documentation to any country from which the United States Government or any agency thereof requires an export license or other governmental approval at the time of export without first obtaining such license or approval. The U.S. Government's list of "Excluded Territories" is subject to change without notice and is therefore not included herein.

6. Confidentiality: By accepting this license, you acknowledge that the SOFTWARE and accompanying materials, and any proprietary information contained in the media, are proprietary in nature and contain valuable confidential information developed or acquired at great expense. You agree not to disclose to others or utilize such trade secrets or proprietary information except as provide herein.

7. Enforcement of Terms: If LICENSEE fails to fulfill any of its obligations under this Agreement, Boston Dynamics and/or its licensors may pursue all available legal remedies to enforce this Agreement, and Boston Dynamics may, at any time after LICENSEE's default of this Agreement, terminate this Agreement and all licenses and rights granted under this Agreement. LICENSEE agrees that Boston Dynamics' licensors referenced in the SOFTWARE are third-party beneficiaries of this Agreement, and may enforce this Agreement as it relates to their intellectual property. LICENSEE further agrees that, if Boston Dynamics terminates this Agreement for default, LICENSEE will, within ten (10) days after any such termination, deliver to Boston Dynamics or render unusable all SOFTWARE originally provided hereunder and any copies thereof embodied in any medium.

8. Governing Law: This Agreement shall be governed by and interpreted in accordance with the laws of the Commonwealth of Massachusetts, excluding its choice of law rules.

9. U.S. GOVERNMENT PURCHASES: If SOFTWARE is acquired for or on behalf of the U.S. Government, then:

a. It is recognized and agreed that the SOFTWARE: (i) was developed at private expense; (ii) was not required to be originated or developed under a Government contract; (iii) was not generated as a necessary part of performing a Government contract; and (iv) was not accomplished during and necessary for the performance of a Government contract.

b. It is recognized and agreed that SOFTWARE is commercial computer software, as that term is used in FAR Parts 12 and 27.4 (and any applicable agency supplements thereto), and DFARS Parts 211.70, 227.4 (OCT 1988), 227.71 (JUN 1995) and 227.72 (JUN 1995).

c. If this purchase is subject to FAR Part 27, and FAR 52.227-19 has been incorporated into the terms of this purchase, the Government's rights in the SOFTWARE are no greater than RESTRICTED RIGHTS as specified in FAR 52.227-19.

d. If this purchase is subject to DFARS Part 227 (OCT 1988), the Government's rights in the SOFTWARE are no greater than RESTRICTED RIGHTS as specified in DFARS 252.227-7013(c)(1)(i).

e. The Government's rights in the SOFTWARE are no greater than the rights expressly stated in the body of this Standard Licensing Agreement, if this purchase is subject to (i) FAR Part 12; (ii) FAR Part 27, and FAR 52.227-19 has not been incorporated into the terms of this purchase; (iii) DFARS Part 211.70; (iv) DFARS Parts 227.71 and 227.72 (JUN 1995); or (v) any other regulation or clause permitting the contractor to deliver commercial computer software under the contractor's standard commercial license.

f. Any related technical data shall be delivered to the Government with no more than limited rights.

10. DISCLAIMER OF SOFTWARE WARRANTY: BOSTON DYNAMICS PROVIDES THE SOFTWARE TO LICENSEE “AS IS” AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR WITH RESPECT TO INTELLECTUAL PROPERTY OWNERSHIP, NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY ANY BOSTON DYNAMICS EMPLOYEE, REPRESENTATIVE OR DISTRIBUTOR WILL CREATE A WARRANTY FOR THE SOFTWARE, AND LICENSEE MAY NOT RELY ON ANY SUCH INFORMATION OR ADVICE.

11. Limited Warranty on Media: Boston Dynamics warrants the media on which SOFTWARE is recorded and provided to LICENSEE under this Agreement to be free from defects in materials and workmanship under normal use for a period of ninety (90) days after the date of the original delivery of SOFTWARE to LICENSEE. Such warranty is solely for LICENSEE's benefit and LICENSEE has no authority to assign, pass through or transfer this warranty to any other person or entity. If LICENSEE returns any defective media to Boston Dynamics or an authorized Boston Dynamics representative during the warranty period with proof of purchase, Boston Dynamics will, at its sole option, either replace such defective media or refund the purchase price for such media. This warranty will not apply to any media that has been damaged by abuse, accident or misuse. The foregoing warranty sets forth Boston Dynamics' entire liability and LICENSEE's exclusive remedy for any defects in any media and is in lieu of, and Boston Dynamics disclaims, all other warranties, express, implied, or otherwise, including without limitation any warranty of merchantability or fitness for a particular purpose.

12. LIMITATION OF LIABILITY: IN NO EVENT SHALL BOSTON DYNAMICS OR ITS LICENSORS BE LIABLE TO LICENSEE FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES OF ANY KIND (INCLUDING WITHOUT LIMITATION THE COST OF COVER, DAMAGES ARISING FROM LOSS OF DATA, USE, PROFITS OR GOODWILL, OR PROPERTY DAMAGE), WHETHER OR NOT BOSTON DYNAMICS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSS, HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY ARISING OUT OF THIS AGREEMENT. THESE LIMITATIONS SHALL APPLY NOTWITHSTANDING THE FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY. BOSTON DYNAMICS' LIABILITY ARISING OUT OF THIS SOFTWARE LICENSE AGREEMENT AND/OR LICENSEE'S USE OR POSSESSION OF THE SOFTWARE, INCLUDING WITHOUT LIMITATION ANY AND ALL CLAIMS COMBINED, WILL NOT EXCEED THE AMOUNT OF THE LICENSE FEE LICENSEE PAID FOR THE SOFTWARE PROVIDED UNDER THIS AGREEMENT.

13. Laws Governing Warranties and Liability: The law(s) of a jurisdiction may define the scope of warranty to be provided for products or the manner in which a supplier's liability may be limited, and such law(s) shall govern this Agreement only to the extent a party protected by such law(s) cannot waive the protection thereof by contract. In the U.S. and other countries, some states, territories or other principalities do not allow the limitation or exclusion of liability for incidental or consequential damages, or allow the exclusion of implied warranties, so the limitation and exclusion above may not apply to LICENSEE, and LICENSEE may have other rights that vary from state, territory or principality to state, territory or principality.

Soldiers in DI-Guy Scenario

Table of Contents

1. Introduction

Welcome to DI-Guy Scenario!

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'll 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.

What is DI-Guy Scenario?

DI-Guy Scenario is a tool for adding life-like human characters, *people*, 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 applications. The back flyleaf of this user guide has examples of DI-Guy Scenario applications.

Image from a training scenario created with DI-Guy Scenario. Soldiers, insurgents, vehicles and terrain come with the standard product. Optional special effects prove haze, smoke, fire, and explosions.

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 rolls 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.

DI-Guy Scenario Comes with People

DI-Guy Scenario comes with a full cast of life-like people. The DI-Guy Scenario 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, so you never have to get bogged down in the animation swamp.

DI-Guy AI is not in the default product

DI-Guy AI, software for creating autonomous, terrain-aware characters and crowds, is a cost option to DI-Guy Scenario. Information about DI-Guy AI is found in the DI-Guy AI manual. This manual focuses on the non-DI-Guy AI aspects of DI-Guy Scenario, but for a full experience users are encouraged to evaluate DI-Guy AI. There is a very strong coupling between the two products.

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 and CrowdBlitzer modes, 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 steps:

Step 1: Load a terrain model (DI-Guy Scenario supports Open Flight models).

Step 2: Choose a character type and appearance.

Step 3: Place the character on the terrain model and tell it where to go and what to do.

Step 4: View the scenario in real time from any point of view.

What Computers does DI-Guy Scenario run on?

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, Open Scene Graph , DirectX, and Vega Prime. DI-Guy can also be customized to a specific graphic 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.

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 that you can purchase to extend the product further. Contact your DI-Guy sales representative for information on 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 Scenarios, and other systems that use these ubiquitous networking standards.
- Expressive Faces – This module provides characters with faces that can show facial expressions, move their eyes, and synchronize their lips with speech.
- Plug-in Capability – DI-Guy Scenario now offers a plug-in architecture that allows you to create your own dynamic libraries (.dll) 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. Because human behavior is so diverse, however, 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 and DI-Guy Scenario. The DI-Guy Motion Editor has been engineered for seamless operation with DI-Guy and DI-Guys 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. See the DI-Guy AI Manual for more details.

Getting Started

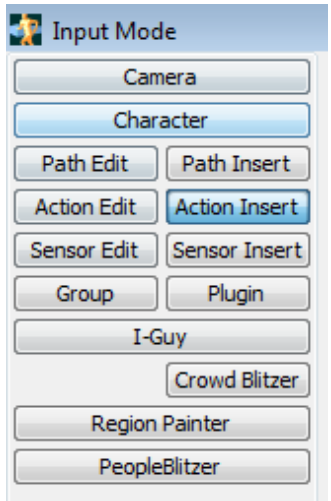
Chapter 2 of this manual is a self-contained tutorial that covers the essential features of DI-Guy Scenario. The tutorial will help you to become a proficient DI-Guy Scenario user in just a couple of hours. The remainder of the manual after the tutorial provides additional information on DI-Guy Scenario's user interface and features. Once you have completed the tutorial, use the manual as a reference.

2. Tutorial

This DI-Guy Scenario tutorial is designed to help you get started using DI-Guy Scenario quickly and easily. The tutorial concentrates on the basic functionality of DI-Guy Scenario, leaving advanced features for later. You can consult the later sections of this manual for advanced features and for instructions on how to "get under the hood" and work the details.

See Chapter 3 for starting DI-Guy Scenario in Dual Screen Mode

Wherever possible, DI-Guy Scenario lets you work graphically using your mouse directly in the 3D window. There are eight basic modes that control the graphical interaction:



Camera Mode	Alter your viewpoint.
Path, Character and Group Modes	Move character performances relative to the scene.
Action Modes	Assign and edit motions.
Sensor Modes	Assign and edit sensor regions that trigger events when characters enter them.
I-Guy Mode	1st person mode. "Drive" a character directly using keyboard, mouse, or joystick.
Plugin Mode	Mouse and keyboard communicate with plugin.
Crowd Mode	Edit or Blitz DI-Guy AI crowds.
Region Painter	Paint DI-Guy AI regions.
PeopleBlitzer	Place a character into the scene.

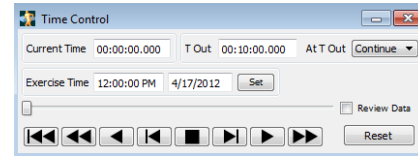
NOTE: DI-Guy AI modes are visible only with a valid DI-Guy AI license. See the DI-Guy AI manual for information on these additional modes.

Overview of the DI-Guy Scenario User Interface

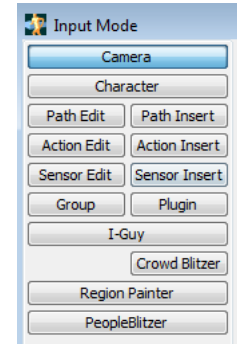
DI-Guy Scenario features eleven palettes and a 3D Display window for controlling the function of the program and characters. These are contained in a Main Window. Scenarios are loaded and saved from the Main Window menus. The palettes and 3D Display Window are managed from there as well.

DI-Guy Scenario 3D Display – Provides a view of the real-time 3D scenario, including the terrain model, people, vehicles, and their behavior. You work directly in the 3D Display window to interactively create, edit, and view scenarios.

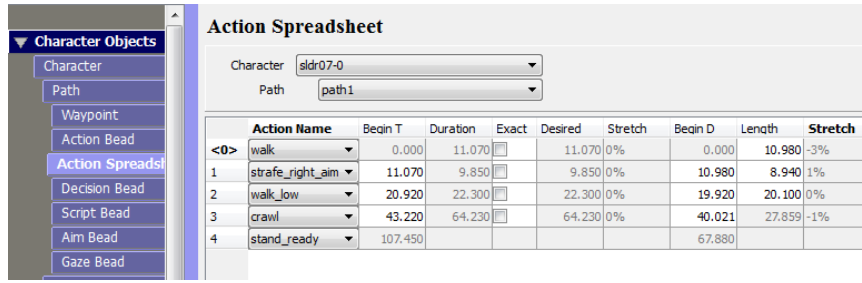
Time Control Palette – is a VCR-like interface that lets you play scenarios as well as reset, control looping, step through and otherwise control time in the simulation.



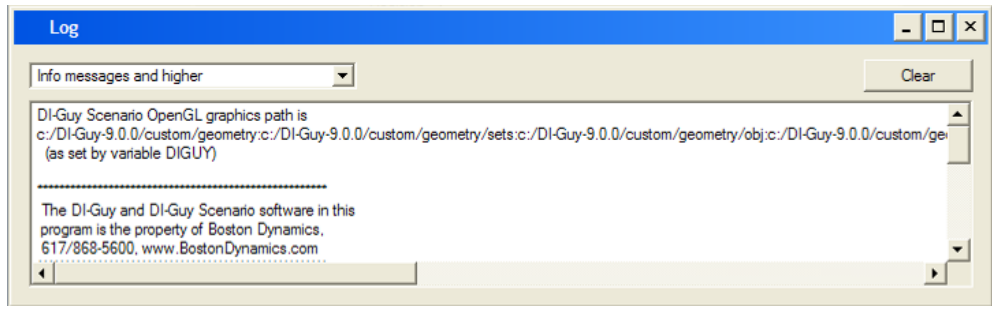
Input Mode Palette – selects the operations your mouse performs when clicked and dragged in 3D Display. The left side (pictured) shows the various modes. The right side (not pictured) changes with each mode and supports the mode selected. Some buttons may not be available if DI-Guy AI license is not active.



Action Spreadsheet Palette – displays detailed information about the selected character's sequence of actions, timing, and location along its paths. Actions can be added, deleted and changed, and numerical data can be typed in directly. This palette is a good place to edit or fine tune the timing of a character's performance.



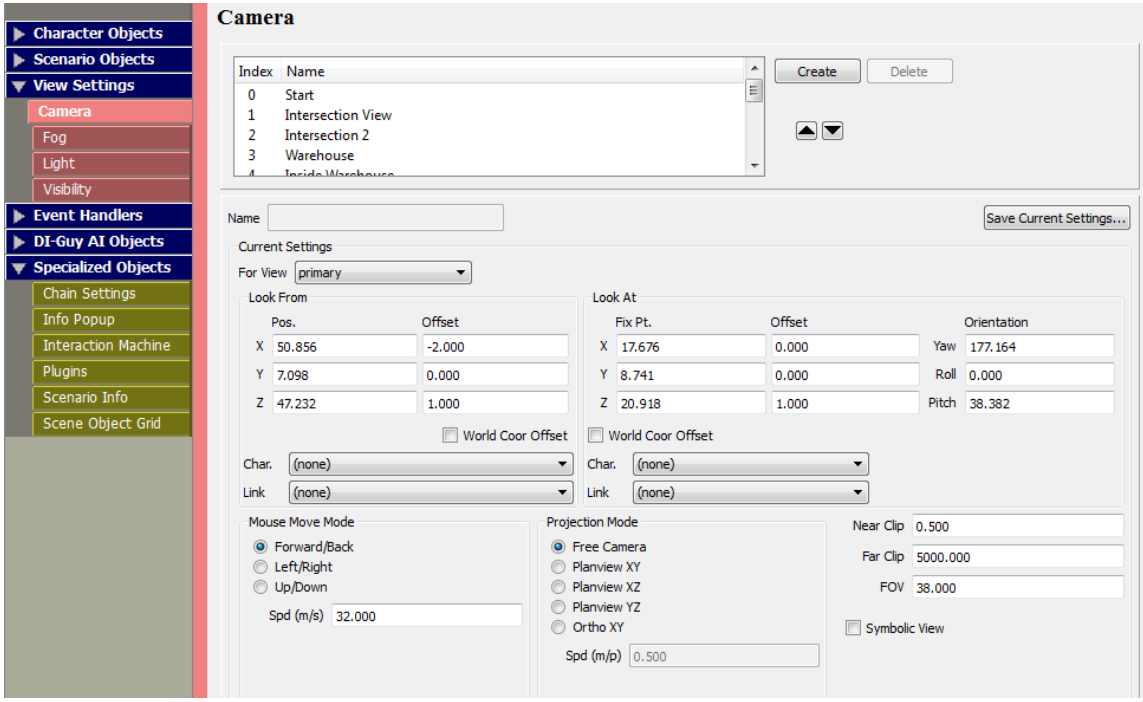
Log Palette – Provides error, warning, and general status messages. It's a good idea not to forget about this window; if things are not going as you expect, the answer to your problem may be found in a message to you here.



o

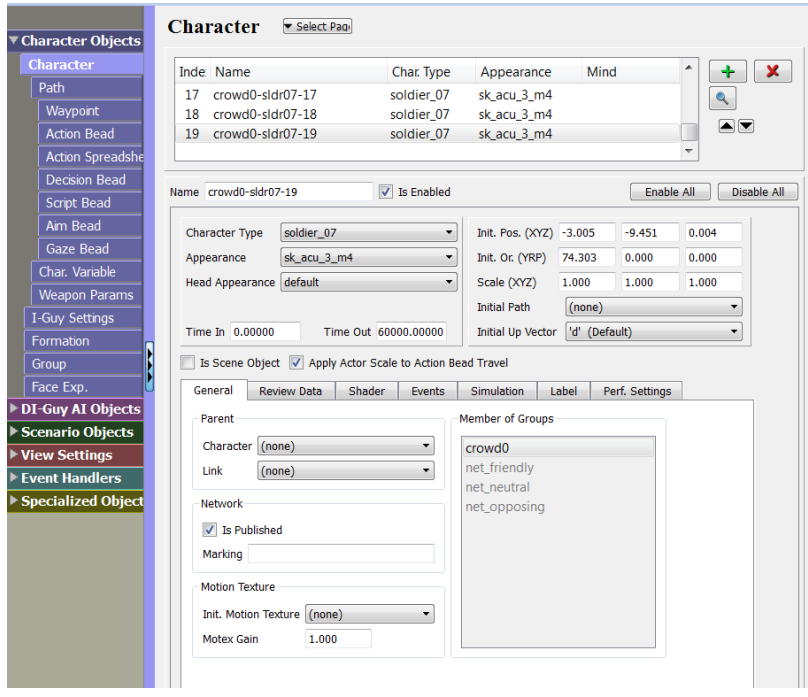
View Settings Palette – provides control over the virtual cameras, lighting, fog, and other parameters of how the scene is rendered. The View palette is composed of three pages or sub-palettes:

- Camera – allows you to define multiple camera settings, select among them, choose the type of camera motion and projection, adjust clipping planes and field of view, attach the camera to a person, and fixate on a person, vehicle, or prop.
- Fog/Sky – allows you to define multiple sky color and fog combinations.
- Light – allows you to specify multiple lighting conditions for the scene, including the parameters of the lighting model and the location of the sun.
- Visibility – allows you to control the visibility characters, paths, regions, AI indicators, particles, and other DI-Guy elements



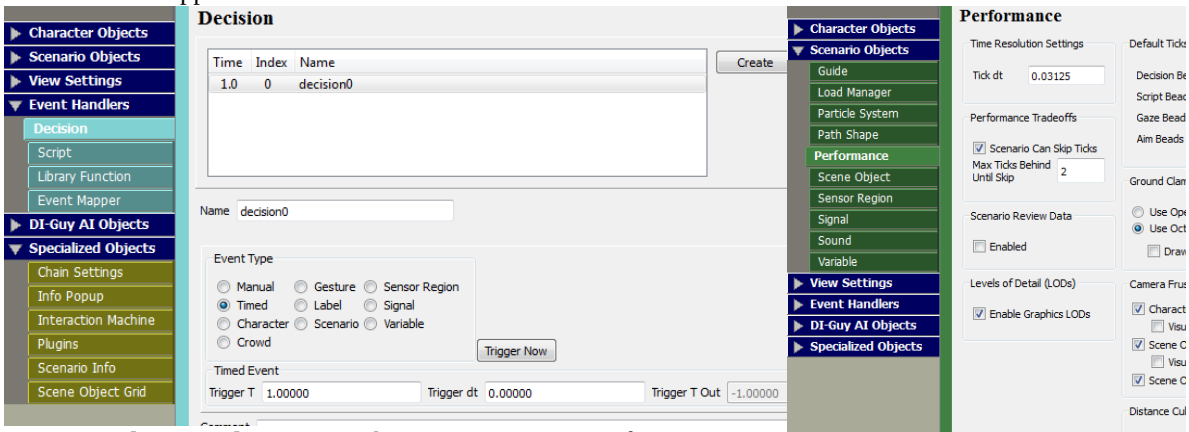
Character Elements Palette – Provides detailed access to individual characters, their paths, actions, decision logic, and other behavior. The Character Elements palette is composed of eleven tabs or sub-palettes:

- Character Tab
- Path Tab
- Waypoint Tab
- Action Bead Tab
- Action Spreadsheet
- Decision Bead Tab
- Script Bead Tab
- Aim Bead Tab
- Gaze Bead Tab
- Variable Tab
- Weapon Parameters
- I-Guy Settings
- Formation
- Group
- Face Exp.



Event Handlers Palette – Provides detailed access to event handlers. The Event Handlers palette is composed of four tabs or sub-palettes:

- Decision Tab
- Script Tab
- Library Function Tab
- Event Mapper Tab



Scenario Objects Palette – Provides access to a variety of parameters that affect the entire scenario. This palette is composed of twenty tabs or sub-palettes.

- Guide
- Load Manager
- Particle System
- Path Shape
- Performance
- Scene Objects
- Scene Object Grids
- Sensor Regions
- Signals
- Sound

Specialized Objects Palette

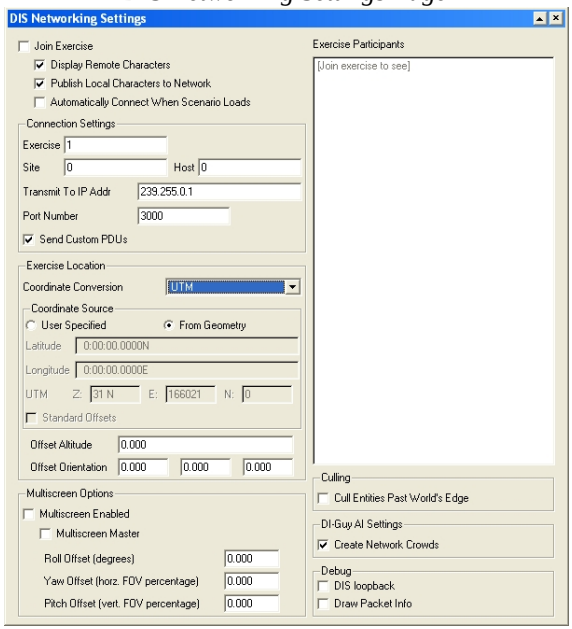
For miscellaneous parameters and settings not included in other palettes

- Chain Settings
- Info Popup
- Interaction Machine
- Plugins
- Scenario Info

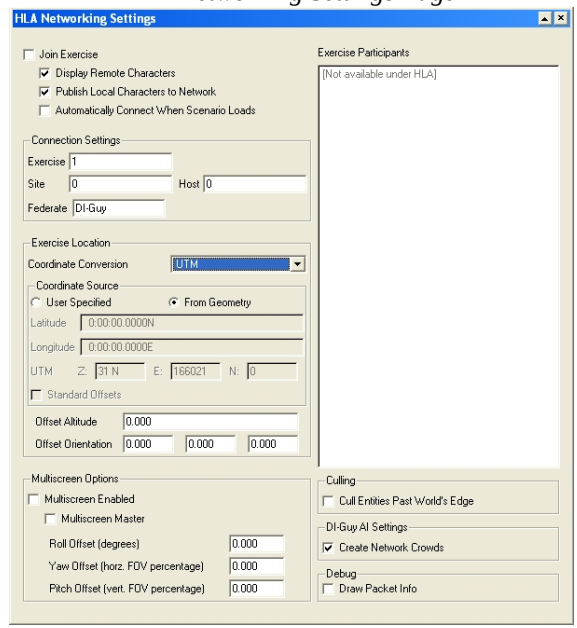
- Scene Object Grid

Network Settings Palette – Provides access to a variety of parameters that affect DIS or HLA networking

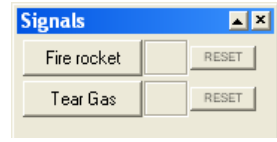
DIS Networking Settings Page



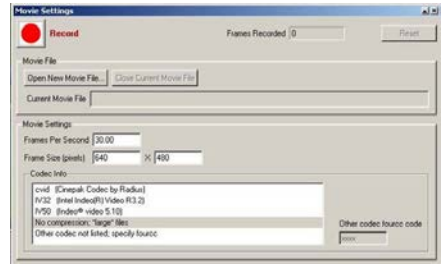
HLA Networking Settings Page



Signals Palette – displays signal buttons defined on the Signal tab of the Scenario Objects palette.



Movie Settings Palette – displays parameters for generating movies directly from DI-Guy Scenario



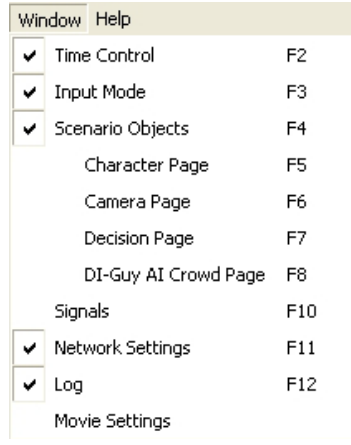
2.1 Tutorial 1: Jump In and Try DI-Guy Scenario!

The easiest way to learn DI-Guy Scenario is to use it. This section will get you started quickly and easily. We cover the basic features and functions, with just a small dose of theory of operation. This section

assumes you have already installed DI-Guy Scenario on your computer. If not, please go to Section 3 for installation instructions, and then come back here to resume the tutorial.

Start DI-Guy Scenario

After installation, find the DI-Guy Scenario icon on your desktop and double click it. It will take a few moments to startup. You will know that the program has completely loaded when the DI-Guy Scenario 3D Display window appears. Several palettes will appear on the screen during the loading process. For this tutorial, you will need the Control palette and the Mode palette. If either of them is not visible, use the Window pull down menu in the Main Window and select them and hide all others.



The Window pulldown on the Main Window allows you to specify which palettes are displayed, and which are hidden. Make sure there is a check next to Input Mode and Camera palettes. Once these palettes appear, move them to convenient positions on your screen.

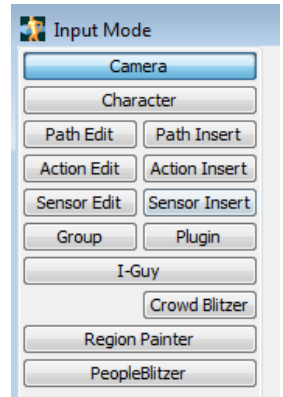
Terrain Models

Terrain models include the ground, roads, trees, buildings, and furnishings that make up a space. Typically users' first step is to load the terrain to be populated. For this tutorial, we will use the default terrain model, the *DI-Guy Scenario Stage*. The Stage is a simple model that offers a combined park, warehouse, and a mid-east city setting. Once DI-Guy Scenario is fully up and running, you should see the Stage in the 3D Display window.

tion

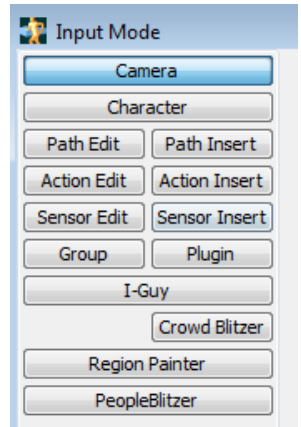
The DI-Guy Scenario Stage, a default terrain model that comes with the tool, is what you will first see in the 3D Display window when you start up DI-Guy Scenario. Use it for practice.

Move the Camera



The ability to move the camera in the 3D scene is an essential skill for using DI-Guy Scenario. So let's practice. Find the Mode palette and enter Camera mode. Place the mouse cursor in the DI-Guy Scenario 3D Display window and click to move the camera. Use the left button to move the camera forward. Use the right button to move the camera backward. Move the mouse while depressing the left or right mouse button to steer the camera motion. Hold down both the left and right mouse buttons at the same time and move the mouse to look around without moving the camera. If you get lost, press the "1" key, which returns you to a stored camera location. If the camera moves too slow or too fast for your taste, type '-' or '+' to adjust the speed.

Adding People

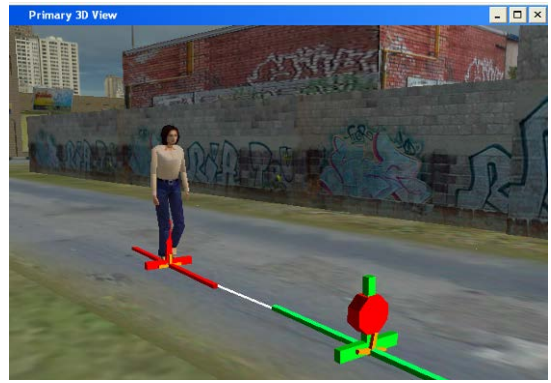


Let's add some people! We'll start by putting a person on the stage and having him walk from one place to another. Using the PeopleBlitzer you can add a person with just one click-and-drag of the mouse. Find the Mode palette and enter PeopleBlitzer mode.

The most efficient use of PeopleBlitzer mode is to click-and-drag, not just click. To click-and-drag, you place the cursor on an initial point of interest. Press the left mouse button and hold it down. With the mouse button still held down, drag the cursor to a second point of interest. Then release the mouse button. Click-and-drag is useful because it allows you to specify a location, direction and distance, all in one operation.

PeopleBlitzer

Now that you are in PeopleBlitzer mode and you know how to click-and-drag, decide where you want the person to start walking and where you want them to stop walking. Position the mouse at the starting point and click-and-drag to the stopping point. *Presto!* You have successfully added a person to the scene!

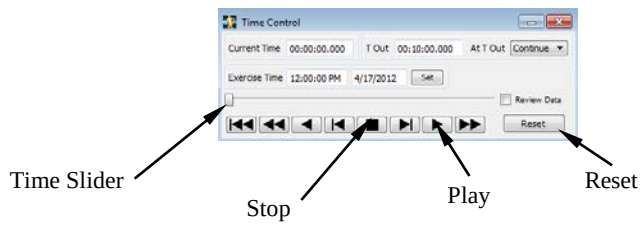


A freshly PeopleBlitzed person with some graphical elements indicating the path the character is on.

Place a second character using PeopleBlitz by just pointing and clicking without dragging the mouse; the person is created with a path that contains just a single waypoint. This mode is useful when you are setting up a new scenario and just want to see where folks should go and move them around. It is also useful for placing props and parked vehicles. Later, if you want these characters to travel, you can give them a path by inserting additional waypoints.

Play the Scenario

Now that you have placed people in the scenario, let's see them do something. Find the Control palette; it is setup like the controls for a DVD or VCR. Press the Reset button, and then press the Play button. The characters will walk along their path. After a while, hit the Stop button. Try playing the action back a second time at 4 times the normal speed using Fast Forward, or have the character Play Backwards. Move the camera around while the person walks backward. Hit the Stop button again



The Time Control palette behaves likes a DVD controller and allows you to control the play of time in the scenario.

You can cause the scenario to loop by selecting the loop box on the Control palette. You can also set the start and end times of the loop. Set the loop to run from t=0 to t=10 seconds and hit play.

Saving Your Work

You can save your work by creating a DI-Guy Scenario scenario file (.dss). To save a .dss file, use the File –> Save or Save As selections on the Control palette. Note that if you double-click on a saved .dss file, DI-Guy Scenario starts up and loads the scenario.

Note: The ability to save files is disabled when running DI-Guy Scenario in Demo Mode without a license.

End of Tutorial Section 1

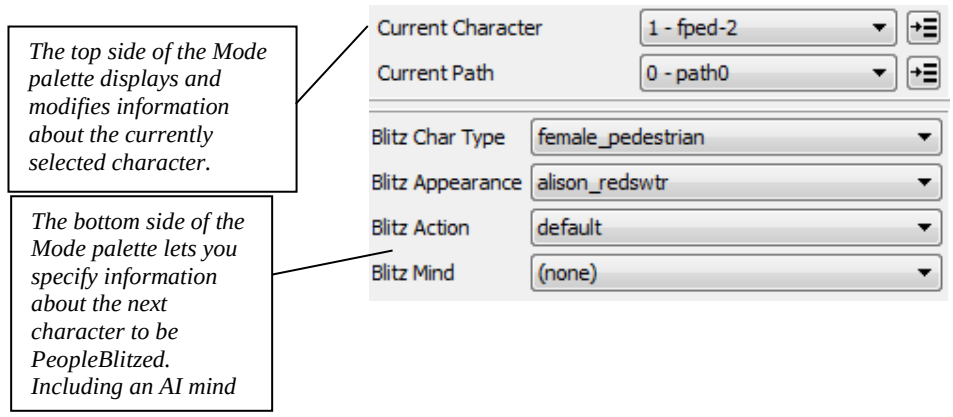
Congratulations. You have completed the first section of the DI-Guy Scenario tutorial. You have learned to navigate the DI-Guy Scenario user interface, manipulate the virtual camera, and create a simple scenario populated with lifelike people. In the next section we will focus on choosing people and their appearances.

2.2 Tutorial 2: Selecting People and their Appearance

Choosing People

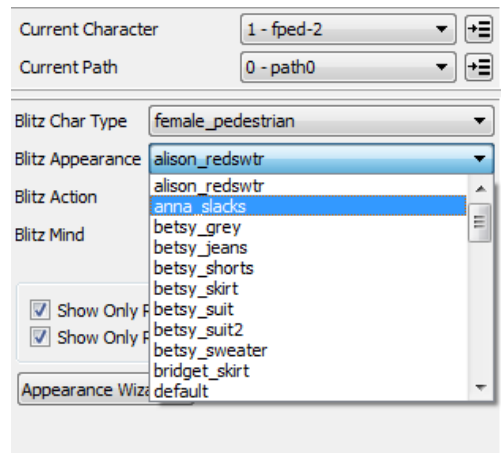
DI-Guy Scenario offers a wide range of characters with a variety of appearance and behavior. When you used the PeopleBlitzer earlier, you added a character with a default character type, appearance, and action. Now we will select character type, appearance, and action using the Input Mode palette and add a new person with desired characteristics.

.



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. You can select the character type for the People Blitzer using the pull-down list on the Mode palette. When you click on the *Char Type* field, it will show you a list of available character types. Choose mped_07 on the PeopleBlitzer side (the right side) of the Mode palette.



> Quick Tip: Survey all the available characters and audition their actions using the license-free ***DI-Guy Character Viewer*** application available from the Start Menu.

Appearances
Appearances specifies what a DI-Guy Scenario person looks like. Appearances include body shape, hair color, and clothing. Appearances can also include certain accessories, such as goggles, backpacks, canteen, etc. For each character type, there is a set of defined appearances. You can see the list of defined appearances for mped_07 by clicking on the appearance tab on the right side of the Input Mode palette. The illustration on the following tab shows what these appearances look like.

> Quick Tip: The mped_07, fped_07, and soldier_07 are the best character types to start with.

> Quick Tip: Appearances with the "sk_" prefix use superior deformable mesh graphics, while other appearances use older segmented polyhedra.

Vehicles
You may have noticed when looking at the list of character types that there is one called *vehicle* and another called *vehicle_wheels*. Character type *vehicle* includes cars, trucks, tanks, boats, aircraft that moves about the scene but don't have any moving joints. *vehicle_wheels* 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.

Like people, vehicles can be peopleblitzed into the scene and travel on paths. Naturally, since they don't have arms and legs, they don't walk, run, and crawl, but you can specify different travel speeds by setting their actions. DI-Guy Scenario comes with a library of vehicles.

Use the DI-Guy Character Viewer or consult the model guide documentation to explore the people, vehicles, and props in the DI-Guy data distribution.

Props
Props are static objects you can put into your scenario to enrich the scene. Examples of props are fences, furniture, food, or even houses. Props can be added 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. See Section 8 for instructions on how to add your own props to the library.

Effects
Effects are particle-based graphics for creating explosions, fire, smoke, dust trails, weather effects, etc. The use character_type "effect".

Changing Appearance

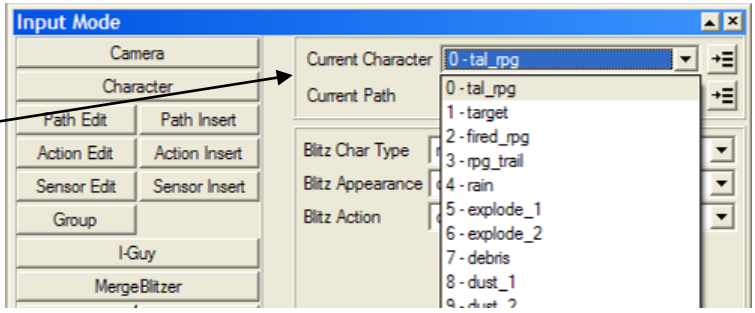
Once a character is in the scene, you can read or modify its appearance using the Input Mode palette. The right side of the Mode palette displays information for the current person. You can specify the appearance by clicking on the drop down arrow and making a selection from the list. Give it a try.

Selecting a Person

Now you select should turn white. If you look at the left side of the Mode palette, it should display the character type, appearance, and action of the person you just selected.

Another way to select a person is to use the Current Character field on the right side of the Mode palette. This field is a pull-down list of all the characters defined in the scene. You can select the one you want by highlighting it on the list.

List of characters in the scenario.



A third way to select a character and make them current is to click on the path associated with the person you want to select, provided you are either in Path Edit or Action Edit mode.

On the Mode palette, choose an appearance for the PeopleBlitzer using the pull-down list selector. Now that you have selected Character Type and Appearance, use the PeopleBlitzer to add a person to the scenario. Create a scenario in which a man and woman walk next to each other as they cross the stage and go back of the house. Play the scenario with the two characters.

Here is a sample of vehicle appearances available in DI-Guy Scenario. Use the Character Viewer to explore all the vehicles available.

Cut, Copy and Paste People

You can cut, copy, and paste people and their paths using standard menu selections or keystrokes. You can copy the current person and path to the clip board by going to the Control palette and selecting Edit –> Copy, or by typing "Control-C" while in PeopleBlitzer mode. This operation puts a copy of the current person, its paths, waypoints, and actions on the clipboard. You can introduce a copy of the person into the scenario by selecting Edit –> Paste in the Control palette, or by typing "Control -V." You can delete the current person using the Edit –> Cut or by typing "Control -X." Deleting a person puts the person on the clipboard, so you can paste it back into the scene.

Cut, copy, and paste works for people and for entire paths. It does not work for individual waypoints or action beads.

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

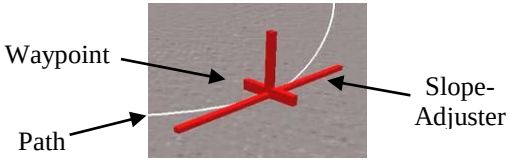
Undo

DI-Guy Scenario has multiple levels of undo. You can undo the last action by using Edit –> Undo, or by type "Control-Z."

End of Tutorial Section 2

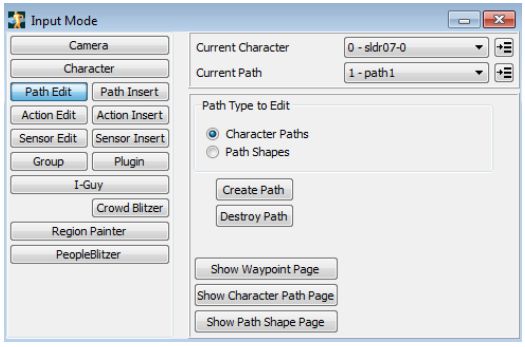
2.3 Tutorial 3: Paths

Paths & Waypoints One v
slope of the path where it passes through the waypoint. You can change the location and shape of a path by 1) repositioning the waypoint, 2) reorienting the waypoint, or 3) changing the slope adjuster.



Select 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; deselected paths are either blue or not displayed. To select a path, enter Path Edit mode on the Mode palette. Then left-click on the path or on the person associated with the path.

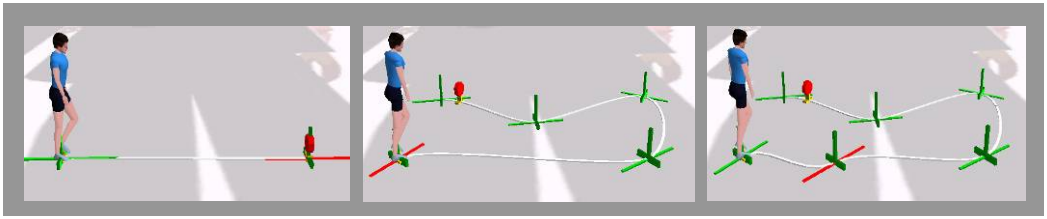
Adjust a Path
Let's change the path for one of the people you created by repositioning the waypoints. Enter Path Edit mode and make sure the path is selected. Using the left mouse button, click and drag on one of the waypoints to reposition it. Now reposition the other end of the path by repositioning the other waypoint.



You can also adjust the curvature of the path by manipulating the waypoint's slope adjuster. Using the left mouse button, click and drag on the slope-adjuster by clicking on either end of the bar. Drag the slope adjuster and observe the effects it has on the shape of the path. The longer the slope-adjuster is, the stronger the influence of the waypoint will be on the curvature of the path. You can make the paths smoother or sharper by using the slope-adjuster. Experiment with the orientation and length of the slope-adjuster. These adjustments will be important later on, when you want to craft detailed paths.

Note: 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.

Extend the Path Paths can have more interesting shapes when they have more waypoints. You can extend an existing path by adding waypoints. Enter Path Edit mode using the Mode palette. Select the last waypoint on the current path by clicking on it. Enter Path Insert mode and left-click where you want the new waypoint to go. Add more waypoints by pointing and left-clicking. Adjust the positions and slopes of the waypoints to contour your path as necessary.

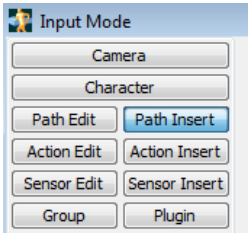


Path right after person was PeopleBlitzed into the scene.

Path was extended by adding three waypoints to end and smoothed by adjusting slopes.

An additional waypoint was inserted right after the first waypoint

Insert Waypoints



It is possible to add waypoints to the middle of a path. One can add waypoints to the middle of a path to get more control over the path shape and where the person will travel. 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. You can select a waypoint by clicking on it in Path Edit mode. If you select the first waypoint in a path and add a new one, the new waypoint becomes the second one and all others are redefined accordingly.

Go ahead and try it.

When you firs

End of Tutorial Section 3

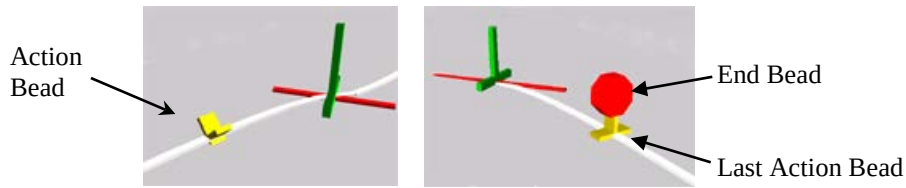
In this section of the tutorial you learned to create and edit paths. The next section of the tutorial focuses on actions.

2.4 Tutorial 4: Actions and Behavior

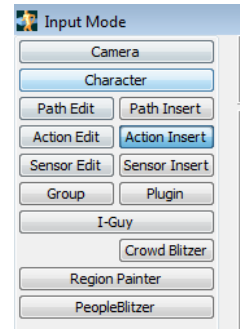
Actions People in DI-Guy Scenario are always performing *Actions*, even when they are just standing. For instance, walking, standing, and dying are all actions. Typically a person acts out a whole sequence of actions during a complex scenario. For instance, a person might start walking, then jog, then powerwalk, 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 in a smooth manner as well as coordinate their actions with their location in the terrain model and with the locations of other people.

Traveling vs. Stationary Actions
Some actions cause the person to travel along the path, such as walking, jogging, and crawling. These are generally 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.

Action Beads
Actions can be specified and controlled in DI-Guy Scenario by using *Action Beads*. Action beads appear in the DI-Guy Scenario 3D Display window 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 the small red stop sign.



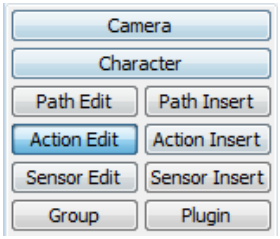
Adding Actions
Typically, you will want each person to perform several actions, one after another. Select a character and then go to the Mode palette and use the *Insert Action* field to choose which action you would like to add. When you click on this field it will give you a list of the actions available for the current character , as defined by its character type. Select one of the traveling actions, such as walking, jogging, or strolling. Then switch to *Action Insert* mode on the Mode palette and left-click on the path where you would like the new action to start. An action bead will appear where you click.



Playback the scenario and watch what happens when your character crosses the new action bead. It should make a seamless transition from the previous action to the one you just inserted.

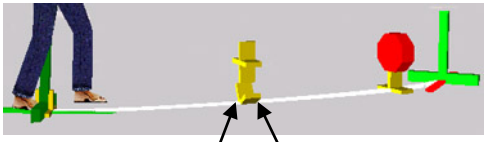
Dragging Actions

Release the mouse button when you are happy with the new action bead position.



Note that you can only drag the action bead along the path as far as the neighboring action bead. When you bump into a neighboring action bead, the two action beads will *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. Try it out.

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



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.

Note: When you specify a stationary action, its key parameter is duration (e.g. stand for 5 seconds) rather than a distance (e.g. walk 5 meters). The Action Spreadsheet palette, which is introduced below, gives you a numeric means of specifying times as well as distances.

Add a stationary action to one of the people in the scene. Select the person. Use the Insert Action field on the Input Mode palette to select a stationary action. "Stand" is a good one. Enter *Action Insert* mode. Point to the position along the path where you want the person to stand. Left click. Verify that the person does the right thing using the Control palette.

Note: You may have noticed that when you inserted the stationary action just now, two new action beads appeared 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.

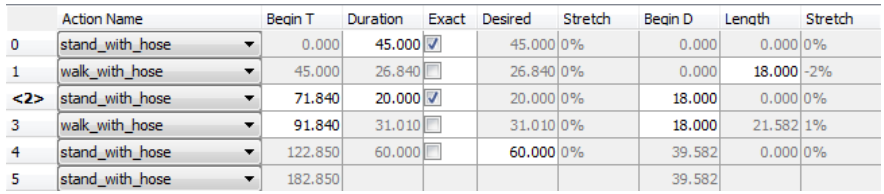
Changing Action

Once an action is specified, you can change it to another action. Enter *Action Edit* mode on the Mode palette. Select the action bead you want to change by left-clicking on it in the 3D Display window. The action bead will indicate that it was selected by rotating about its axis. Once the action bead is selected, you can change it by manipulating the pop-up list found in the Mode Palette's *Current Action* field.

Change the action to another traveling action, and verify the result with the VCR section of the Control window.

Action Spreadsheet

So far, you have used the Control palette, Mode palette, and 3D Display to add and modify characters in DI-Guy Scenario. Now you will use the *Action Spreadsheet* palette, an additional tool for viewing and manipulating the actions of a single DI-Guy character. The Action Spreadsheet is particularly helpful when working on complex sequences that involve both traveling and stationary actions.



	Action Name	Begin T	Duration	Exact	Desired	Stretch	Begin D	Length	Stretch
0	stand_with_hose	0.000	45.000	☑	45.000	0%	0.000	0.000	0%
1	walk_with_hose	45.000	26.840	☐	26.840	0%	0.000	18.000	-2%
<2>	stand_with_hose	71.840	20.000	☑	20.000	0%	18.000	0.000	0%
3	walk_with_hose	91.840	31.010	☐	31.010	0%	18.000	21.582	1%
4	stand_with_hose	122.850	60.000	☐	60.000	0%	39.582	0.000	0%
5	stand_with_hose	182.850					39.582		

The Action Spreadsheet palette displays a list of the actions a character will perform on his path. Each row of the Action Spreadsheet describes an individual action, including the name of the action, the starting time, duration, starting position, and displacement for the action. The starting position and displacement ("Length") are distances measured along the path, referenced to the first waypoint.

The Action Spreadsheet is a useful tool for crafting the actions of a character's performance. Use it to insert, delete, and modify actions. To insert an action, first go to the Input Mode palette to specify the specific action to be inserted. Then, in the Action Spreadsheet, select an existing action that is adjacent to where you want to add the new action. Use the *Insert* button to add the new action just before the selected one or use the *Append* button to add the new action just after the selected one.

To delete an action, select it in the left-most column of the Action Spreadsheet or by clicking on the action bead in the 3D window. Then click the *Delete* button.

To modify the duration or displacement of an action, you can type numerical data directly into any of the white cells of the spreadsheet. You can also change the named action from one to another using the pull-down lists.

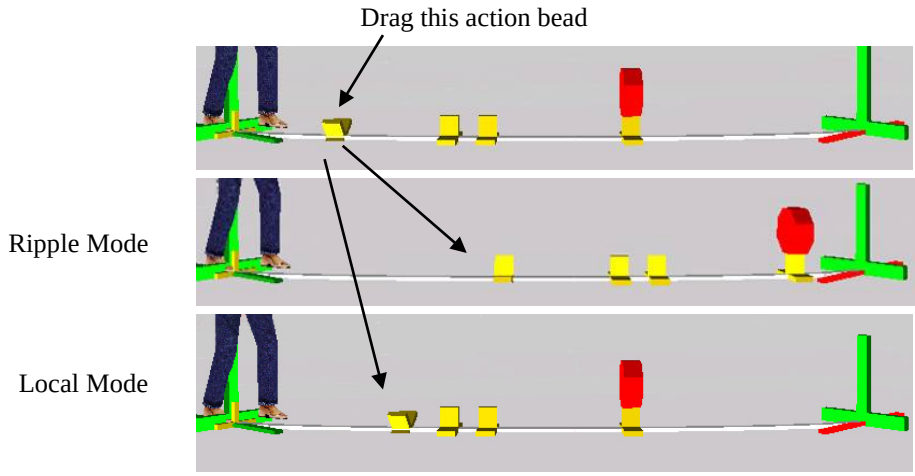
Local Mode and Ripple Mode

DI-Guy Scenario has two editing modes that affect what happens when you insert, delete, and drag action beads.

In Local Mode, DI-Guy Scenario attempts to localize the effects of any changes you make. When you drag an action bead, none of the surrounding action beads move. Local mode is useful when you have already specified the activity of a person later in the path and do not want to move it while working on earlier parts of the path. So far in this tutorial we have only used Local Mode.

In 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 a new action is deleted, all subsequent actions move toward the beginning of the path by the amount of the deleted action.

Similar behavior occurs when you drag an action bead in Ripple Mode. Ripple Mode is useful when you want to reposition a whole sequence of actions with respect to the terrain



When an action bead is dragged in Ripple Mode, all subsequent action beads move with the dragged bead, and the spacing between action beads is maintained. When an action bead is dragged in Local Mode, only the dragged action bead moves and subsequent action beads stay in position along their path.

Try Ripple Mode

There is a pair of toggle buttons on the Action Spreadsheet that allow you to alternate between Local and Ripple Modes. Switch to Ripple Mode, enter Action Edit mode, and experiment dragging action beads in the 3D Display. You will notice that all subsequent actions, including the end bead, keep a fixed distance from the action bead you drag.

Insert a new action. When you insert a traveling action in Ripple Mode, DI-Guy Scenario introduces one complete cycle of the new action, and shifts all subsequent actions toward the end of the path to make room. When you delete an action in Ripple Mode, DI-Guy Scenario shifts all subsequent actions earlier by the displacement of the deleted action.

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 the 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.

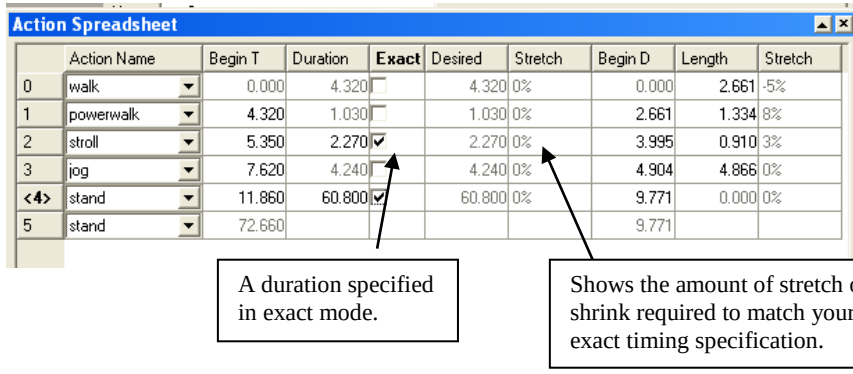
If you notice excessive feet slipping by a character as it travels, it is probably because the path is excessively stretched/shrunk. Examine the Action Spreadsheet and tune the behavior by adjusting the locations of nearby action beads to reduce the stretch or shrink shown in this field. A stretch/shrink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

Specifying Duration

DI-Guy Scenario calculates the duration of an action by choosing an even multiple cycles of the underlying motion. You can change the duration of a stationary action by changing the desired duration on the Action Spreadsheet. It will automatically round off to the duration that corresponds to an integer number of motion cycles.

Precise Timing

If you want to specify an exact duration for an action, click the *time exact* checkbox on the Action Spreadsheet and type in the required duration. 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. Keep in mind that the exact duration includes the transition time to get from the previous action to the current one.

End of Tutorial Section 4

In this section you learned about actions. The next section provides information about decision logic.

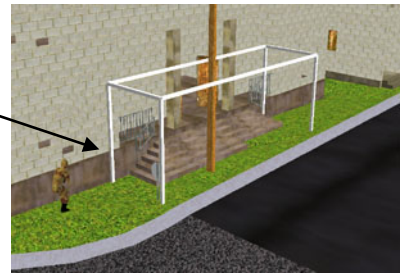
2.5 Tutorial 5: Decision Logic

DI-Guy Scenario has a decision-making capability that can be attached to any character. This section introduces the Sensor Region, Decision palette, Script Events, and Signals. It tells you how to hook them together to make people react to events in the scenario.

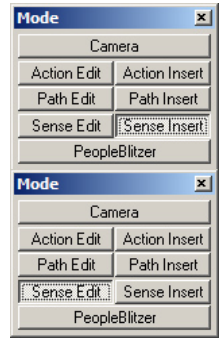
Sensor Regions

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 their presence and provides the information to the decision logic system.

Sensor Region used to test for person entering or leaving building.



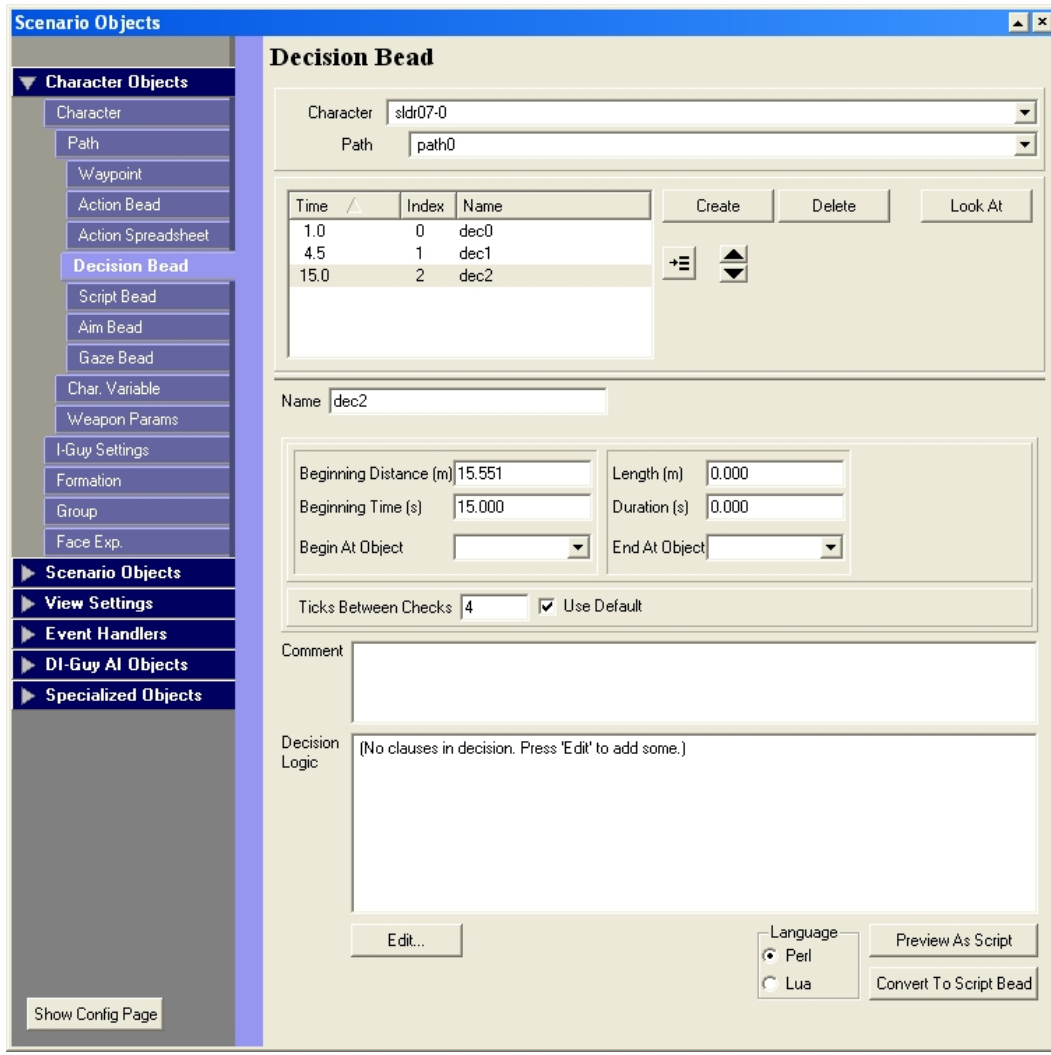
To create a new sensor region, select Sensor Insert on the Mode palette and then click-drag in the 3D scene. The first place you click will be one corner of the sensor region. The place where you release the mouse will be the diagonal corner of the sensor region.



Once you have created a Sensor Region, you can change its size or position by editing it. To edit a sensor region select Sense Edit on the Mode palette. Then use the mouse to drag the vertical lines that define the sensor region.

You can assign names to the sensor regions by going to the Sensor Region tab of the Scenario Objects palette.

Creating Decision Logic



DI-Guy Scenario lets you create reactive behavior for people and assign it to events in the scenario. The Decision tab of the Character Elements palette makes it easy to create decision logic without any programming!

To create a decision bead for your character, first use the “Create” button to create a bead, and then push the Edit button located at the bottom of the palette. You can then define an IF/THEN/ELSE clause that captures the reactivity you need. The Decision Bead tab of the Character Elements palette lets you do this graphically, and prompts you by providing lists of valid items selections at each step of the process.

Character 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. As you change the current character (by clicking on the character in the 3D Window in Path Mode, or by using the Mode Palette or the Character tab of the Character Elements palette the displayed set of decisions changes to those for the newly selected character.

Note that scenario-level decisions can also be created in the Event Handlers palette when a decision is more appropriately applied at the scenario level than at the character level.

IMPORTANT: 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, be careful! 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.

Creating Decision logics with programming: Lua Scripts

DI-Guy Scenario can do more advanced decision-making using Lua Scripts. Scripting is available on the Script Bead tab of the Character Elements palette, which is discussed in the reference section of the user manual.

Here is an easy way to get started on writing a Lua script: Create a decision bead as shown above. Then use the *Convert to Script Bead* button. Go to the script bead tab and you will see the decision logic converted into proper Lua code, ready for editing. You can use this feature as a template for creating your own Lua code.

End of Tutorial Section 5

In this section you learned about adding decision and script beads to a DI-Guy character path. The next section provides some information about camera, lights, fog, terrains, and keyboard shortcuts.

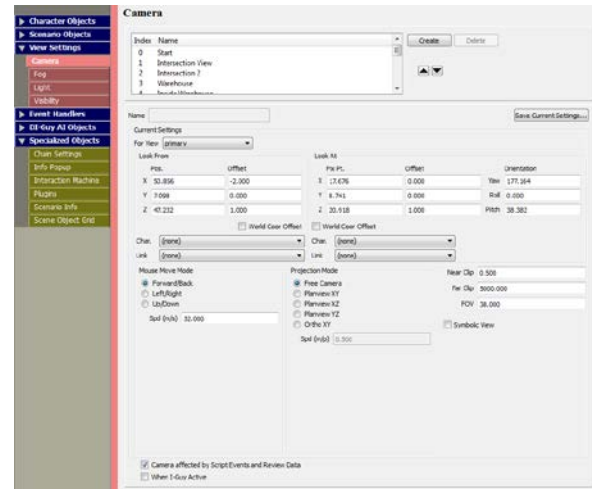
2.6 Tutorial 6: Cameras, Lights, Fog, Terrains, and Keyboard Shortcuts

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

Multiple Saved Camera Settings

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

Using the mouse, position the camera to achieve the view you want. Then type "<shift>#" where "#" is a digit from 0-9. This saves the camera location and viewing parameters as camera setting "#". You can instantly load saved camera settings by typing just the digit. You may need to click in the 3D Display window to make it active before typing the camera digit. You can access camera settings beyond the first 10 using the Camera palette.



You can use the Camera tab of the View Settings to alter a camera setting. Select the camera setting, then change the setting either numerically by altering values in the tab or by using camera mode in the 3D Window. Select *Save Current Settings* to then save the updated setting.

Additional Camera Motion

Normally the camera is set up to move forward and backward when using the left and right mouse buttons in the 3D Window when in Camera Mode. You can steer while traveling forward and backward by moving the mouse left/right, up/down in the 3D Display. 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 Display.

There are also modes that translate the virtual camera up and down and from side to side when you click the mouse. You can enter these modes through the Camera palette (Mouse Move Mode), or through hotkeys (Up/Down "d" Left-Right "s", Forward/Backward "f".) For any of these changes to take effect, the 3D Display must be active, which may require you to click in its window.

'Go There' Camera Control

Rather than 'fly' the camera from one location to another, the 'go there' camera automates movement of the camera. Just enter camera mode, 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.

Camera Tether

You can attach a camera to any person or vehicle in the scenario. When attached, the camera will travel with the person. Use the Camera palette to select the camera you want to attach. Then find the "Look From" box on the Camera Tab of the View palette. Use the "Char" pull-down field to select the person to attach to.

Once the camera is attached to a person or vehicle, you can adjust the *relative* positioning of the camera with respect to the person by mousing the camera position and orientation. Left and right mouse buttons take you closer or farther from the person. Holding both mouse buttons at once lets you change the camera's angle of view with respect to the traveling direction of the person. You can also adjust the camera travel settings by typing values at the "Pos Offset" fields in the Camera tab.

Camera Tracking

You can also create a camera setting that tracks a person or vehicle as it travels in the scenario. Specify tracking using the "Look At" filed on the Camera tab. By setting the camera's "Look From" to one person and its "Look At" to another character, you can show the relationship between two characters as they travel.

Camera POV

DI-Guy Scenario has camera modes that support 1st person POV (point of view). 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. Select POV mode by going to the Camera tab, specify the person or vehicle you want to attach to in the "Look From" field. Also select the same character in the "Look At" filed. Then use the mouse to position the camera to get the view you desire. The left mouse button positions the camera closer to the character. The right mouse button pulls the camera back from the character. Hold both mouse buttons down at the same time and drag to change the viewing angle.

You can save the POV camera view by saving the camera settings. Give the newly saved camera settings the name "POV." Adjust the camera so that it looks just over the shoulder of a person as they jog through the scenario. Adjust the camera so it looks sideways at a person as they walk. Now adjust the camera so that it looks at a person's face as they walk.

Advanced Camera Settings

You can adjust several other camera parameters, including the near and far clipping planes, field of view, and projection mode. These adjustments are done on the Camera tab of the View palette. Make the Camera tab visible using the Windows menu in the Control palette. All of these settings will be saved when you save camera settings and restored when you load one of the saved camera settings.

Loading Your Own 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, go to the Scenario Objects palette, click on the Scene Object tab, and select a new file to load for the scene object. You can also browse to your terrain model.

Note: 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.

Terrain models and tools for creating them are available from a number of sources. We are happy to help create terrains for users, and can also recommend terrain building specialists.

We have included several terrain models with the DI-Guy Scenario distribution for demos and for you to experiment with. They are located in $DIGUY/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 – built for DI-Guy 11, a few blocks of a western city and a few blocks of a Mideast street.
- Default Stage 7 – built for DI-Guy 7, a combination of a park, warehouse, and an Arab street.
- Default Stage – a very simple house on a very simple road.
- Desert – A Boston Dynamics model.
- Environment sky dome.
- Jailhouse
- Khost Afghaniston
- Middle East Town – A Boston Dynamics model.
- 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 PA. The model was created by San Jose State University, Fakespace Inc, and MultiGen-Paradigm.
- Railroad crossing
- Sadr City
- Sunnyvale – A model of Sunnyvale California 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.

Keyboard Shortcuts

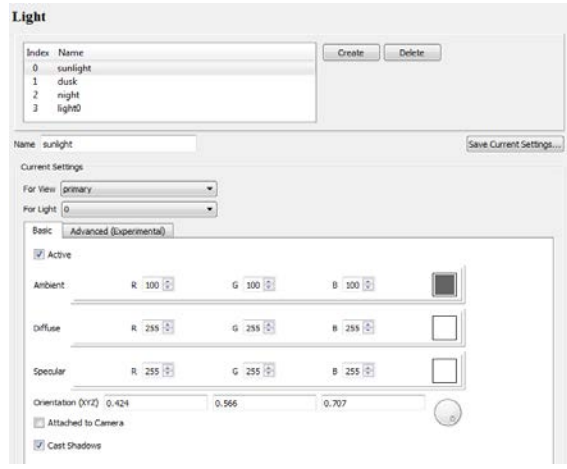
DI-Guy Scenario has keyboard shortcuts and hotkeys that accelerate use of the tool, once you become familiar with the functionality. The most important shortcuts are provided by the <ALT> and <CTL> keys. Holding the <ALT> key down temporarily switches you into Camera Mode, no matter which mode is selected on the Mode palette. When you release the <ALT> key the system reverts to the previous mode. This hotkey makes it easier to reposition the camera while crafting paths and action sequences.

<CTL> is another special hotkey. When you are in Path Edit, Action Edit, or Sense Edit modes, the <CTL> key momentarily shifts you into Path Insert, Action Insert, or Sense Insert modes, respectively. This allows you to work on positioning a set of waypoints, action beads or sensor regions, then momentarily switch to an insert mode to add a new element, and then revert back to tweaking positions, orientation, etc.

For a full list of available hotkeys, see Section 14.

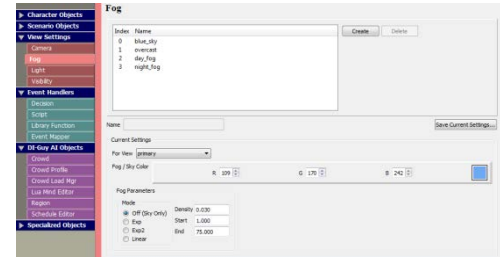
Lighting

There are several parameters that allow you to adjust lighting effects in the 3D scenario. These parameters include the position of the light source, and the ambient, diffuse, and specular components associated with the light source. These parameters are adjusted on the Light tab of the View Palette.



Fog

DI-Guy Scenario lets you adjust sky color and introduce fog or haze effects. These features are available on the Sky Tab of the View Palette.



End of Tutorial Section 6

In this section you learned more about cameras and settings, about loading your own terrain models, about adjusting the sky color and lighting, and about creating fogs. The next section of the tutorial provides advanced information about branched paths and parenting one character to another.

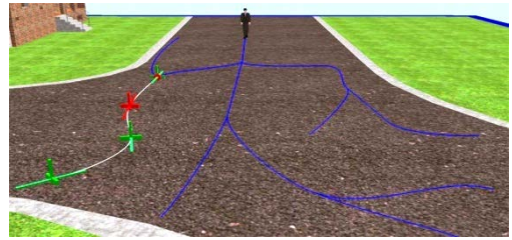
2.7 Tutorial 7: Branched Paths, Parenting, and Gestures

Branched Paths

Suppose you want a character to decide which way to go in a scenario, based on a decision. While using DI-Guy 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.

To create additional paths for a character, select the character, and then go to the Path palette of the Character tab. Click the create button. That action creates a new path whose first waypoint is coincident with the selected waypoint on the previous path.

You can edit the new path, adding and repositioning waypoints, by entering Path Insert and Path Edit on the Mode palette. You can also assign actions and decisions to the newly created path. You can add any number of paths to characters using this approach, creating a network of branching, as shown in the figure. This style of pathing uses global paths.



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 installation.

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

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 is accomplished through the *set_parent* action of decision logic. To create a parent/child relationship, first select the child character. Then create a decision bead that uses the set_parent function

in the THEN or ELSE sections of the Decision tab of the Character Elements palette. The set_parent action takes the name of the parent character as an argument.

Once parented, the child will be 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 fiddle to get the child where you want it. One trick to help accomplish this is to load the parent as a scene object at the origin and then blitz the child character onto the scene object. Later you can blitz in the parent as a regular character and then perform the parenting.

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.

Head nods and shakes

Head nodding (*yes*) and head shaking (*no*) are two gestures available in DI-Guy Scenario. 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 motion 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, select the character of interest in the normal way. Then create a decision bead for that character. In the THEN or ELSE field of the decision bead specify the *nod_head* or the *shake_head* action. The two parameters following the action on the Decision Bead tab of the Character Elements palette are the duration of the action, then 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 in a similar way.

General 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, use the *execute_gesture* action using decision logic. You can find a list of available gestures in the on-line documentation located at $DIGUY/doc/index.html, available once you have done a full install of DI-Guy Scenario on your system. Click on DI-Guy Data reference and scroll down to the gesture section. 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.

End of Tutorial Section 7

That's it! You have completed the entire DI-Guy Scenario tutorial. DI-Guy Scenario has more features and functions, but you know enough to get some real work done and create some exciting scenarios. So go practice and get busy. There is a lot of power hidden inside the Character and Scenario Objects

palettes, and skilled users can get even more functionality by using embedded scripting or by writing C++ plugins to DI-Guy Scenario that take full advantage of the DI-Guy API. When you get stuck, read the reference section of the manual to learn more.

Additional Documentation

When you install DI-Guy Scenario on your computer, online documentation for DI-Guy Scenario, the DI-Guy SDK, and the DI-Guy Motion Editor are installed automatically. They can be accessed through the start menu under DI-Guy, or you can access the *DI-Guy Documentation Master Index*, which is located at $DIGUY/doc/index.html. It has pointers to a variety of information about DI-Guy Scenario, DI-Guy characters, appearances, behavior, gestures, and the DI-Guy SDK,

Your Feedback

Please let us know how DI-Guy Scenario is working for you and your scenarios. If you have any problems with DI-Guy Scenario or with this tutorial, or even if you want to say something nice, send your comments to diguy@diguy.com

3. Launching DI-Guy Scenario

3.1 Installation

To install the DI-Guy Scenario software on your computer, put the DI-Guy Applications CD in your CDROM drive. On most systems an install menu will automatically appear. If it does not appear on its own after about a minute, find SETUP.EXE on the CD and double click on it. Then follow the installation directions on the screen.

After installing the DI-Guy Application, install DI-Guy Data CD1. Put DI-Guy Data CD1 in your CDROM drive. On most systems an install menu will automatically appear. If it does not appear on its own after about a minute, find SETUP.EXE on the CD and double click on it. Then follow the installation directions on the screen. At the appropriate time, the installer will prompt you to insert the other data CDs.

3.2 Starting DI-Guy Scenario

There are several ways to start DI-Guy Scenario:

1) double click on the DI-Guy Scenario desktop icon,
2) click on DI-Guy Scenario in the DI-Guy section of the Windows Start menu,
3) double click a scenario file with the .dss extension,
4) via the DOS command prompt.

For brevity's sake, the directions that follow assume DI-Guy Scenario has been installed in C:\DI-Guy.

Use the Desktop Shortcut

When you install DI-Guy Scenario on your computer, a shortcut is placed on the Windows desktop. You can use it to start DI-Guy Scenario.

Launch DI-Guy Scenario from the Start Menu

1) Navigate the Start Menu to the DI-Guy Scenario folder.
2) Select the DI-Guy Scenario shortcut.
3) Once the program has started, select File | Open and select `Surface_to_Air.dss` from the list of files.

Double-click a .dss file

1) Open the `c:\DI-Guy\scenarios` folder in Windows Explorer.
2) To start the program running the demo scenario, double-click `Surface_to_Air.dss`.

Launch DI-Guy Scenario from a command prompt

1) Change to the drive in which DI-Guy Scenario was installed.
   ```
   c:> c:
   ```
2) Change to the scenarios directory.
   ```
   c:> cd \DI-Guy\scenarios
   ```
3) Run the demo scenario.
   ```
   c:\DI-Guy\scenarios> diguyscenario Surface_to_Air.dss
   ```

3.3 Dual Screen Mode and other Startup Options

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

c:> diguyscenario –fullscreen2 –hide_menu –hide_status –no_digs_overlay

c:> diguyscenario –fullscreen

3.3.1 Startup Options

You can start *DI-Guy Scenario* with several options. The options can be specified either by changing the target line of a *DI-Guy Scenario* shortcut, or typing the option name after "DIGuyScenario" on a DOS command line. Here is a list of options

```
DIGuyScenario [options] [files]
```

`-D`	Start DI-Guy Scenario in Demo Mode.
`-p <play_mode>`	Specifies the initial play mode when DI-Guy Scenario first 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 press the play button.
`-fullscreen`	DI-Guy Scenario starts up with the 3D Display filling the entire screen of the display, without a border.
`-fullscreen2`	DI-Guy Scenario starts up with the 3D Display filling the entire screen of the display, without a border, on the second screen.
`-hide_menu`	Hide menu bar
`-hide_status`	Hide status bar
`-net`	If network module enabled, go immediately on-line.

`-N <int>`	Specifies verbosity of file log output, 1 = nothing, 15 = everything.
`-L <int>`	Specifies verbosity of 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. This can be done to reduce or increase the working set of available characters. The default is `diguy.cfg`. The configuration file is typically placed relative to the `$DIGUY/config/` directory.
`-V <cfg_filename>`	Use specified config file as network configuration file (default is “diguy/scenario/network.cfg”)
`Files`	Scenario files (.dss extension) listed on the command line will be opened when DI-Guy Scenario starts. The first file will be opened as if by File | Open. The rest will be opened as if by File | Merge.
`+plugin <plugin>`	Add specified plugin (dll)
`-plugin <plugin>`	Remove specified plugin (dll)
`+module <module>`	add module (modules listed below) 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 module (see list above)
`-pmu`	Have mouse_func() called in plugin when in plugin mode.

Examples:

Start the demo.dss scenario and begin immediate playback:

```
c:> DIGuyScenario -p play -N 0 demo.dss
```

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

```
c:> DIGuyScenario -N 6 -f log.txt demo.dss
```

4. The Main Window, Time Control, and Input Mode palettes

DI-Guy Scenario is controlled by a number of dialog boxes, or palettes, contained within a Main Window (actually, palettes can be floated outside the Main Window as an option, but the concept remains). This chapter describes the 3 most

4.1 Main Window

The Main Window menubars load, merge, and save Scenario files and it is the place to go to exit from DI-Guy Scenario. In addition, managing the palettes (GUI windows) occurs here.

The current scenario filename is listed in the title bar. A '*' next to the filename means the scenario has not been saved since last modified.

File Menu

The file menu is used to load DI-Guy Scenario scenario files (.dss) and terrain models, to save Scenario files, and to exit the program:

`File | New`	Create a new scenario (containing default terrain)
`File | Open`	Open a saved scenario or terrain model. When a scenario file is opened the program closes any existing scenarios (as if File | Close were used).
`File | Merge`	Merge a scenario file into the current scenario. The elements in the merged file will be appended to those that already exist.
`File | Save`	Save changes to current scenario file.
`File | Save As`	Save to a new scenario file name.
`File | Close`	Close and remove the existing scenario, including all characters, waypoints, and beads. If the file has been changed since the last save, a warning will appear.
`File | Exit`	Exit the DI-Guy Scenario program.

Note: When merging two scenarios via the Main Window, both will have the same start time and they will play simultaneously.

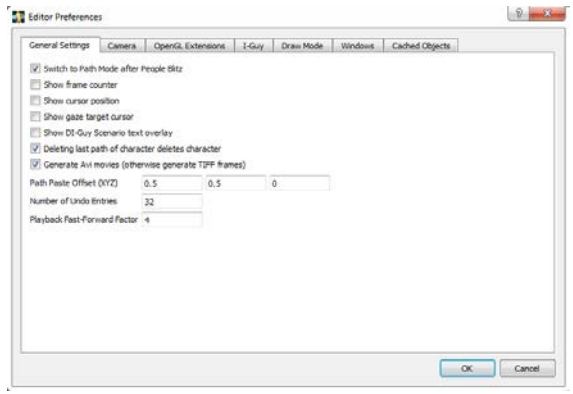
Edit Menu

The Edit Menu is Mode-sensitive. Depending on whether you are in camera mode, PeopleBlitzer mode, path mode, action mode, or sensor mode in the 3D Window, the edit functions will cut, copy, and paste camera settings, characters, paths, etc. Following is a description of edit when in PeopleBlitzer mode.

`Edit	Cut`	Remove the current character from the scenario. The character is saved on the scenario clipboard.
`Edit	Copy`	Copies the current character to the scenario clipboard.
`Edit	Paste`	If there is a person on the scenario clipboard, that person is pasted into the current scenario. The program may change the person's name to avoid name collisions with existing people.

4.1.1 Preferences

Preferences allow you to customize DI-Guy Scenario's behavior to your personal tastes. You can display the Preferences Palette by selecting Edit –> Preferences from the Main Window.



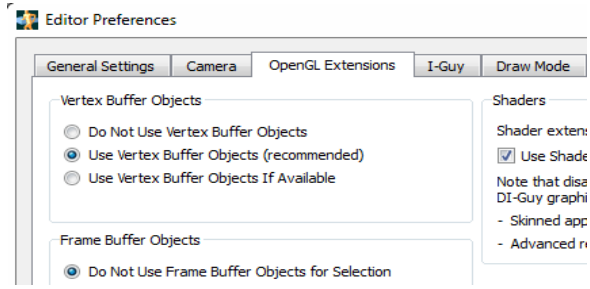
Preferences - General Settings

Switch to Path Mode After PeopleBlitz	When you use the PeopleBlitzer to add a new character to the scene, it is often convenient to adjust the character's path by adjusting the waypoints that define the path and by adding additional waypoints. This preference automatically switches DI-Guy Scenario from the PeopleBlitz mode into the Path Edit mode right after adding a character to the scene.
Show frame counter	This preference causes the frame rate to be displayed in the title bar of the 3D window.
Show cursor position	This preference causes the 3D coordinates of the mouse cursor to be displayed in an overlay of the 3D window. 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	This preferences causes a marker to be display in the 3D window that shows the fixation point for the character's gaze
Show DI-Guy Scenario overlay	Toggle display of "DI-Guy Scenario by Boston Dynamics".
Moving camera with mouse disables camera script events	When the user points into the 3D scene in with the camera mode selected, it means the user wants to adjust the camera location. This preference automatically turns off scripted camera events so they do not interfere with the user's desire to position the camera on their own.
Moving camera with mouse clears Camera Palette "Look From" character	When the user points into the 3D scene with the camera mode selected, it generally means the user wants to adjust the camera view. If the camera is locked to move with one of the characters, it is difficult to adjust the scene. This preference automatically frees the camera from any character when the user attempts to move the camera.
Moving camera with mouse clears Camera Palette "Look At" character	When the user points into the 3D scene with the camera mode selected, it generally means the user wants to adjust the camera view. If the camera is locked to track one of the characters, it is difficult to adjust the scene. This preference automatically frees the camera from any character when the user attempts to move the camera.
Deleting last path of character deletes character	Some characters have more than one path. When you delete a path, DI-Guy Scenario does not necessarily delete the character. This preference causes the character to be deleted when you delete that character's last path.
Show Avi Movie Maker	Defaults to on, uncheck to show the DI-Guy 8.0 style 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 will attempt to track, the more entries the more memory DI-Guy Scenario will use.

Playback Fast Forward Factor	The speed that fast forward runs at.

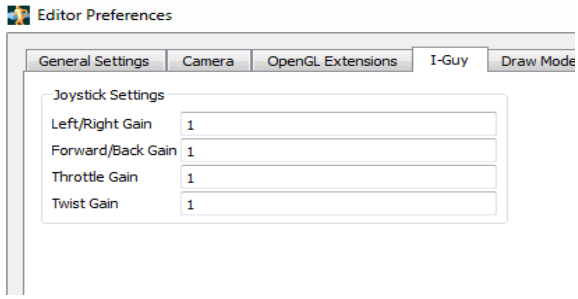
Preferences – OpenGL Extensions

This preference balances the use of Display Lists vs. using Vertex Buffer Objects. Using the toggle buttons, you can select to not use VBOs, use them, or use them when available.



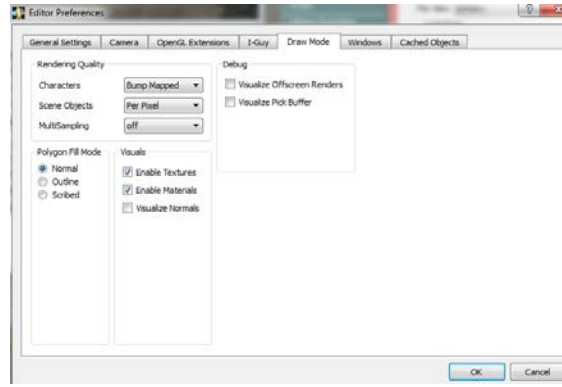
Preferences – I-Guy

Adjust the various gains related to joysticks.



Preferences – Draw Mode

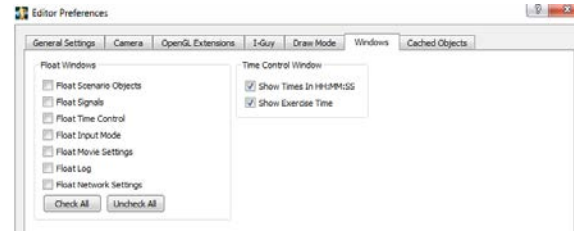
The Draw Mode tab of the View palette changes the rendering mode of DI-Guy Scenario. 3 x 2 modes are available, as shown below.



Images created using normal draw mode with textures (upper left), outline draw mode with no textures (upper right), scribed draw mode with textures (lower left), and normal mode with no textures

Preferences – Windows

DI-Guy Scenario 7.0 introduced the Main Window. By default, this window contains all the sub-windows (palettes). However, using these preferences, the various palettes can be *floated* so that they escape the Main Window constraint.



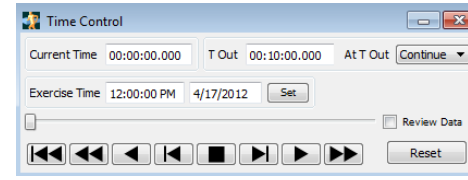
Window Menu

`Window | Time Control`	F2	Toggles visibility of Time Control palette.
`Window | Input Mode`	F3	Toggles visibility of Input Mode palette.
`Window | Action Spreadsheet`	F5	Toggles visibility of Action Spreadsheet
`Window | Signals`	F6	Toggles visibility of Signal palette.
`Window | Scenario Objects`	F7	Toggles visibility of Scenario Objects palette
`Window | Character Elements`	F8	Toggles visibility of Character Elements palette
`Window | View Settings`	F9	Toggles visibility of View Settings palette.
`Window | Event Handlers`	F10	Toggles visibility of Event Handlers palette
`Window | DI-Guy AI Objects`	F11	Toggles visibility of DI-Guy AI palette (only with AI module)
`Window | Log`	F12	Toggles visibility of Log palette
`Window | Network Settings`		Toggles visibility of Network Settings palette
`Window | Movie Settings`		Toggles visibility of Movie Settings palette
`Window | Secondary 3D View (0-8)`		Toggles visibility of a secondary view

Help Menu

`Help | About`	DI-Guy Scenario about box. Shows version number and creation date.
`Help | Host ID`	Displays the Host ID of the computer running DI-Guy Scenario. This ID is used by Boston Dynamics to create license keys.

4.2 Time Control Palette



This palette uses a simple VCR-type paradigm to stop, start, reset, and manipulate time for the scenario. If time is stopped, people will not move. If time is playing, people will

move forward in real time along their paths performing their scripted motions. Reset, rewind, fast forward, and loop all provide even greater control.

At T Out Options

This drop-down allows you to specify what should happen when the scenario reaches its end time. In loop mode, the scenario will reset and play from the beginning, which is useful for demos and tradeshows. In stop mode, playback will terminate. In continue mode, playback goes on, which can be useful for experimentation.

Continue	Continue to play
Loop	Reset scenario, continue to play
Stop	End playback

Time edit boxes

`Time`	current time
`Start Time`	starting time of playback range
`T Out`	ending time of playback range

Play buttons

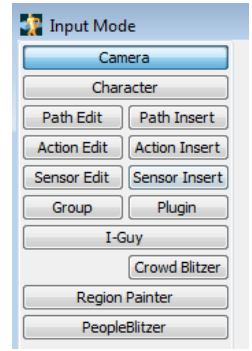
`|<`	rewind to Start Time
`<<`	fast reverse (4x realtime)
`<`	play in reverse
`<]`	step back one frame (.033 seconds)
■	Stop
`[>`	step forward one frame
>	Play
>>	fast forward (4x realtime)
`Reset`	turns off events and resets characters, options and time back to zero

Time slider

Moving the slider will scrub through time between Start Time and Loop Time.

Exercise Time - Correlates the start time with a real-world time

4.3 Input Mode Palette



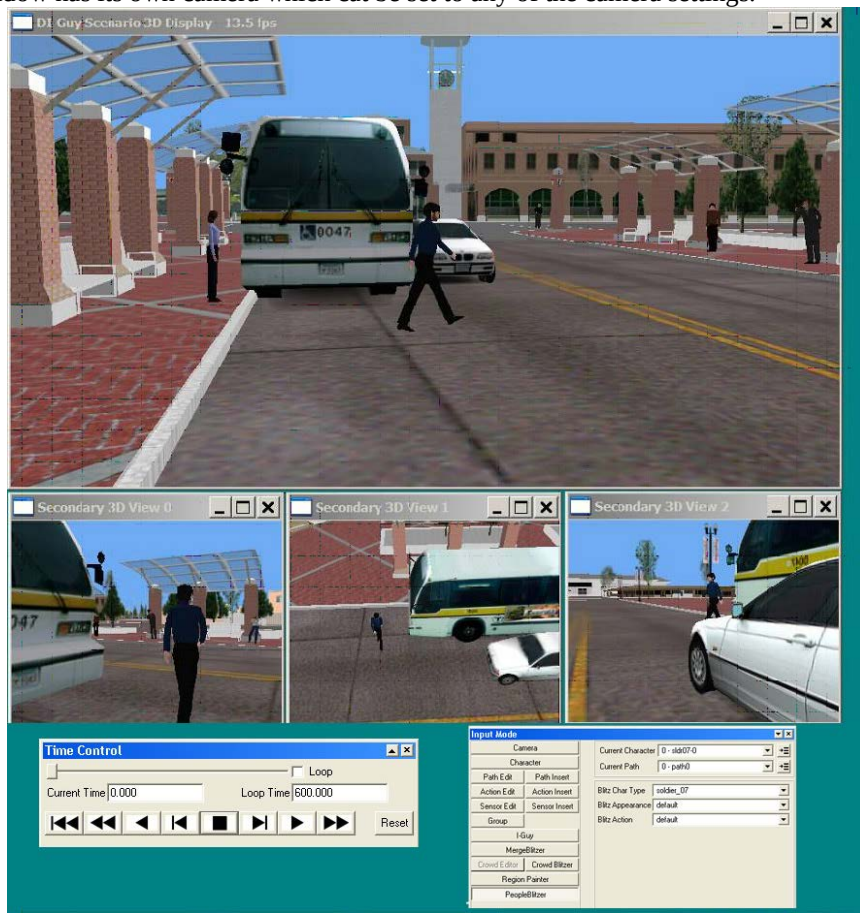
The buttons in this window determine the mode of operation when using the mouse in the DI-Guy Scenario 3D Display window. The rows determine the five major modes of operation. The columns separate edit operations, which allows the user to modify existing features, from. Insert operations, which allow the user to add new items.

Please see the DI-Guy AI manual for information on extra modes enabled by that product.

`Camera`	Use the mouse to change the current camera position and orientation
`Character`	Use the mouse to move characters or access their right-click menus
`Path Edit`	Use the mouse to select, move, and edit Waypoints
`Path Insert`	Use the mouse to create new Waypoints
`Action Edit`	Use the mouse to select and move existing Action Beads
`Action Insert`	Use the mouse to create new Action Beads
`Sensor Edit`	Use the mouse to select and move an existing Sensor Region
`Sensor Insert`	Use the mouse to create a new Sensor Region
`PeopleBlitzer`	Use the mouse to create new people that have paths and actions
`Plugin*`	If you are using a plugin that operates on mouse input, this mode will also be available.

4.4 Using Multiple 3D Windows

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



To create a secondary window, simply activate it from the Main Window using the Windows pulldown. At the bottom of the pulldown menu are Secondary 3D Window 0 through 8. Selecting any window from the menu will immediately instantiate the window. Once a secondary 3D window is selected by clicking on it, the user can adjust the view shown in it using the mouse, camera hot-keys, or via decision logic.

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

5. Scenario Objects Palette

The Scenario Objects palette is a centralized palette where all advanced authoring capabilities reside. There is a tremendous amount of functionality contained within this user interface. It is divided 6 main sub-palettes:

1. Character Objects (work with characters individually)
2. Scenario Objects (a first tier of Scenario Object features)
3. View Settings (camera, fog, light, and visibility)
4. Event Handlers (scenario level decisions and scripts, event mapping)
5. DI-Guy AI Objects (requires purchase of DI-Guy AI module – see that manual for specifics)
6. Specialized Objects (a second tier of Scenario Object features)

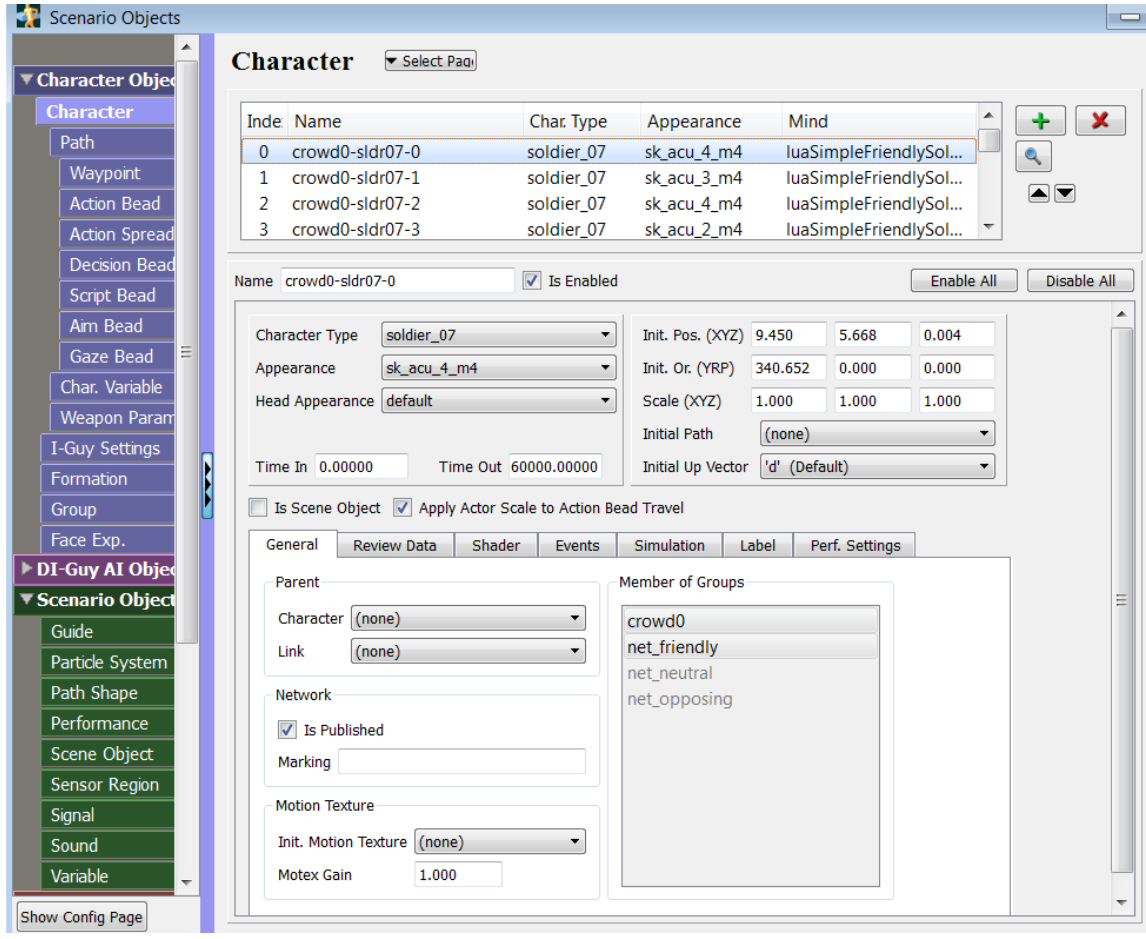
5.1 Character Objects Tab

The Character Objects palette is a multi-tab palette that provides you access to various character-centric parameters.

5.1.1 Character Objects -> Character Tab

The Character tab is the first tab of the Character Elements palette.

It starts with a simple list of all the characters in the scenario. Highlighting a character in the list immediately reveals basic information about that character. The other tabs in the Character tab will also reflect information about the selected character.



The character tab with character #0 highlighted

Creating a Character

Press the Create button.

A new character is added to the scenario. The new character is given an index one higher than the currently selected character. The indices for all people following the added character will be increased by one. The new character becomes the current character.

Deleting a Character

You can remove a character from the scenario:

1) Select the character to be deleted.
2) Press the Delete button.

3) The indices of all the following characters will be decreased by one.

Character Name Field

You can give the character a name. Enter the name in the Name field. Do not use the same name twice.

Character Type

Every character in DI-Guy has a Character Type that defines the actions and appearances available for that character.

Appearance

Characters of a particular Character Type share the same link/joint hierarchy and can perform the same set of actions. But these characters can have a wide range of visuals. The Appearance field determines the appearance of the character, 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. See Chapter 8 for images of several appearances.

Head Appearance

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

Initial Path

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

Initial Position and Orientation

The starting position and orientation of the character when the scenario starts.

Scale

Characters can have their size scaled in height, width and/or depth. Motions will be scaled accordingly.

Time In/Time Out

Characters can have a time in and time out independent of the scenario time in and time out. This can be useful for effects.

Is Enabled

Disabled characters do not appear in, nor influence, a scenario.

Is Scene Object

Used mostly for props, a character that is a scene object will act as part of the terrain, both in terms of collision and altitude functions for AI characters.

Apply Actor Scale to Action Bead Travel

Whether changes in scale affect distance traveled. Defaults to on.

All Character Enable and Disable buttons (below the Advanced Tabs section)

All the characters in the scenario can be enabled or disabled with one button press.

ADVANCED TABS:

General

Network Settings

The *Is Published* check box determines whether this character should be broadcast when DI-Guy Scenario is networked using HLA or DIS.

Parent

Parent the current character to another character.

Review Data

Review Data determines whether and how data should be saved for this character for future playback. *Siz***e** indicates the initial amount of data to be allocated for this purpose. *Increment* is the size of subsequent chunks of data to allocate if the initial amount of data from Size is surpassed. For characters, review data encompasses the state of all joints in the body at every dt.

Group

Group shows which group(s) the selected character belongs to. Groups are simply a collection of one or more characters. The networking module has three pre-defined groups, “net_friendly”, “net_neutral”, and “net_opposing”. Groups can be used to classify characters, vehicles and props to facilitate applying decision logic to classes of entities, rather than to individual entities. Note that this sub-tab works in

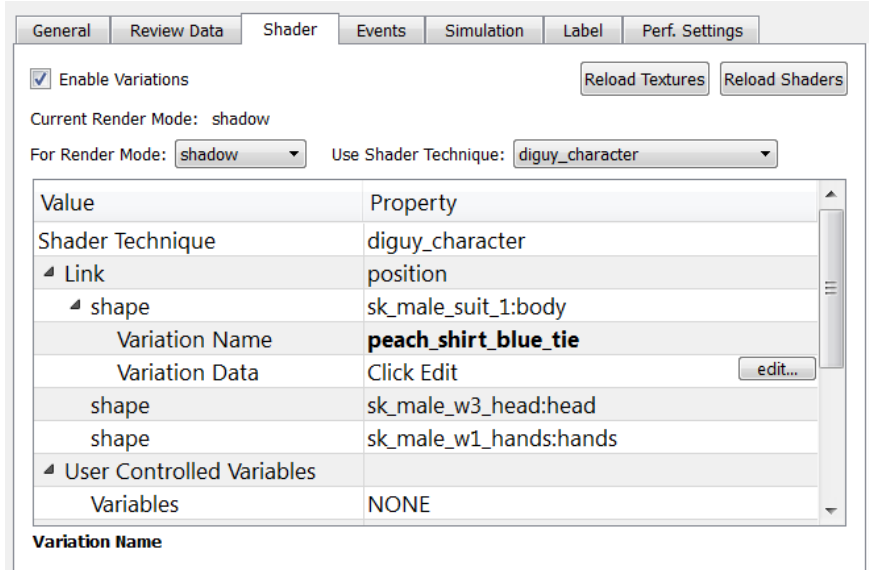
conjunction with the Group tab found just after Formation in the Character Objects. That tab is devoted to managing groups.

Shader and Variations

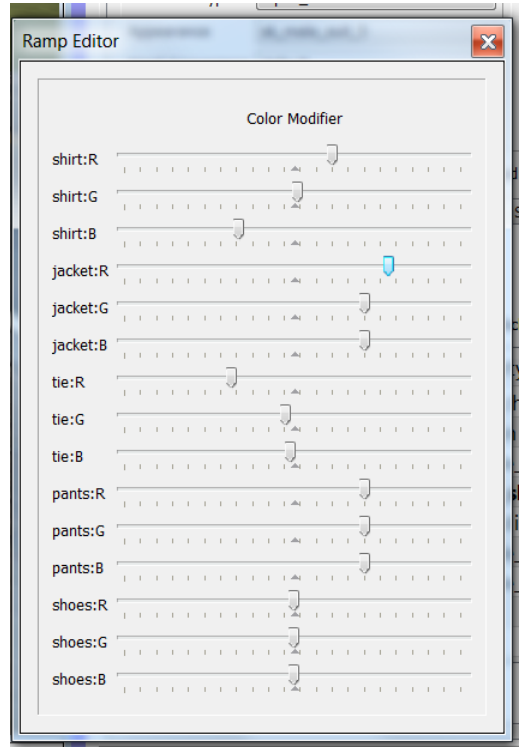
Specify the shader be applied to the character for rendering (Advanced Users only). Many 12.5 Characters and later characters are set up to for our variation system where different parts of the character can be adjusted in real-time via GPU based shader.

Three sk_male_suit_5 characters with different variations

To activate the variation system select “Enable Variations” in the shader UI. If the character supports variations one will be randomly chosen. And the end user should see a bold list of options that the end user can pick.



The Variation Name combo provides a list of base lines. Clicking Variation Data edit… will bring up an editing UI which allows you change the various pieces of the character in real-time.



Variation changes will save to disk as part of the character's description as well as flow over the network via a custom PDU.

Events

Map event handlers to callback events.

Simulation

By default, the character performance is controlled by the DI-Guy Motion Engine. However, some characters may use ragdoll physics, chain simulator, etc.

Label

A text label that will show up in the 3D Window can be added to the character . By default, the label is the name of the character. The labels have been recently enhanced with many features for displaying text.

Note: be sure to enable character labels in the Visibility tab of View Settings.

Performance Settings

If not using the load manager, similar parameters can be set here. Consult the Load Manager tab information of Scenario Objects for more information.

Motion Texture

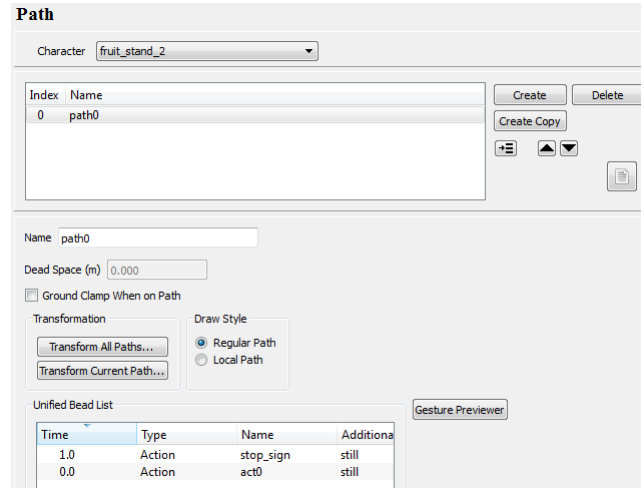
DI-Guy Scenario allows you to specify a Motion Texture that is added onto whatever motion is generated by the action. A typical Motion Texture varies each of the joints in a characteristic way to provide richness and randomness to the behavior. The *Initial Motion Texture* field specifies the motion 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.

Interactive Guy (I-Guy)

If you want to control your character using a joystick, check the I-Guy box. *Collisions* will stop DI-Guy from walking though walls, etc.

5.1.2 Character Objects - Path Tab

In DI-Guy Scenario, 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.



See the Tutorial section of this manual to learn more about paths and waypoints, and about how to edit them directly in DI-Guy Scenario's 3D Window.

Use this tab to create or select a path, name it, and then the following tabs to work with the waypoints and beads on the path.

Visual Representation

Paths are represented in the 3D Display Window by thin, curved lines. The current path is white; all other paths are blue.

Select a Path

Select by clicking a path from the list of available paths on the Path tab to do path editing.

Creating a Path

Press the Create button.

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.

Deleting a Path

1) Make the path to be deleted the current path.
2) Press the Delete button.

Indices of paths following the deleted path will be decreased by one.

Path Name edit box

You can assign a name to a path by typing the name in the path name edit box. Be careful not to use the same name twice.

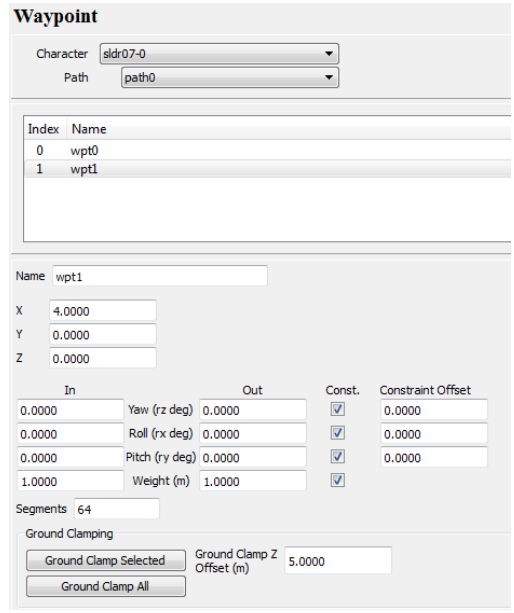
Dead Space edit box (Advanced Users)

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.

Gesture Previewer Button

When the gesture previewer button is selected, the 3D Window will move to 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.

5.1.3 Character Objects - Waypoint Tab



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.

See the Tutorial section of this manual to learn more about waypoints and paths, and about how to edit them directly in DI-Guy Scenario's 3D Window.

Individual waypoints can be selected from the list by clicking on them

Visual Representation

Waypoints are represented in the 3D Display window by green tick marks. The tangent lines of the current waypoint are red; all other tangent lines are green.

Each waypoint has a slope-adjuster which 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.

Creating Waypoints using the 3D Display Window

It is easiest to create new waypoints in the 3D Display window. New waypoints are added to a path right after the currently selected waypoint. If you have selected waypoint 2 and insert a new waypoint, then the new one will be waypoint 3 and the index of all subsequent waypoints will be increased by one. To insert a waypoint:

1) Use the Input Mode palette to select Path Edit mode (or alternatively use the 'x' hotkey).

2) Select one of the existing waypoints. (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.

3) Use the Input Mode palette to select Path Insert mode (or alternatively use the CTRL hotkey to active Path Insert mode from Path Edit mode).

4) Point in the 3D Display window directly where you want to the waypoint to go. Press the left mouse button. Without releasing the left mouse button, drag the cursor to determine the orientation and length of the slope-adjuster, which in turn determines the orientation and smoothness of the path. Release.

Repeat these steps to create additional waypoints. Each new waypoint will be added after the last, and will automatically become the current waypoint.

Creating Waypoints using the Character Elements Waypoints Tab

You can also create waypoints through the Waypoint tab of the Elements palette.

1) Set the current waypoint index to one before the desired index of the new waypoint. For example, to create a waypoint between existing points 2 and 3, set the current waypoint index to 2.

2) Press the Create button.

The default position of the new waypoint created in this way depends on whether it will be the first waypoint (index 1), the last waypoint, or somewhere in between.

`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.

Deleting Waypoints

1) Select the waypoint to be deleted.

2) Press the Delete button.

Indices of following waypoints will be decremented.

Editing Waypoints using the 3D Display Window

It is easiest to move waypoints in the Display window.

1) Use the Mode palette to select Path Edit mode (or alternatively use the ‘x’ hotkey).

2) Click on a waypoint on the current path to select it. The selected waypoint’s slope-adjuster will turn red.

3) Click and drag on the waypoint to change its position.

4) Click and drag on the slope-adjuster to change the orientation of the waypoint. The slope-adjuster is sensitive to the mouse at its endpoints.

Editing Waypoints using the Character Elements Waypoints Tab

Waypoint Name edit box

You can assign a name to each waypoint. Be careful not to use the same name twice.

X Y Z edit boxes

Although it is generally easier to edit waypoints by clicking and dragging them in the 3D Window, sometimes you may want to type values that define parameters of waypoints. You can edit the position of a waypoint in world coordinates using the edit boxes on the Waypoint tab of the Elements palette.

Yaw, Roll, Pitch and Weight edit boxes

Just as you were able to type values to edit the position of a waypoint, you can type in values to edit the orientation of a waypoint.

`Yaw`	Rotation around the z (up) axis
`Roll`	Rotation around the x (forward) axis
`Pitch`	Rotation around the y (side) axis

Each waypoint has a weight that determines how much influence the waypoint will have on the shape of the path. You can edit the weight of a waypoint by typing a value in this field.

The orientations and weight can have separate values for entering and leaving the waypoint. If these values differ, the path will be discontinuous at the waypoint. If these values are the same, the path will stay smooth through the waypoint.

Waypoints are created with the smooth checkboxes checked. If you want to vary the In and Out values separately for any of the orientations or the weight, deselect the corresponding checkboxes.

Moving Several Waypoints as a Group in the 3D Window

It is possible to move several waypoints at one time while working in the 3D Display window:

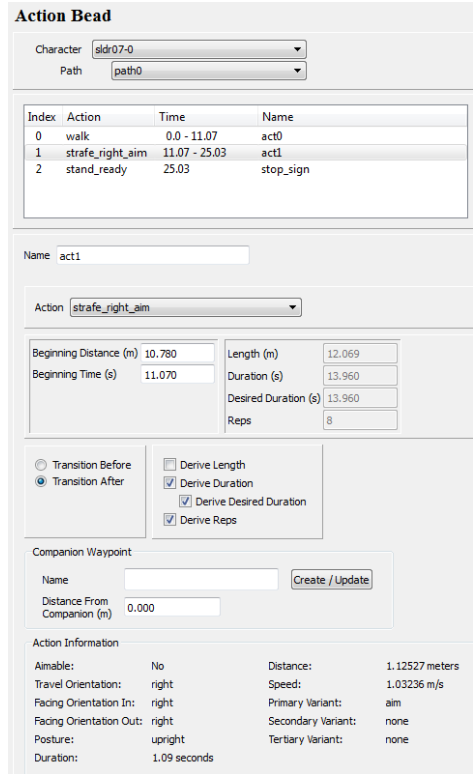
1) Use the Mode palette to enter Path Edit mode.
2) In the 3D Display window click on the first waypoint to be moved.
3) Shift-click the last waypoint to be moved. This will select all waypoints between the first and last waypoints. All selected waypoint slope-adjusters will turn red to indicate selection of the waypoint.
4) Continue to hold the shift key and drag the mouse to move the group of waypoints.

Note: When dragging a collection 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 of the group 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.

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*. At the bottom of the Waypoint tab of the Character Elements palette there are buttons that allow you to ground clamp the selected waypoints on the selected path, or to ground clamp all of them. In some cases you may want to adjust the altitude of the ground clamping algorithm. The *Ground Clamp Z Offset* field is provided for this purpose.

5.1.4 Character Objects - Action Bead Tab



Each Action Bead along a path tells the character to perform a new action. 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. However, if you wish to get lower-level access to the action beads you can do so through this tab.

To edit an action bead for the currently selected character or vehicle, click on the Action Bead tab in the Character Elements palette. Each action bead has an index. The first action bead has index 0, the second index 1, etc.

Note: The Action tab of the Elements Palette provides access to the action beads of the current path. If there is no current path, you can not create, modify, or delete action beads.

Visual Representation

Action Beads are represented in the 3D Display window by stationary yellow gear-shaped objects that strung on the path. The current action bead is an exception; it isn't stationary, it spins on its axis. The last action bead is also an exception; it is a red octagonal symbol, similar to a stop sign.

Creating Action Beads

Set the current action bead index to one before the desired index of the new action bead. For example, to create a action bead between existing beads 2 and 3, set the current action bead index to 2. Then, press the Create button.

Upon completion, the new action bead will become the current one. The indices of the subsequent action beads will be incremented to make room for the new 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. This distinction is important when inserting and editing the actions of a character.

Action Bead Name Edit Box

You can assign names to Action Beads to help distinguish among them.

Action

The name of the action associated with the action bead.

Beginning Distance

Indicates the distance into the path that this action starts.

Beginning Time

Indicates the time this action starts.

Length

Distance that is covered by this action.

Note: Editing length or duration for an action bead will cause a ripple effect on all subsequent action beads in the path. See the Action Spreadsheet for more information on this effect.

Duration

Amount of time required to perform this action

Desired Duration

Amount of time desired to perform this action.

Reps

Every action has a natural distance and time associated with it. Each time an action is specified, the DI-Guy 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 will stretch or shrink the actions to make it fit exactly. See the Tutorial section to learn more about the various types of actions and how DI-Guy Scenario calculates displacements and times.

Transition Before/Transition After Toggle Buttons

When one action is complete and another action is starting, a transition motion will occur to smoothly and naturally have the character change his motion. DI-Guy automatically performs this transition for you. The distance or duration of these transitions are calculated when determining reps, 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 shouldn't need to modify these toggles.

Derive Length Check Box

When this box is checked, DI-Guy Scenario derives the displacement for the action based on the time you specify or the number of repetitions you specify.

Derive Time Check Box

When this box is checked, 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

When this box is checked, the desired time is set equal to the actual duration of the action. When this box is unchecked, 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.

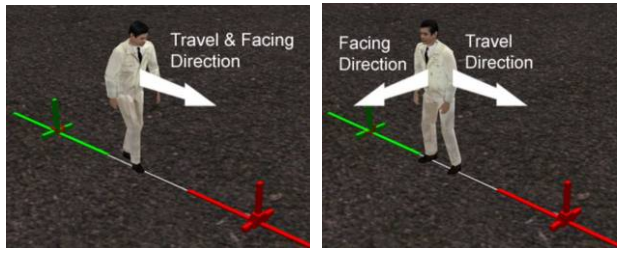
Derive Reps Check Box

When this box is checked, DI-Guy Scenario automatically calculates the number of times the action is looped to best fill in the available space or time.

Non-Forward Actions and Companion Waypoints

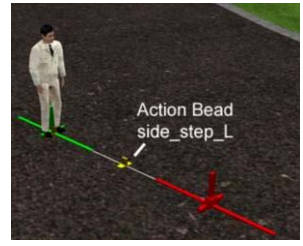
People generally face in the same direction they travel. For instance, a person walking forward faces forward as they go, keeping their face, chest, and pelvis aligned with their forward motion. Similar rules generally apply for strolling, jogging and running. These are all called *forward actions*.

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.

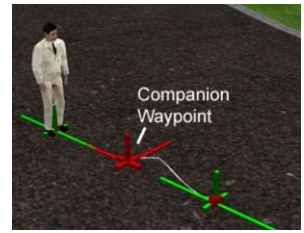
This is all easier to see and understand if you do an example using the DI-Guy Scenario tool. Perform the following steps:



Step 1: Blitz a character of Character Type *controller* into the scene. The controller is one of many characters in DI-Guy that has non-forward traveling actions.

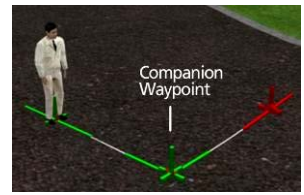
Step 2: Select action bead *side_step_L* and insert it at the center of the path using *action insert*.

Step 3: On the Action Bead tab of the Character Elements palette, select the side_step_L action. Then find the box labeled *Companion Waypoint*, near the bottom of the tab. Click the Create/Update button. Observe that a new waypoint is located 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: hence its name *companion* waypoint.



Step 4: Adjust the waypoint placement and path shape as desired. Press reset, then play to observe the behavior of the character.

Step 5: If you want the character to resume forward travel, insert an action bead for forward travel later on the path and create an additional companion waypoint for it using steps 3 and 4. Press reset, then play to observe the resulting behavior.

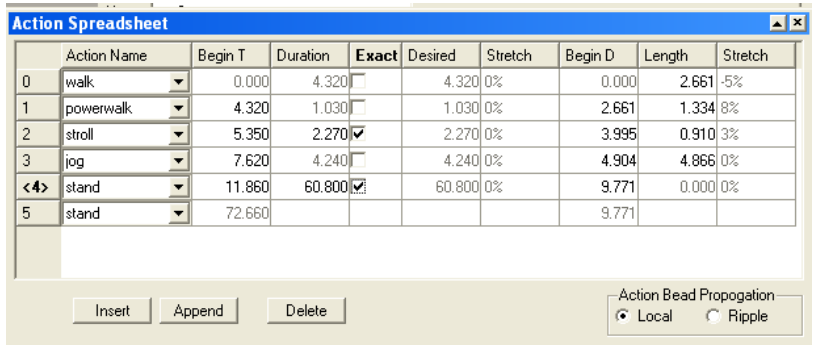


You may notice that the companion waypoint created looks slightly different than a normal waypoint. Its slope adjuster bar (the long post) no longer passes straight through the waypoint, but instead is angled (in our example, 90 degrees) to reflect the desired change in input and output facing directions.

After completing the above example, 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. Go to the Action Bead tab of the Character Elements palette and again press the Create/Update button in the Companion Waypoint box near the bottom of the tab. The waypoint will be properly updated, taking into account the change in reorientation that will occur at the action bead. You will probably want to move the final waypoint for a completely satisfying overall motion.

5.1.5 Character Objects - Action Spreadsheet Tab

The Action Spreadsheet palette 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 typed in directly. This palette gives access to timing information for the current character, in addition to 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 using the PeopleBlitzer, Path, and Action Input Modes in the 3D window to establish a character's basic behavior.



Each row gives the data for one action. The column definitions are as follows:

- **Action Name** – name of the action
- **Begin T** – The absolute time this action begins
- **Duration** – Amount of time the action requires. Note that the user sets this for stationary actions and when using Exact.
- **Desired (Duration)** – Amount of normally required to perform the action (or repeated loops of the action)
- **Stretch (Duration)** – Amount motion time was dilated to achieve time when action is duration-constrained.
- **Begin D (Displacement)** – Start distance from beginning of path for this action.
- **Length** – Distance to travel. Should be zero for stationary actions.
- **Stretch** –Amount the motion was compressed or elongated in travel distance to achieve the length constraint.
- **Exact** – For advanced users, allows user to constrain motion both in displacement and duration.

Action Bead Propagation

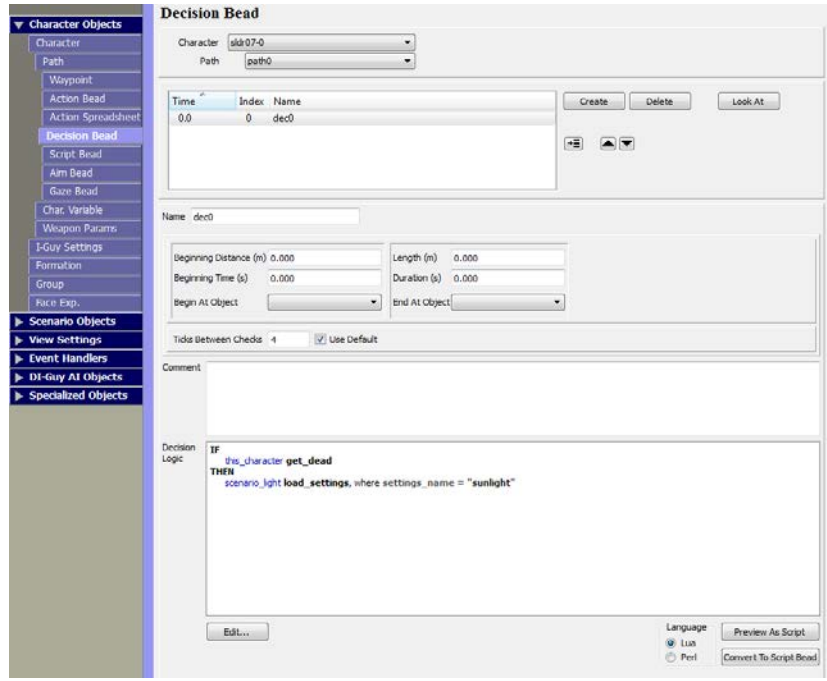
When an action is inserted into the middle of a sequence of actions, (or when a length or duration constraint is changed), the effect of that action on subsequent actions is governed by the setting of these toggle buttons. In local mode, the original position and timing of the subsequent actions are effected as little as possible. In ripple mode, the subsequent actions are pushed out by the amount of duration or displacement inserted.

5.1.6 Character Objects - Decision Bead Tab

Decision Beads are an easy way to make DI-Guy Scenario characters react to events in the scenario that requires no programming.

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 Decision Bead Tab of the Character Elements palette is designed to lead you through specification of the decision logic, so you do not have to remember the operators or do any programming.



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

Visual Representation

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

Decision Bead basics

Similar to all the other tabs on the Character Elements palette, all the decision beads on the current path are listed in a box in the top left corner of the tab and are indexed starting at zero. You can create or delete decision beads.

There is a “look at” button that will have the camera look at the current Decision Bead.

The decision bead can be given a name.

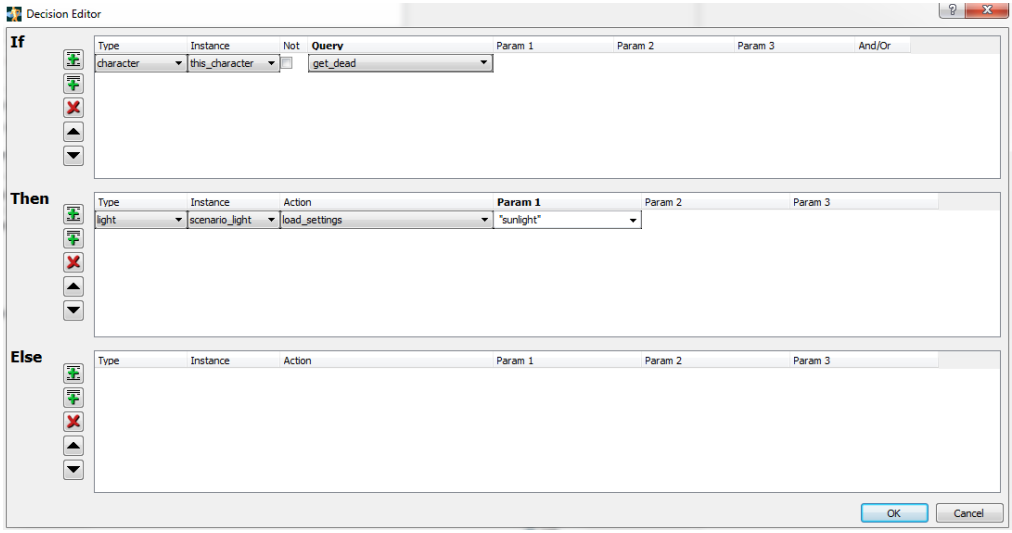
Beginning Distance, Beginning Time, Length, and Duration specify where, when, and for how long a decision bead remains active.

Note: A common error is to not modify the defeault zero values for both length and duration. Unless you only want your conditions to be checked once, at time zero, it is important to not neglect these critical values.

Ticks between checks indicate how frequently the condition should be checked; sometimes it is a good idea for performance reasons not to check for a condition every frame.

Creating Decision Logic

After creating a decision bead, click the Edit button. This brings up the Decision Editor.



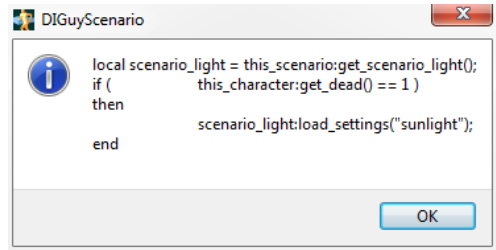
In this decision, if the character is dead, then the light is changed to sunlight. Undoubtedly, the character must be a villainous deity of some sort!

The Decision Editor is designed to be EXCEEDINGLY EASY to use. All Decisions are put in an "If…Then…Else" logic. But it is often the case that you will use even easier subsets. For example, if you know you want to activate a particular camera at time t=10, simply set the decision bead to have a beginning time of 10.0. Then in the logic, say "THEN CAMERA…etc" Not only are you not using the ELSE, you are not even using the IF!

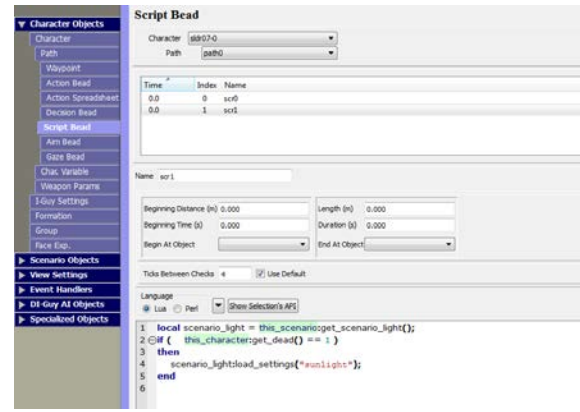
The best way to learn how to use decision beads is to play with them. Experiment first with different "Then statements", then explore the "If" section. The decision palette uses simple pulldown boxes that require no typing. Essentially, function calls are made from this interface to the DI-Guy API, so the boxes refer to the objects, functions, and arguments that those functions required. At each stage of use, the Decision palette dynamically shows the available objects, operators and arguments.

Converting Decision Beads to Script Beads

Converting Decision Beads to Script Beads is a great way to get a head start on writing a Lua script on those occasions where the decision logic and choices don't offer exactly what you are looking for. Script Beads are described in the next section. To automatically convert a decision bead into a script bead, click on the *Convert to Script Bead* button located on the Decision Bead tab. You can then select the Script Bead tab of the Elements palette to edit and create arbitrarily complex decisions and behavior.



5.1.7 Character Objects - Script Bead Tab



The Script Bead tab behaves exactly the same as Decision Bead tab except that instead of using the Decision Editor to generate the logic, a script is used.
DI-Guy supports scripting via the Lua scripting language. The vast majority of the DI-Guy API can be directly accessed inside of scripts. You can get at all the other elements in the scenario, including the other characters and Scenario Objects.

Converting a Decision Bead into a Script Bead (see previous sections) is a really easy way to get started on your road to becoming a DI-Guy Scenario Lua Scripting Expert. Don't be shy, give it a try!

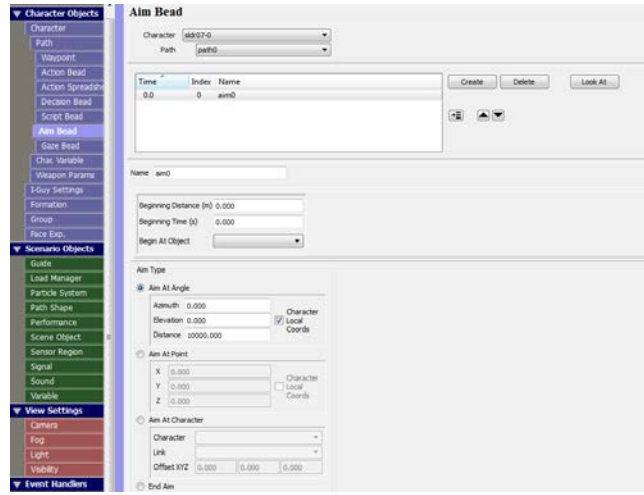
A majority of the DI-Guy API (over 250 functions) is available through the scripting interface. You can find descriptions of these functions in the documentation that is installed when you install DI-Guy Scenario on your computer. Look in: $DIGUY\doc\diguy_sdk_reference.pdf.

A good reference on the Lua scripting language is Programming in Lua (http://www.lua.org/pil/)

5.1.8 Character Objects - Aim Bead Tab

For those actions that support aiming (action name will contain the word "aim", such as "stand_aim"), aim beads can be used to program the direction a person is aiming. The aim bead is positioned along the path using the *Beginning Distance* or *Beginning Time* field of the palette. When the person crosses the aim bead their aim changes from the previous value to the new value specified.

Aim Type



There are three modes for specifying the aim point;

1. *aim angle mode,*
2. *aim point mode,*
3. *aim character mode.*

In addition, *end aim* can also be chosen.

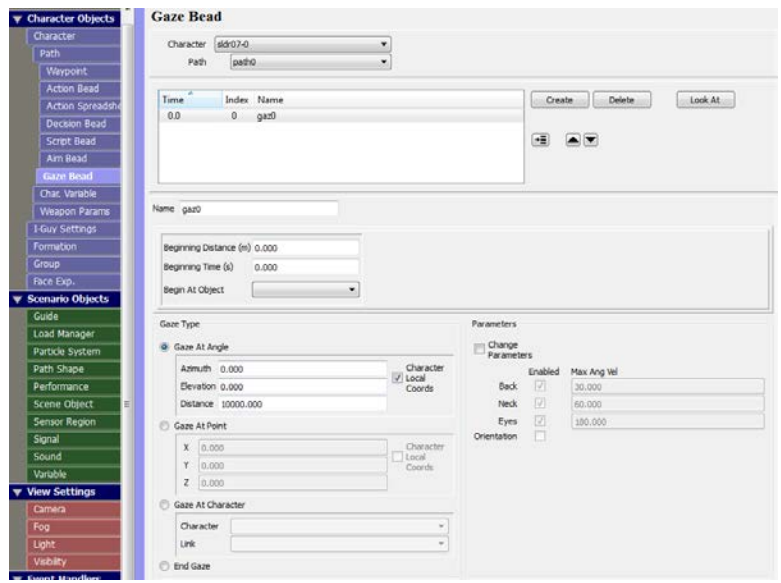
In *aim at angle* mode, the direction of aim is given as an azimuth and elevation. Note that to have a character aim up or to the right, the value is expressed as a negative value. An elevation of –45 will cause the point of aim to be upwards at a 45 degree angle. The heading and elevation can be expressed in global or local coordinates.

In *aim at point* mode, the character aims at the point specified. The point can be expressed in global or local coordinates.

In *aim at character* mode, the character aims at one of the persons, vehicles, or props in the scenario.

The aim bead will persist until a new aim bead or an *end aim* bead is encountered.

5.1.9 Character Objects - Gaze Bead Tab



Gaze refers to the direction a character is looking. You can specify the gaze for a character using Gaze Beads. The gaze bead is positioned along the path using the *Beginning Distance* or *Beginning Time* field of the Gaze Bead palette. When the character reaches the gaze bead their gaze changes from the previous value to the new value specified.

There are three modes for specifying the fixation point for gaze:

1. *gaze angle mode,*

2. *gaze point mode,*
3. *gaze character mode*

In addition, an *end gaze* bead can be specified.

In *Gaze at Angle* mode, the direction of gaze is given as an azimuth and elevation with respect to straight ahead. Not that to have a character gaze up or to the right, the value is expressed as a negative value. An elevation of –45 will cause the point of gaze to be upwards at a 45 degree angle. The heading and elevation can be expressed in global or local coordinates.

In *gaze at point* mode, the character gazes at the specified point in space. The point can be expressed in global or local coordinates.

In *gaze at character* mode, the character gazes at the specified person, vehicle, or prop in the scenario.

The gaze bead will persist until a new gaze bead or an *end gaze* bead is encountered. You can specify an end gaze bead using this tab of the Character Elements Palette.

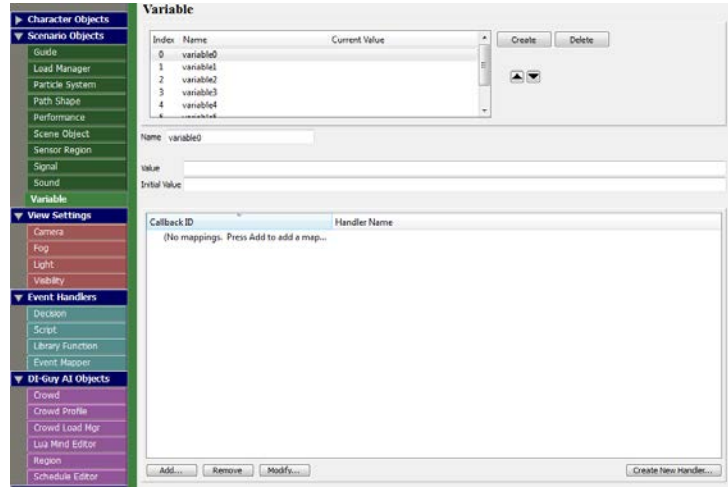
The Parameters section lets users specify how the character will turn the various parts of its body to achieve the gaze goal.

5.1.10 Character Objects - Variable Tab

Users can define variables that are associated with the character on this tab. A list of variables is displayed in the upper left corner, with create and delete buttons available for managing the list.

Each variable should be given a descriptive name and an appropriate initial value.

There is a wealth of API Functions available for dealing with character variables. They all have the form diguyCharacter->variable_<etc>. Note that the variables also appear as fields in the decision tab. Variables are useful when combined with decision and script logic.



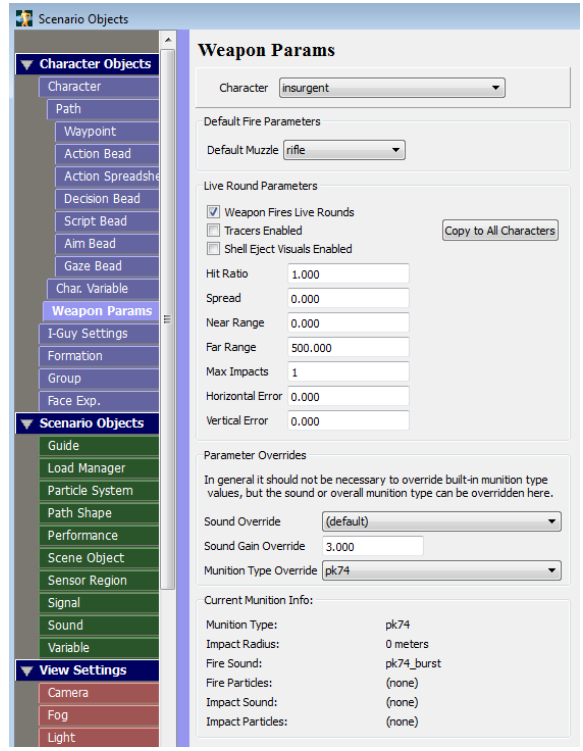
5.1.11 Character Objects - Weapon Parameters Tab

Many DI-Guy characters can use weapons. This section governs how those weapons work when fired.

A Default Fire Parameter defines the basic type of weapon (e.g. – rifle, grenade, etc)

The Live Rounds Parameters defines how the weapon fires. Does it use tracers? Is is a "live round", capable of intersecting another entity? If so, how good a shot is this character?

The Parameter Overrides section (Advanced Users) overrides sound and munition information. Note that munition information is often communicated out of DI-Guy Scenario to other simulations in a distributed simulation environment using DI-Guy Scenario DIS/HLA networking.



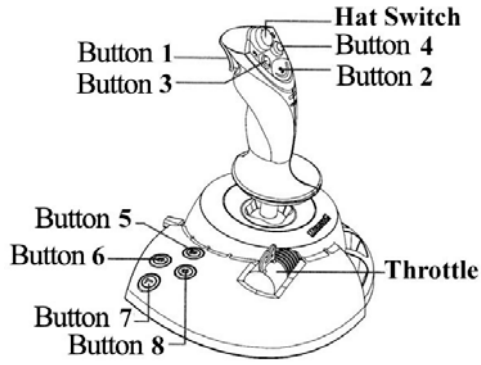
5.1.12 Character Objects - I-Guy Settings Tab (Using a Joystick or Keyboard for 1st person/3rd person control of character)

I-Guy Mode

Note that the Input Mode palette has a button called I-Guy Mode. The I-Guy character will be under joystick or keyboard/mouse control when that button is selected and the 3D Window has the focus. Hit the ESC key to return to normal DI-Guy Scenario Operation

Joysticks

DI-Guy Scenario allows you to use a joystick to control the behavior of any character. DI-Guy Scenario



The Microsoft® Sidewinder Precision 2 USB joystick was used to develop and test joystick functionality. The button numbering shown in the figure corresponds to the functionality given in the text.

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.

To place a DI-Guy Scenario character under joystick control, select the character on the I-Guy Settings tab. Then make sure the play button is active on the Control Palette.

The joystick can control two kinds of actions:

- Body actions (walking, jogging, running, backing up, etc.)
- 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.

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. Turn on collision detection by selecting the I-Guy character, then checking the *Collisions* checkbox on the Character tab of the Elements palette.

Body Action

- 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.
- Pressing button 3 toggles between first-person point of view and third-person point of view. You can adjust each of the views using the camera controls.
- Pressing button 7 moves character further upright. If he is crawling, he will crouch. If he is crouching, he will stand upright. (Not all characters have all of these behaviors.)

- Pressing button 8 moves character closer to the ground. If she is standing, she will crouch. If she is crouching, she will crawl. (Not all characters have all of these behaviors.)
- Pressing button 6 causes the character to respond more quickly to action requests, at the expense of smoothness. There are three responsiveness levels.
- Pressing button 5 causes the character to respond more smoothly to action requests, at the expense of responsiveness. There are three responsiveness levels.
- Press button 4 to toggle *sticky view* mode.

Head Action (third person point of view)

- Press Button 3 to toggle between first-person point of view (POV) and third-person point of view.
- Pressing button 2 toggles *aim* mode. Entering aim mode stops the motion of the character, and he aims the weapon. In this mode, the hat switch controls the direction in which the weapon is aimed. Only operative for characters with a weapon.
- While in aim mode, button 1 fires the weapon, generating muzzle flash and playing sound "gunfire". For sound to be heard, a sound element named "gunfire" must be created.

Head Action (first-person point of view)

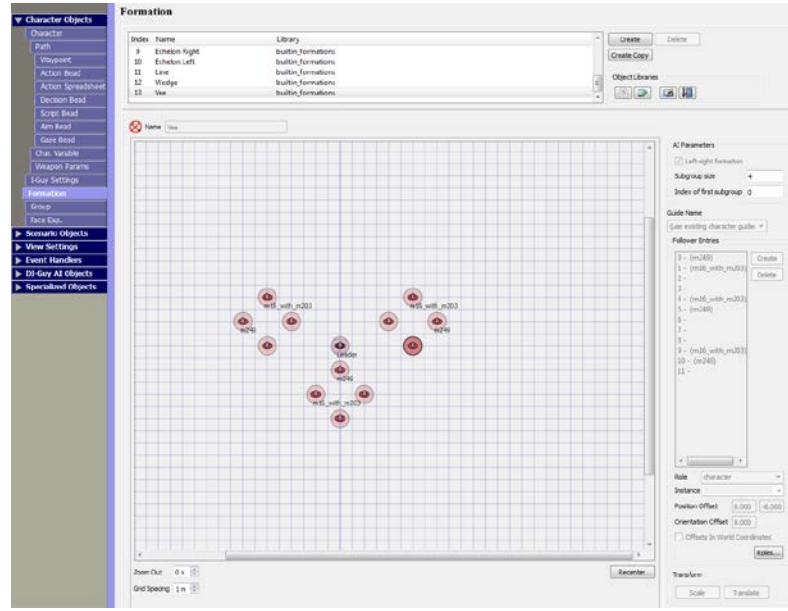
- Press button 3 to toggle the point of view between first-person and third-person. Once you are in either of these modes, you can adjust the details of the view using the *look-from* and *look-at* fields on the camera palette.
- The hat-switch moves the head and your view up and down.
- Button 4 toggles *sticky view* mode.
- Button 2 toggles between aim mode and ready mode. In aim mode, the hat-switch aims the weapon.
- Button 1 fires the weapon when in aim mode.

Keyboard and Mouse Mode

The I-Guy can be controlled using the W-A-S-D and other keys to control movement and posture, with the mouse dictating the gaze and orientation.

5.1.13 Character Objects - Formation Tab

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.



The Formation Tab of the Scenario Objects Palette allows you to define one or more formations for use in the 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.

To use the Formation Tab, perform the following steps:

1. Create a new formation using the create button near the top of the tab.
2. Name the formation by typing a new name in the name field.
3. Adjust the scale of the grid by specifying the spacing of the rows and columns using the *Grid Row Spacing* and *Grid Col Spacing* fields located just below the grid.
4. Create followers. Decide how many followers you need. In the *Follower Entry* box at the right of the tab, create the number of followers you need (use the *create* button).
5. Position the followers. In the grid at the left, first select a follower by clicking on it, and then click again at the desired empty location to position the follower. You can reposition the followers again later, if you wish. Create characters that will follow. These can be existing characters in the scenario, or ones you create for this purpose. You can create the followers by PeopleBlitzing them into the scene or using the create function on the Character Tab of the Character Elements palette.

6. Assign follower characters to each of the follower locations. Click in the grid to select a follower location. Then use the *Instance* field at the right to select which character is assigned to that location. Assign characters to all of the follower locations. (Ignore the *Class* field.)
7. Select a Guide using the *Guide Name* pull-down menu. Guides determine the travel control algorithm used by each follower as it travels within the formation. Guides are defined on the Guide Tab of the Scenario Objects palette, described next in this section.

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.

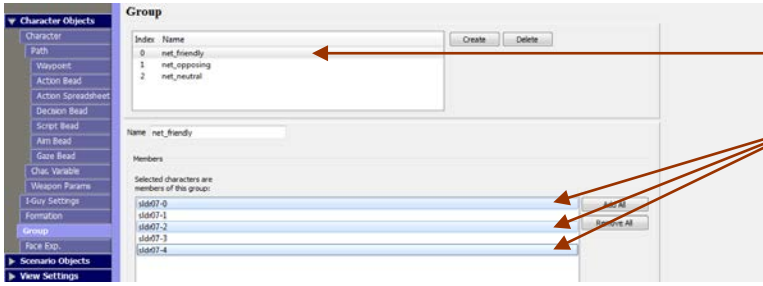
Subgroups and Roles:

If you specify a subgroup size other than zero, then the formation can contain subunits. Set the "index of first subgroup" field to 1 (or 2, 3…) if you'd like the formation to have a leader (or 2 or 3…) who isn't in a subgroup. Once characters enter a formation, those in a particular subgroup will stick together no matter how the formation reorients itself.

The "role" dropdown is used to ensure that a character of a particular type will fill a particular spot in a formation (for instance, in a military application, you might always want a machine-gunner on the right flank). If it doesn't matter, simple choose "character" for the role. Note that the formation code treats roles as a strong suggestion, but will ignore them when the actual characters in a formation don't have appropriate role settings.

5.1.14 Character Objects - Group Tab

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, e.g. shoot hostiles, protect friendlies, observe neutrals. Generally Groups facilitate applying decision logic to classes of entities, rather than to individual entities.

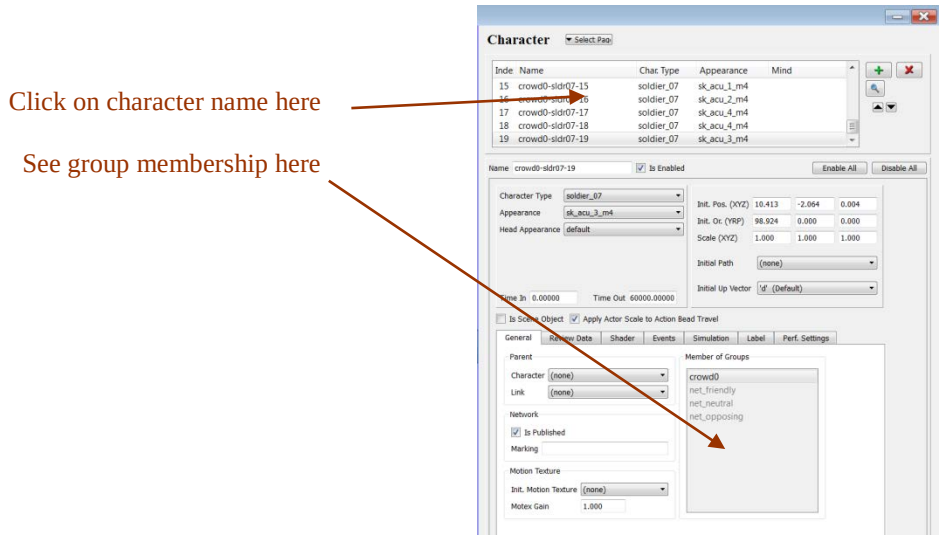


Create a new group by clicking the *Create* button. Assign your own name to the group by typing in the *Name* field. To assign a character to a group, first select the group by highlighting its name in the upper box on the Group tab of the Scenario Objects palette. Then go to the *Members* box at the bottom of the Group tab and click on the names of each character you want to include in that group. The characters

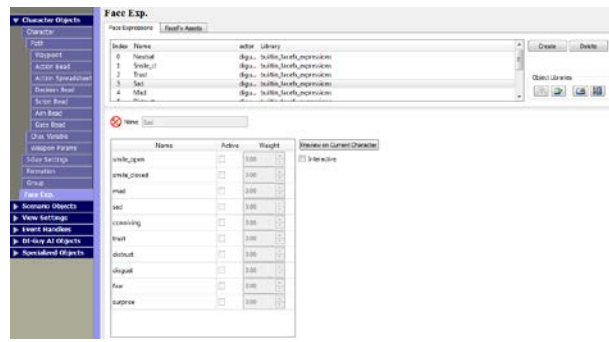
highlighted are the members of the group. You can review your selections by clicking on each of the group names above and examine the highlighted entity names below.

Once you have created groups, you can check any character's inclusion in groups on the Character tab of the Character Elements palette. Select a character and the names of all the groups it belongs to are highlighted.

Once groups are defined you can use them in decision logic and script beads to control the flow of behavior in the scenario. See the Group tab of the Scenario Objects palette to create groups.



5.1.15 Character Objects - Face Expression Tab



The Face Expressions tab of the Scenario Objects palette is for users who 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. DI-Guy version 12 adopted a new 3rdparty, FaceFX, for facial animation. Look for updates to this section of the manual.

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.

To preview a new face expression, use the *preview on current character* button located at the bottom of the tab. It applies the selected expression to the selected person.

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

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

See the Expressive Faces chapter for more information.

5.1.16 Notes on Character Labels and 3D Window Labels

Labels and the DI-Guy Interactive Machine are both methods for adding supplementary graphics to simulations in DI-Guy Scenario. This can be very useful particularly when using DI-Guy Scenario to create end self-training applications, SAF operator stations, or just adding more information and interactivity to the 3D Window.

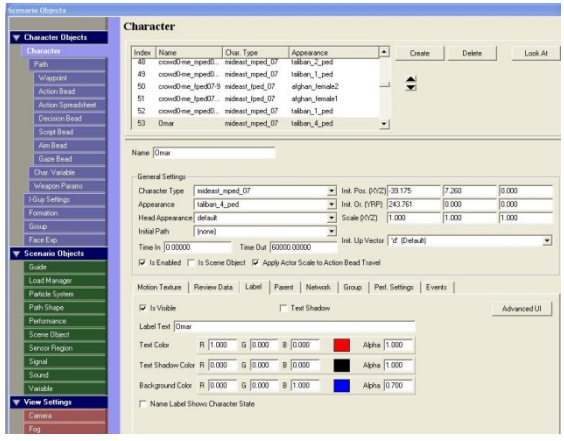
Labels:

A Label is a two-dimensional graphic that can be added to the rendered 3D scene. There are two basic types of labels available in DI-Guy Scenario:

1. Character Labels – these are the most common 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.

2. Screen / Button Labels – Screen labels stay at a fixed position on the screen. Screen labels can be "read only" or they can be made "clickable" and serve as on-screen buttons to trigger events in a scenario. Support for these labels is exclusively through the API and are therefore best implemented via scripts.

Default Character Labels



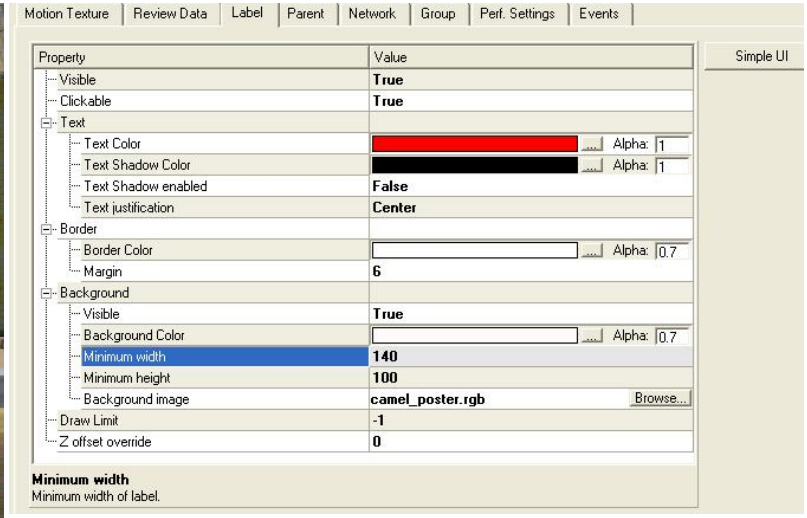
In the left picture, you can see the character “Omar” who is displaying his name in a character label. Specialized colors have been selected for his text box using the Character page ->Label tab as shown in the right image. The second character, “crowd0_me_fped07-6” displays her default name-only character label.

Overall character label display is controlled from the View Settings Visibility page, where the user can select whether none, the current character, specific characters, or all characters display labels. Specific character label display is controlled by the “Is Visible” checkbox in the Label tab of the Character page.

The “Name Label Shows Character State” option

In this image, the checkbox “Name Label Shows Character State” has been selected. This displays more detailed information about Omar.

5.1.17 Advanced Label GUI



The advanced UI lets you more explicitly control the label of a character using a DI-Guy property sheet. Instead of using text, we have set the background image to “camel_poster.rgb” and specified a minimum width and height of the image. Creative use of images includes transparent thought bubble and soldier rank graphics.

Labels and API

There are a sizeable number of functions that let users manipulate labels directly from scripts or plugins. The first place to look for information about these is in the DI-Guy SDK Reference manual, particularly the documentation regarding the diguyViewLabel class. This class lets you setup labels and all the details inside of them including:

- text
- colors
- justifications
- position
- background
- border
- bar graphs
- interactivity and callbacks

Screen and Button Labels

The sample scenario "AI_Crowd_Blitz_Demo.dss" 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.

To begin to understand this demo, go to the Event Handlers page ->Script tab and examine the script "initialize_labels", which gets executed at time 0. The script initializes the 8 labels (code for 2 shown below):

```
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 -= 30.0;
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);.
.
.
```

Most of this initialization is self-evident. The first block of code initializes the green "Click for Event:" screen label while the second block of code initializes the clickable "IED Explosion" label. Both sections construct the labels and then set their text, position, width, and colors. Note that because the second label is interactive, it calls the "set_is_clickable" function with a value of one.

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

But how is the callback script joined to the screen label? To understand that, you need to go to Event Handlers page ->Event Mapper tab. On this tab, the upper left square has a list of the types of events that can be mapped. Highlight "label". You will then see a list in the upper right box of all the labels defined in the simulation. Highlight "label_ied", the label we defined in the second block of code shown above and that we now want to hookup. You will see that 3 callback functions have been defined mapped to two callback events. (If you click on the "Add…" button, you can see a list of all the candidate callback events for a particular event type - labels have 3.) The 3 callback functions act for the label_ied act as follows:

1. pressed_red: the color of the button is changed to indicate it is depressed.

2. explosion: when the button is released, the explosion is triggered. This is the label script we were examining earlier.

3. released_red: the color of the button is reverted back to its starting color. Like pressed_red, this label script can be found in the Event Handlers->Scripts palette.

5.2 Scenario Objects Tab

The Scenario Objects palette is a multi-tab palette that provides access to various scenario features.

5.2.1 Scenario Objects - Guide Tab

Guides are an alternative means of controlling the travel of characters and vehicles, typically used for formations and incoming network entities. 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.

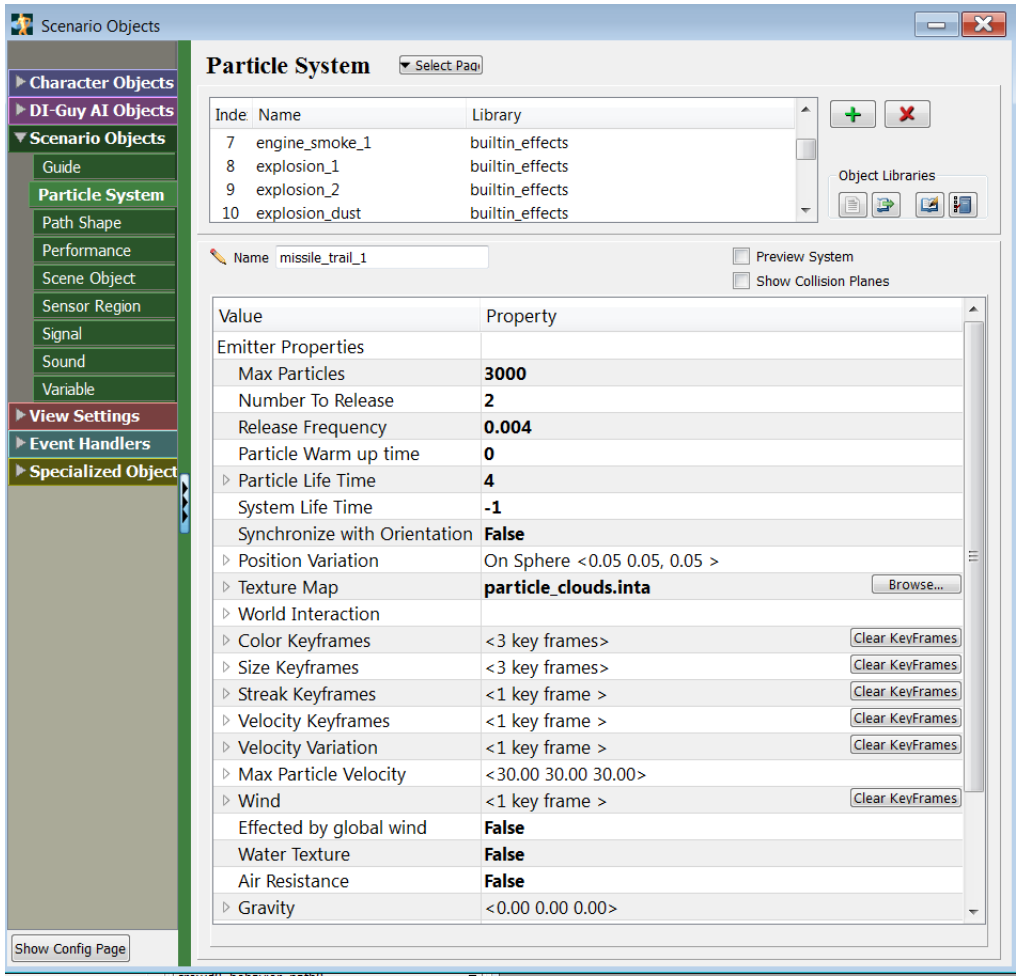
Each guide has a name, specifies a control algorithm, and a set of control parameters.

At the time of this writing, five control algorithms are defined. Additional algorithms may be added in the future.

Here is a brief description of the five control algorithms currently available:

1. Exact – Force the character to stay precisely in the desired location at all times.
2. Drift1 – This algorithm 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.
3. Follow1 – This algorithm 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 will switch to standing until the position error returns to positive. The smoothing parameters keep the character from oscillating too much in difficult following situations.
4. Follow2 – This control algorithm is just like Drift1, except that the method of reducing error is to speed the character up or slow them 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.
5. *Missile1*

5.2.2 Scenario Objects - Particle System Tab



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 15 predefined special effects: using the Particle Tab of the Scenario Objects palette 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”. The simplest way to add a special effect to the scene is to use the PeopleBlitzer. Use the Mode palette to select character type effect and the appearance you would like (e.g. fire or smoke). Then blitz the effect into the scenario.

When the scenario is paused the particles will display as a plain box. When the scenario is being played the place holder box will disappear and particles will emit from the base location of character. As long as the

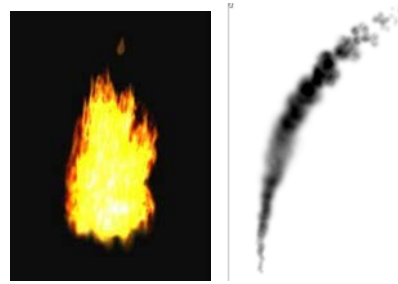
character is visible the particles will be emitted, when the object disappears it will stop emitting new particles but the particles already emitted will still be visible in the scenario.

The default set of special effects is defined in the configuration file $DIGUY/config/diguy/particles.cfg. If you have an old scenario that does not include the effects definitions, but you'd now like to add them, you can use the *Load Descriptions* button to add them to your scenario.

Special effects can be previewed by checking *Preview System*.

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

Show Placeholders indicates whether the blue cubes will be displayed or not when the simulation is paused.



Snapshots of predefined fire and smoke special effects.

A large set of parameters determines the look and behavior of the particle systems. They are described in the following section.

Particle Effects Parameters

The particle effects module is very powerful because it can be parameterized so many different ways. Note that each parameter has a small description written at the bottom of the page to help you remember what the parameter does.

Particle Parameters

Max Particles: The maximum number of particles that this system will create; once the max is reached, the system will wait 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. (Currently bounded by update rate.)

Particle Life Time: How long in seconds each particle lives.

System Life Time: How long the entire effect should live.

Synchronize with Orientation:

If the effect is parented, it will use the orientation of the parent link.

Position Variation: Where particles should first appear. Sub-parameters Noise Type, Min, and Max allow for variation in particle position start.

Textures and appearance

Texture Map: The texture map to use on the particles.

Number of X, Y repeats:

This allows the texture map to be sliced up into smaller sub textures which is useful for two things: If Animation speed is set to zero, a new particle will be 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 will play through the sub textures like the pages of a flip book, allowing for animated particles.

Rotation Speed: How fast 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 blending off for more solid looking particles.

Alpha Threshold: Controls the clipping of textures based on an alpha threshold. Useful for controlling the shape of solid particles.

World Interaction

Coordinate Space: Controls if particles move along with the emitter that created them. Particles in local space will 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.

Collision bounce factor:
How much to multiply the velocity with (usually reduces velocity) as a result of the collision.

Collision Result: What to do when the particle strikes a collision plane. *Bounce* will have the particle bounce but the particle maintains its life. *Bounce and die* has the particle die. *Stick* has the particle maintain its position as well as its life. *Die* means the particle is immediately terminated.

Keyframes are used to vary the parameters of the particle systems as a function of time. All keyframes have a time associated with them which 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 attempt to maintain this requirement.

Keyframe Properties

Color:	The color and alpha value to draw the particle with, colors will linear interpolate between key frames.
Size:	The size of particles, the size will linear interpolate between key frames.
Streak:	The amount the particles should smear in the direction that they are moving.
Velocity:	An impulse to apply to a particle as they hit the key frame time.
Velocity variation:	A random velocity to apply to a particle as they hit the key frame time
Wind:	An acceleration factor to apply to the particles, will linear interpolate between key frames.
Air resistance:	When set to true, particles will not go faster then the wind speed.
Gravity:	A constant acceleration force applied every frame. <0,0,0> is good for lighter then 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:	This is applied to each particle when it is emitted. This can give the particles random variations in color.
Size variation:	This is applied to each particle when it is emitted; it's a fudge factor in addition to the size key frames for varying particles size.
Render Style:	Indicate how the graphics are drawn. This is typically set to *billboard*. *CameraAligned* will align the normal of the billboards to the camera. *velocityAligned* will align the normal of the billboards in the direction of the particles velocity. *Sparks* uses line graphics instead of billboards.

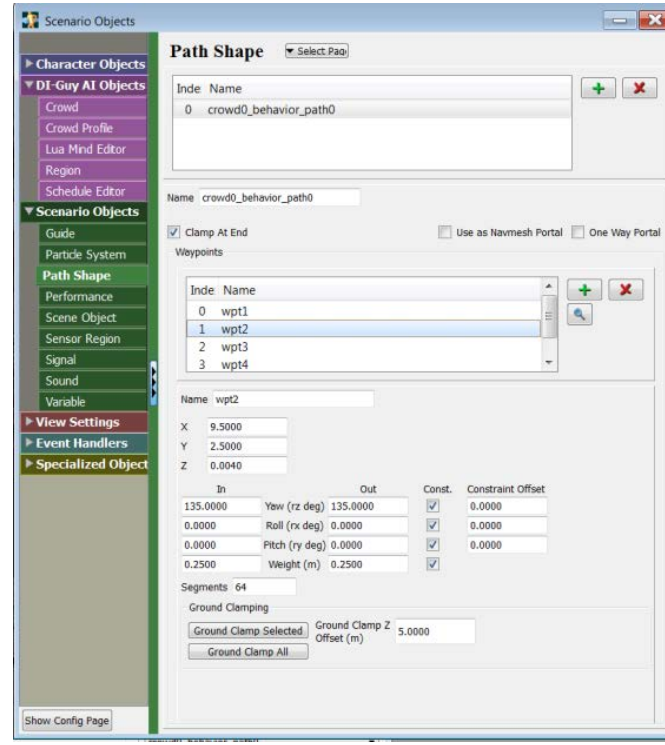
5.2.3 Scenario Objects - Path Shape Tab

The Path Shape Tab allows users to create paths (without characters or beads) for use through the DI-Guy API. Once you hit the *create* button, any waypoints added to the scenario by clicking in the 3D Window in *path insert* mode will become part of the current path shape.

Waypoints on this tab behave just like the waypoints used to define normal character paths.

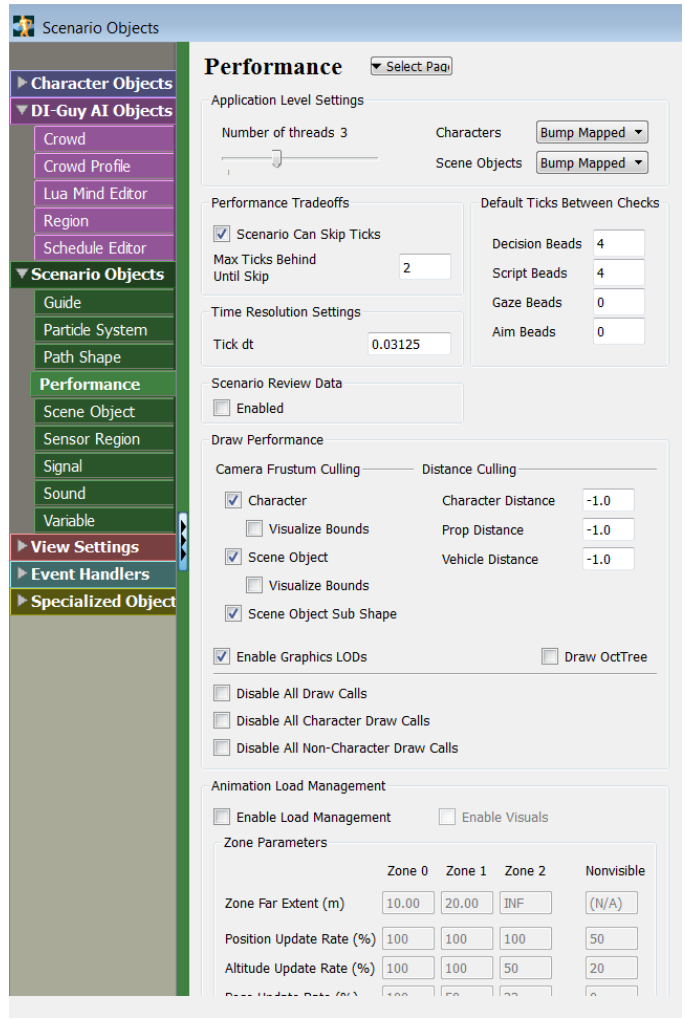
Note that because the path shapes are available to all characters, elements like action beads, script beads, etc., cannot be used on them.

Characters can be made to use these paths using the API calls that leverage path shapes.



5.2.4 Scenario Objects - Performance Tab

The Performance tab of the Scenario Objects palette provides several parameters that allow you to adjust performance of DI-Guy Scenario, so as to accommodate the various processing tradeoffs in real-time simulation.



Tick dt

DI-Guy Scenario tries to recalculate behavior (*update*) for characters at the rate *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 with no improvement in scenario visual results since changes that fast are undetectable by the human eye.

Default Ticks between Checks

There are four kinds of script activity that can burden performance within DI-Guy Scenario. They are the Decision Bead, the Script Bead, Gaze Bead, and Aiming Bead. The burden is large when the conditionals that trigger these activities span an extended period of time, rather than trigger on a single event. Oftentimes the rate these beads need to be checked can be less than the frame rate. You can reduce the burden of the conditionals using the fields provided here. For example, by setting the Decision Logic skip parameter to 3, the conditional associated with the decision logic is tested every 4 ticks, rather than every tick. The larger these numbers, the smaller the load on the DI-Guy Scenario.
It takes time to search for and possible evaluate beads.

Performance Tradeoffs

If DI-Guy Scenario cannot run at real time, this method allows the software to stay synchronized with real time.

Ground Clamping

DI-Guy Scenario offers two methods for doing ground clamping, OpenGL and OctTree. OctTree is typically the recommended setting. Depending on the specifics of your scenario and hardware, opengl might be chosen if memory is scarce.

Levels of Detail (LODs)

DI-Guy Scenario provides seven LODS for most of its character appearances to help engender high rendering performance. This can be turned off by unsetting the checkbox.

Camera Frustum Culling

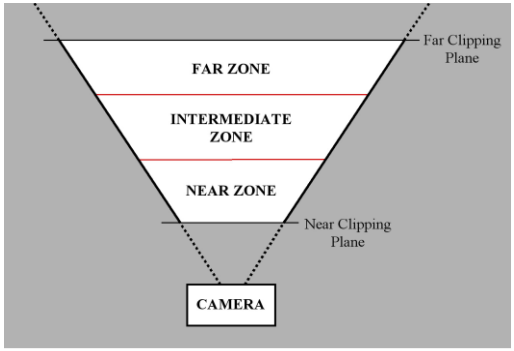
By checking the Character checkbox, characters outside the view frustum will not be rendered. This is achieved by associating extent boundaries with each character appearance. For debugging purposes, you can visually check these boundaries.

Load Manager

The Load Manager allows the user to make performance improvements when working with scenarios that contain many characters.

The Load Manager only works for single view situations. If you are using multiple views, do not use the Load Manager.

The viewing world is first divided into two regions; the visible region and the non-visible region. In the non-visible, it is important to keep track of where the character is, since you need to know when the character may move into the visible region. This being said, with many characters it may be desirable to not check every frame and stagger the checks across the characters. The Position Update Rate in the Non-Visible Characters box accomplishes this by allowing the user to set the percentage for how often a non-visible character has his position updated.



The visible region (i.e. – the frustum) is then divided into three zones (think of them as near, far, and very far). The two Zone Far Plane parameters indicate the distance from the camera to the two "walls" that separate the 3 zones from each other. The user can then set how often the position update happens in each zone, similar to the non-visible characters. In almost all circumstances however, you would want this to be 100%. The user also has two other fields to work with that are quite useful. Pose Update indicates how often the non-position joints are updated. Motion LOD indicates which joints get updated (see table below).

Motion LOD	Description

1	animate all joints
2	stop animating wrists and ankles
3	stop animating elbows and knees
4	stop animating everything but pelvis and position
5	stop animating everything but position

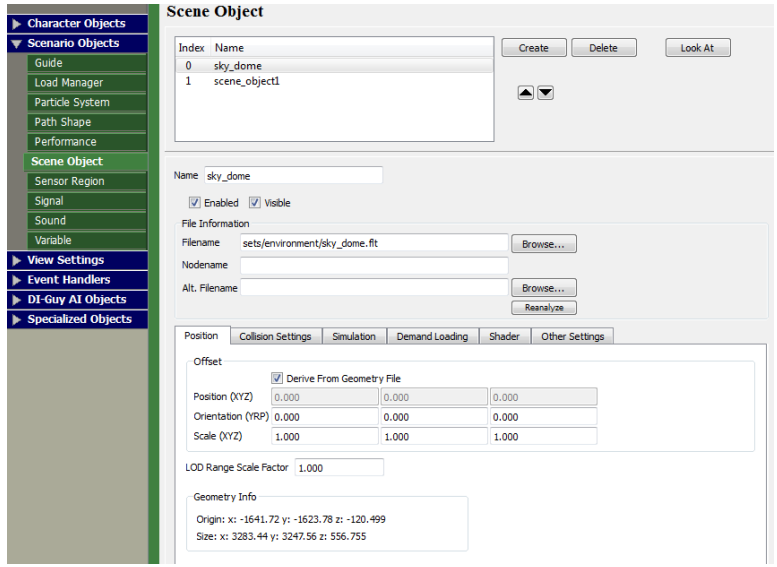
5.2.5 Scenario Objects - Scene Object Tab

DI-Guy Scenario loads terrain models and other objects that can become part of the scenario. The Scene Object tab of the Scenario Objects palette lets you manage and load your scene objects.

Use the Create and Delete buttons to add and remove scene objects. Once you have created a scene object specify the file name of the OpenFlight (.flt) file to be displayed.

The Scene Object tab also lets you change the scale of a scene object and reposition and reorient it within the scenario.

Sometimes you will load a scene object and not be able to see it because the geometry is located far from the camera. You can locate the scene object by clicking the "Look At" button on the scene object tab. It causes the current camera to aim at the average position of all vertices in the current scene object.



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.

Demand Loading

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:

1) 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.

2) Else, if the camera is closer to the bounding box of the scene object than the load radius and if the scene object is not being displayed, then DI-Guy scenario displays the scene object. DI-Guy Scenario will load 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.

Turn on demand loading by checking the *Demand Load Enabled* check box. Specify the load and unload radii in the available fields.

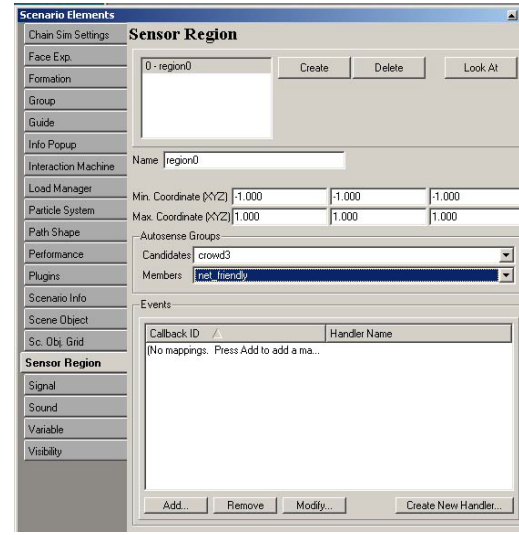
See the Scene Object Grid tab for information on supporting tiled terrain databases.

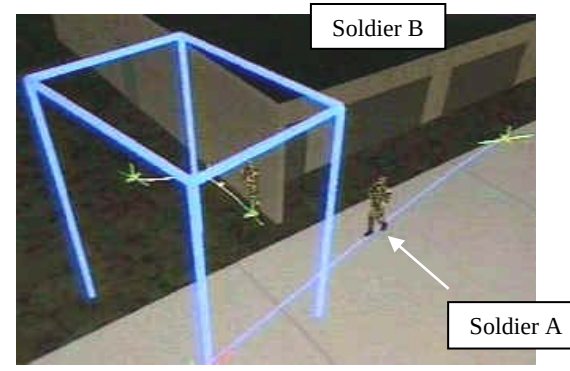
5.2.6 Scenario Objects - Sensor Region Tab

A sensor region detects when a person or vehicle enters a specified area and makes the information available to the system. The information can be used to control the actions of people in the scenario.

You specify the area for a sensor region by defining a 3D rectangular solid. The parameters are entered on the

Sensor Region tab of the Scenario Objects palette.





When Soldier A enters the sensor region, Soldier B will shoot him. The sensor region is show as the blue wireframe.

Visual Representation

Each sensor region is represented in the 3D Display window by a wireframe box. The current sensor region is drawn white. All other sensor regions are drawn blue.

Creating Sensor Regions

To create a new sensor region with index *n*, select sensor region *n-1* and press the "create" button.

The new sensor region will become the selected sensor region. Indices of following sensor regions will be increased by one to make room for the new sensor region.

Sensor regions are not attached to specific paths or people, so it does not matter which person and path is selected when editing sensor regions.

Sensor Region Name

You can assign names to sensor regions to help distinguish among them. Do not use the same name twice.

Deleting Sensor Region

To delete a sensor region, select a region and click 'delete'.

Sensor Region Position

The position and size of the sensor region is specified by the X, Y, and Z positions of diagonal corners of the rectangular box, given in world coordinates.

X	Min is bottom near left corner X position default is –1 Max is top far right corner X position default is 1
Y	Min is bottom near left corner Y position default is –1 Max is top far right corner Y position default is 1
Z	Min is bottom near left corner Z position default is –1 Max is top far right corner Z position default is 1

AutoSense Groups

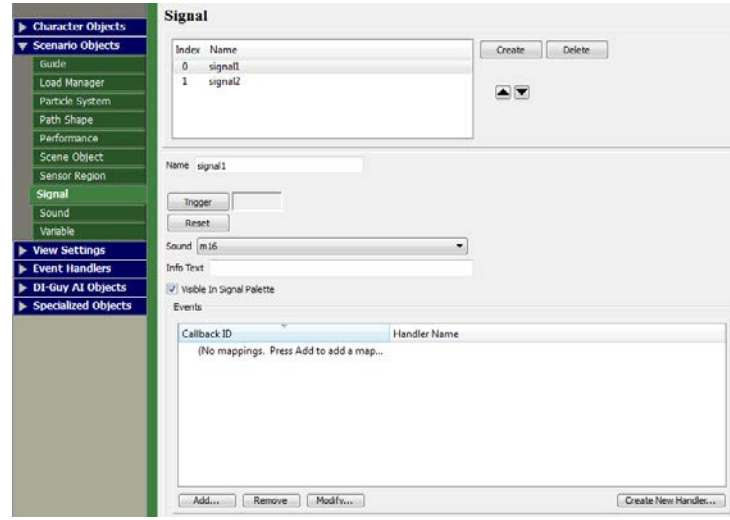
A group of characters, as defined in the Groups tab of the Scenario Objects panel, can be made eligible for autosensing. Characters that are autosensed can be made to call callback functions that have been registered with the `CALLBACK_ID_AUTOSENSE_CHARACTER_ENTERED or CALLBACK_ID_AUTOSENSE_CHARACTER_LEFT` ids.
Use the *candidates* field to specify a group of characters for autosensing.
You can also have an "output" group that will be populated by characters in the Candidates group that are currently in the sensor region. Specify the name of this group in the *members* field.

5.2.7 Scenario Objects - Signal Tab

Signals are used to coordinate behavior of people and vehicles, and to allow one person or vehicle to react to the behavior of others. Signals can be used to qualify decision logic (decision beads, script beads, and script events,) and they can be triggered by decision logic. This tab allows you to define signals you will need in your scenario.

Sounds can be associated with signals, so the sound is played when the signal is triggered. To associate a sound with a signal, first define the sound on the Sound Tab of the Scenario Objects palette.

If you would like a signal to appear on the Interactive Signals Palette, check the *Visible in Signal Palette* checkbox at the bottom of the tab.

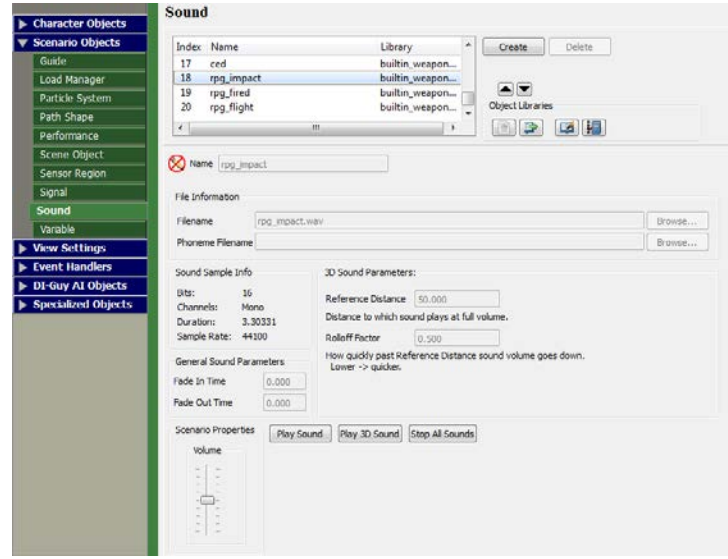


5.2.8 Scenario Objects - Sound Tab

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. The Sound tab of the Scenario Objects palette is a means of defining sounds, giving the sounds names, and associating them with a specific .wav or .ogg file.

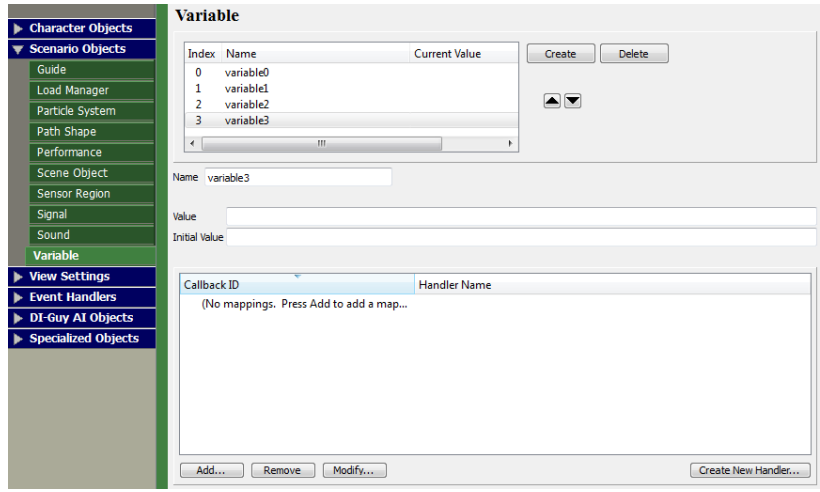
DI-Guy Scenario supports use of .wav sound files. To add a sound to the system click the "Create" button, give the sound a name, then browse the desired .wav file into the system. You can play the sound you have specified by clicking on the "Play" button. You can attach a sound to the scenario using 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.



5.2.9 Scenario Objects - Variable Tab

Just as users can create their own variables associated with each character, users can also create variables associated with the entire scenario.

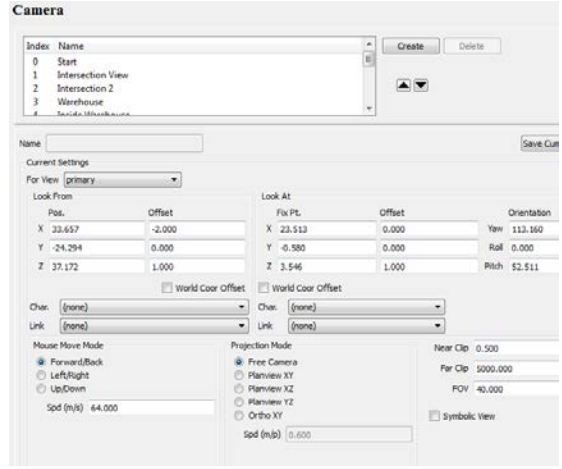


These variables can be used a variety of purposes, particularly for logic. Variables created may be accessed from other place in DI-Guy Scenario, or by user applications using the DI-Guy SDK that may have loaded the scenario created in DI-Guy Scenario. Variables can be set to a value through the GUI,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

5.3 View Settings Tab

The View palette lets you control overall viewing conditions in the 3D window(s). You can change where the virtual camera is located and where it is pointing on the Camera tab, change the sky color or add fog atmospheric effects on the Fog/Sky tab, and change the color and intensity of lighting on the Light tab.

5.3.1 View Settings - Camera Tab



The DI-Guy Scenario 3D window has a virtual camera that has a position (where the camera is looking from), a fixation point (where the camera is looking at), and a number of other parameters.

A scenario can have any number of saved camera settings that can be loaded into the camera by clicking on a Saved Settings entry in the Camera tab. (Note: Decision Bead logic and hotkeys also can load a saved setting into the current camera)

The hotkeys for loading camera settings into the current camera are:

Hotkey	Function
0	Load camera settings 0
1	Load camera settings 1
…	…
9	Load camera settings 9

You can also save the current camera settings into a saved settings slot by using the following hotkeys:

Hotkey	Function
<shift>0	Save camera settings 0
<shift>1	Save camera settings 1
…	…
<shift>9	Save camera settings 9

Camera Motion using the Mouse

There are several modes for steering the camera via the mouse in DI-Guy Scenario. You can select the mode you want using the Mouse Move Mode toggle buttons. The following table describes how the buttons on the mouse affect camera motion in each mode.

`Forward/Back`	Left click moves the camera forward into scene. Right click moves backward out of scene.
`Left/Right`	Left click moves the camera to the left. Right click moves the camera to the right.
`Up/Down`	Left click moves the camera up, away from the ground. Right click moves the camera down toward the ground.

In each of these modes, motion of the mouse, left right up and down, reorients and steers the camera.

DI-Guy Scenario also provides hotkeys for selecting the mode for camera motions:

`Hotkey d`	switch camera to Up/Down mode
`Hotkey f`	switch camera to Forward/Back mode
`Hotkey s`	switch camera to Left/Right (side to side) mode

Note: In all camera modes, holding both mouse buttons simultaneously will make the camera rotate about its own axis.

Camera Speed

To change the speed of camera travel, change the value in the Spd (m/sec) field. A higher value causes the camera to move faster; a lower value causes the camera to move slower.

DI-Guy Scenario also provides hotkeys for selecting the mode for camera motions:

`Hotkey +`	increase camera speed
`Hotkey -`	decrease camera speed

Camera Projection Mode

> Note: In all camera modes you can hold the left and right mouse buttons down simultaneously and drag the mouse to rotate the camera view.

There are 4 types of camera angles from which the scenario can be viewed:

`Free Camera`	Perspective view, user can rotate and translate freely.
`Plan View XY`	Viewing the XY plane along the Z axis, translate only.
`Plan View XZ`	Viewing the XZ plane along the Y axis, translate only.
`Plan View YZ`	Viewing the YZ plane along the X axis, translate only.
`Ortho XY`	Orthographic view in the XY plane.

You can select among these modes using the Projection Mode toggle buttons.

Near Clip Distance

This field indicates the distance from the camera to the near clipping plane in meters.

Far Clip Distance

This field indicates the distance from the camera to the far clipping plane in meters.

FOV (Field of View)

This field specifies the virtual lens angle in degrees. A large value gives DI-Guy Scenario a wide-angle lens. A small value gives the virtual camera a long lens.

Symbolic View

Symbols such as arrowheads are used for characters when desiring a more abstract visualization.

Look From field

Use *Look From* when you wish to attach a camera to a person or vehicle so that it moves through the scene wherever the character moves. The *Look From* portion of the camera palette allows you to select which person or vehicle to attach the camera.

The *Link* field allows you to further specify the link (using the incoming joint name of the link) on the character the camera will be attached to. If you want to attach the camera to the right hand, you would specify link as wrist_r. Here is a list of all the joint names for the standard people models in DI-Guy Scenario *back, cervical, shoulder_r, elbow_r, wrist_r, shoulder_l, elbow_l, wrist_l, hip_r, knee_r, ankle_r, hip_l, knee_l, ankle_l.* If you do not specify a link the camera will move with the base or pelvis of the character.

Once you have selected which person to attach the camera to, and perhaps a link, you can offset the camera from the incoming joint position by specifying *xyz offsets* in the *Look From* portion of the Camera palette. These values can be specified by typing in values, or by putting the system in camera mode and using the mouse to fly the camera.

Look At field

When a camera looks at a person or vehicle, the camera focuses on a point on the person or vehicle and thus tracks it (thus keeping the character in the center of the view) as it travels. The *Look At* portion of the camera palette allows you to specify any person or vehicle as the look at target or fixation point. The drop-down menu in the *Look At* section allows you to select which person or vehicle to look at.

The *Link* field allows you to further specify which link of the character the camera looks at (using the incoming joint name of the link). If you wanted to look at the right hand, you would specify link name wrist_r. Here is a list of all the joint names for the standard people models in DI-Guy Scenario *back, cervical, shoulder_r, elbow_r, wrist_r, shoulder_l, elbow_l, wrist_l, hip_r, knee_r, ankle_r, hip_l, knee_l,*

ankle_l. If you do not specify a link the camera will look one meter above the reference point for the person

Once you have selected which person or vehicle to look at, and perhaps a link, you can displace the fixation point from the origin by specifying *xyz Offsets* in the *Look At* portion of the Camera palette.

Point of View

A common need is to have the camera follow a person, and roughly show what the person would see. This can be done by setting both the *Look From* and *Look At* fields to the same person. Once the camera is set up in this way, you can make a variety of adjustments with the offsets, either typing them or by using the mouse in camera mode.

'Go There' Camera Mode

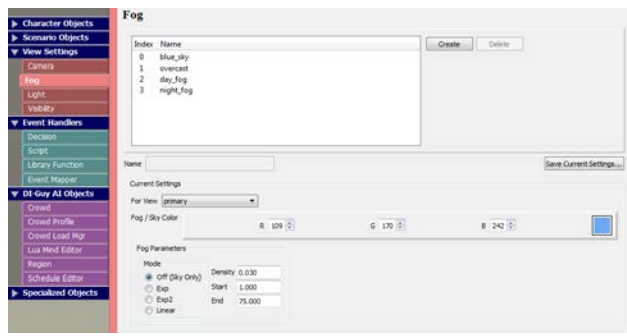
The 'Go There' camera mode automates motion of the camera once the user designates the new location. The user enters camera mode, points the mouse in the 3D window to indicate where the camera should be placed, then shift-left-click. The camera flies to the new location, and spins around to face where it came from.

5.3.2 View Settings - Fog Tab

The Fog and Light tabs of the View palette allow you to adjust lighting effects in the 3D scenario. Sky color and fog parameters are specified on the Fog Tab of the View Palette.

The Fog tab of the View Palette allows you to change the background sky color and to add one or more fog effects to the scenario. Each sky/fog combination you create is assigned a name that can be referenced in the decision logic.

The fog can be calculated in several ways depending on the decay parameters. Enter fog parameters in the available locations on the Fog Tab. You can change the color of the fog by specifying its RGB values, or by selecting a color from the color palette.

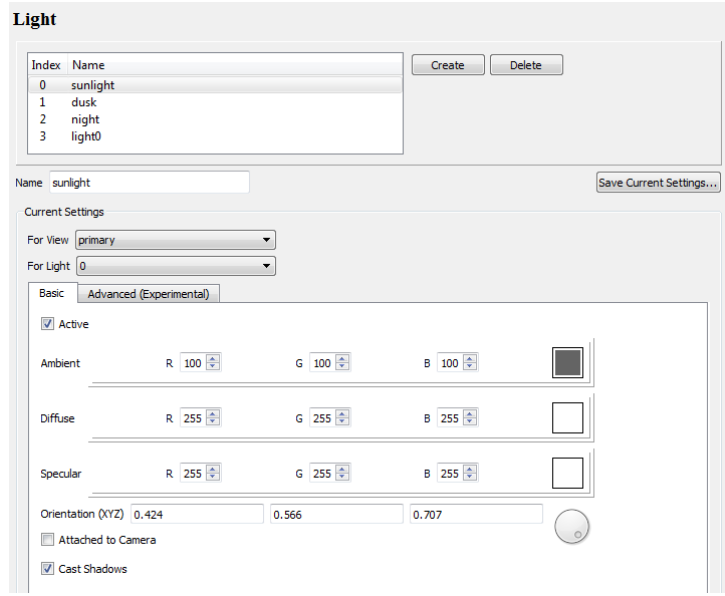


The Sky Tab of the View Palette is used to adjust sky color and fog, as shown in these images.

View Settings - Light Tab

There are several parameters that allow you to adjust lighting effects in the 3D scene. Parameters that control the light source's ambient, diffuse, and specular colors, as well as its direction, are specified on the Light Tab of the View Palette.

The same scenario viewed with three different light settings, including an addition of fog from the Sky tab.



5.3.3 View Settings - Visibility Tab

Controls on this tab affect the visibility of people, paths, regions, crowds, particle systems, waypoints, and other features in the 3D Window.

Character Visibility

You can choose to display no people, only the current person, or all people in the scenario by making a selection on the Visibility tab of the Scenario Elements palette.

`Show None`	Turns off all characters
`Show Current`	Shows only current character
`Show All`	Shows all characters

Character Labels

`Show None`	Do not show character labels
`Show Current`	Show character labels on the current character only
`Show All`	Show character labels on all characters
`Per Object`	Use the per-character setting from the character tab

Aim Trajectories

Show None	Do not show aim trajectories
`Show Current`	Show current character's aim trajectory
`Show All`	Show all characters' aim trajectories

Path Visibility

You have a choice of which paths to display in the 3D Display window. Depending on the settings you choose on the Visibility tab of the Scenario Objects palette, you can make all paths visible, only the current

path visible, or no paths visible in the 3D Display window. When displayed, the currently selected path is shown white and other paths are shown blue:

`Show None`	Turns off display of all paths
`Show Current`	Displays only the path of the current character
`Show All`	Displays paths of all characters

Waypoint Visibility

`Show None`	Turns off display of all waypoints
`Show Current`	Displays only the waypoints of the current character
`Show All`	Displays waypoints of all characters

On-Path Beads Visibility

`Show None`	Display no path beads
`Show Current`	Display only path beads for current person
`Show All`	Displays path beads of all people

Action Labels

Show None	Do not show action labels
`Show Current`	Show current character's action labels
`Show All`	Show all characters' action labels

Sensor Region Visibility

`Show None`	Display no sensor regions
`Show Current`	Displays only the current sensor region
`Show All`	Displays all sensor regions

Region Visibility

`Show None`	Display no regions
`Show Current`	Displays only the current region
`Show All`	Displays all regions

Crowd Region Visibility

`Show None`	Display no crowd regions
`Show Current`	Displays only the current crowd region
`Show All`	Displays all crowd regions

Crowds and AI

Display options similar to the above exist for member feelers, member influence area, member behavior visuals, and Lua objects. These settings are useful for analyzing crowd behavior and AI, making it easier to visualize the decision-making process of characters.

Particle Systems, Lights, and Frills

Provide extra visuals to indicate the location of particle systems and lights.

5.4 Event Handlers Tab

The Event Handlers tab handles scenario level decisions and scripts, library functions, and an event mapper for linking events to event handlers.

5.4.1 Event Handlers - Decision Tab

The Decision tab allows users to create logic without programming at the scenario level.

The upper left box shows the Decisions that have already been created. Decisions can be managed using the Create and Delete buttons

There are 10 different Event Types that can be associated with the Decision:

Manual – The user must use the trigger now button on this tab to force execution of the decision logic

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 / Crowd – The decision logic is called every time a character or crowd satisfies a particular condition. The Event Mapper is used to hook up the decision to the particular character or crowd action.

Gesture – to be implemented.

Label – A label graphic in the 3D window can be clicked on to trigger the decision.

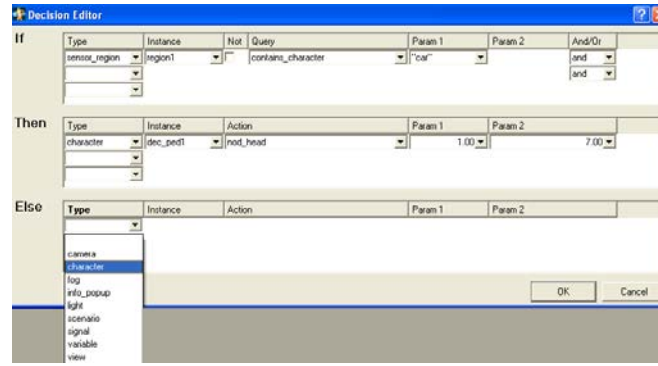
Scenario – The decision logic is called every time the scenario satisfies a particular condition. The Event Mapper is used to hookup the decision to the particular scenario action.

Sensor Region – The decision logic is called every time a particular condition is satisfied involving a sensor region. The Event Mapper is used to hookup the decision to the particular sensor region action.

Signal – The decision logic is executed when a signal is received. The signal is hooked up to the decision using the Event Mapper tab.

Variable– The decision logic is executed when a variable changes. The signal is hooked up to the decision using the Event Mapper tab.

Creating Decision Logic



In a manner very similar to the logic generator for a decision bead on the Character palette, using the Edit button brings up a easy-to-use interface for generating logic. Please see the Decision Bead Tab of the Character Elements Palette sub-chapter for more information on decisions and creating decision logic.

Again, similar to the Decision Bead used in the Character palette, a Decision can be converted into a script. This is often a good way to get started on writing a script. The bottom right buttons allow both previewing and conversion of Decisions into Scripts.

5.4.2 Event Handlers - Script Tab

The Script tab of the Event Handlers palette behaves identically to the previous Decision tab, with the exception that a script is used in place of the GUI-generated logic.

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

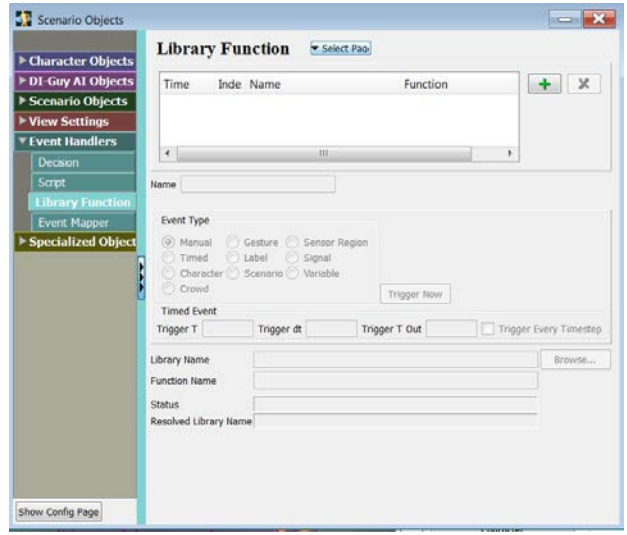
A majority of the DI-Guy API (over a thousand functions) is available through the scripting interface. You can find descriptions of these functions in the documentation that is installed when you install DI-Guy Scenario on your computer. Look in diguy_sdk_reference

5.4.3 Event Handlers - Library Function Tab

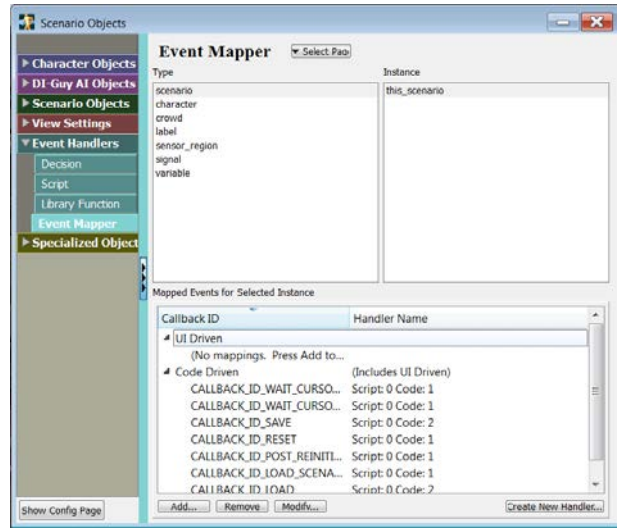
The Library function tab of the Event Handlers palette behaves identically to the previous Decision tab, with the exception that a library function from a dynamically linked library (dll), that is associated with DI-Guy Scenario as a plugin, will be called in place of the GUI-generated logic.

The user can browse for the library and the designate the function in the library to be called. The status and Resolved Library name field will display whether DI-Guy Scenario was able to find the function.

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



5.4.4 Event Handlers - Event Mapper



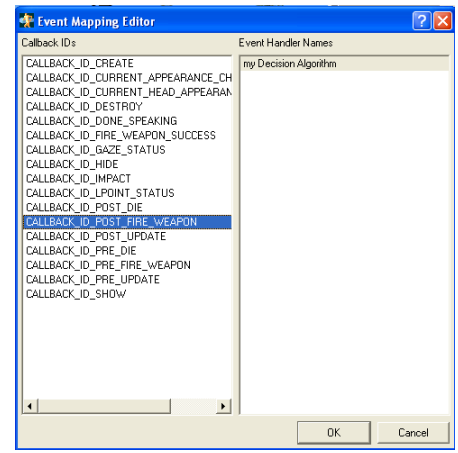
The Event Mapper maps events generated ***from*** characters, scenarios, sensor regions, or signals ***to*** decisions, scripts, or library functions as defined in the three previous Event Handlers tabs.

The steps for doing this are simple.

1. Select the type of event you are trying to map (character, scenario, sensor region, or signal). The instance window will then populate with all the available object instances of the type selected. For example, if you select character, you will then see all the characters in your scenario appear in the instances box.

2. Select the instance of the event type that you want to map.

   OK, now we have now defined the type and particular instance of the event "emitter". We now need to add one or more specific mappings from this "emitter" to an event handler.



3. Press *Add…* The Event Mapping Editor will now appear. Depending on the particular event type, you will see a different list of all the available callbacks for that type in the left column. In the right column will be all the decisions, scripts, and library functions that you have designated as Event Handlers for this particular event type on three previous tabs of the Event Handler palette.

4. Select the Callback ID you are interested in mapping.

5. Select the Event Handler to which the callback will be mapped.

6. Select *OK*.

Callback IDs

The following is a sampling of callback IDs classified by Event Type. Note that this is NOT an exhaustive list and users are encouraged to consult the include files such as `diguyCharacter.h`, `diguyCrowd.h`, `diguyCharacterGesture.h`, `diguyScenario.h`, `diguySensorRegion.h`, etc., for the latest information regarding callbacks.

Character Callbacks

CALLBACK_ID_CREATE — This callback will be called when a new character is created.

CALLBACK_ID_CURRENT_ACTION_CHANGED
This callback will be called when a character's current action is changed.

CALLBACK_ID_CURRENT_APPEARANCE_CHANGED
This callback will be called when a character's current appearance is changed.

CALLBACK_ID_CURRENT_HEAD_APPEARANCE_CHANGED
This callback will be called when a character's current head appearance is changed.

CALLBACK_ID_CURRENT_TOUT_REACHED

This callback will be called when time equals the character's tout value.

CALLBACK_ID_DESIRED_ACTION_CHANGED
This callback will be called when a character's desired action is changed.

CALLBACK_ID_DESIRED_ACTION_REACHED
This callback will be called when a character's desired action is reached.

CALLBACK_ID_DESTROY — This callback will be called when a character is destroyed.

CALLBACK_ID_DONE_SPEAKING — This callback will be called when the character has finished speaking the contents of a `speak()` function call.

CALLBACK_ID_END_OF_PATH_REACHED
This callback will be called when a character reaches the end of its current path.

CALLBACK_ID_FIRE_WEAPON_SUCCESS

This callback will be 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 This callback will be called after the character's gaze has experienced a status change.

CALLBACK_ID_MANUALLY_INVOKED
This callback will be called when an event handler is manually invoked.

CALLBACK_ID_GUIDE_ORIENTATION_ACQUIRED
This callback will be called when the character has reached its desired orientation as set by set_desired_orientation().

CALLBACK_ID_GUIDE_ORIENTATION_UNACQUIRED
This callback will be called if the character turns too far away from its desired orientation after it has been previously reached

CALLBACK_ID_GUIDE_POSITION_ACQUIRED
This callback will be called when the character has reached its desired position as set by set_desired_position().

CALLBACK_ID_GUIDE_POSITION_UNACQUIRED
This callback will be called if the character moves too far away from its desired position after it has been previously reached

CALLBACK_ID_HIDE This callback will be called when the character is being hidden for any reason.

CALLBACK_ID_IGUY_INTERACT Called when the user clicks on another character in IGuy mode

CALLBACK_ID_IGUY_INTERACT_WITH_SUBJECT
Called when the user clicks on another character in IGuy mode

CALLBACK_ID_IMPACT This callback will be 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 This callback will be called after the character's lpoint (left arm pointing) has experienced a status change.

CALLBACK_ID_POST_CREATE This callback will be called after a character's full creation.

CALLBACK_ID_POST_CREATE_GEOMETRY
This callback will be called after a character's geometry is created.

CALLBACK_ID_POST_DIE– This callback will be called when the character has been told to die, after a die action has been selected and initiated.

CALLBACK_ID_POST_FIRE_WEAPON	This callback will be called when the character has been told to fire its weapon, after a final decision has been made to fire.
CALLBACK_ID_POST_UPDATE	This callback will be called after the character is updated.
CALLBACK_ID_PRE_DESTROY	This callback will be called before a character's full destruction.
CALLBACK_ID_PRE_DESTROY_GEOMETRY	This callback will be called before a character's geometry is destroyed.
CALLBACK_ID_PRE_DIE	This callback will be called when the character has been told to die, before a die action has been selected and initiated.
CALLBACK_ID_PRE_FIRE_WEAPON	This callback will be called when the character has been told to fire its weapon, before a final decision has been made to fire.
CALLBACK_ID_PRE_UPDATE	This callback will be called before the character is updated.
CALLBACK_ID_SHOW	This callback will be called when the character is being shown for any reason.
CALLBACK_ID_USER_SELECTED	This callback will be called when the character is selected in DI-Guy Scenario.
CALLBACK_ID_USER_UNSELECTED	This callback will be called on a currently selected character when a different character is selected in DI-Guy Scenario.

Scenario Callbacks

CALLBACK_ID_CREATE This callback will be called when a new scenario is created.

CALLBACK_ID_DESTROY This callback will be called when scenario is destroyed .

CALLBACK_ID_LOAD This callback will be called when a scenario is loaded.

CALLBACK_ID_LOAD_SCENARIO_FILE
This callback will be called just before a scenario loads a new file.

CALLBACK_ID_RESET This callback will be called when a scenario is reset.

CALLBACK_ID_SAVE This callback will be called when a scenario is saved.

CALLBACK_ID_SAVE_SCENARO_FILE This callback will be called just after a scenario saves a new file. With a char * pointer to the file name.

CALLBACK_ID_SCENE_OBJECT_IMPACT
This callback will be called when a weapon was fired and a scene object was struck.

CALLBACK_ID_WAIT_CURSOR_HIDE This callback will be called when a DI-Guy operation that caused a wait cursor to be shown has completed, allowing an application to hide the wait cursor.

CALLBACK_ID_WAIT_CURSOR_SHOW

This callback will be called when a DI-Guy operation that causes a wait cursor to be shown.

CALLBACK_ID_POST_DRAW This callback will be called just after a scenario finishes it's 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);
```

Sensor Region Callbacks

CALLBACK_ID_CREATE This callback will be called when a new sensor region is created.

CALLBACK_ID_DESTROY This callback will be called when a sensor region is destroyed.

CALLBACK_ID_DETONATION This callback will be called when a detonation occurs within the sensor region.

CALLBACK_ID_AUTOSENSE_CHARACTER_ENTERED

This callback will be called when a candidate character has entered the sensor region.

CALLBACK_ID_AUTOSENSE_CHARACTER_LEFT

This callback will be called when a candidate character has left the sensor region.

Signal Callbacks

CALLBACK_ID_PRE_TRIGGER — This callback will be called whenever the `trigger()` function is called, whether internally by DI−Guy or explicitly by the user in C++ or in a script. It will be called before the default handler of the function.

CALLBACK_ID_POST_TRIGGER — This callback will be called whenever the `trigger()` function is called, whether internally by DI−Guy or explicitly by the user in C++ or in a script. It will be called after the default handler of the function.

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

```
diguyCharacter* diguyCrowd::get_callback_character();
```

and

```
diguyImpact* get_callback_impact();
```

inside the callback handler.

CALLBACK_ID_CREATE — This callback will be called when a new crowd is created.

CALLBACK_ID_DESTROY — This callback will be called when a crowd is destroyed.

CALLBACK_ID_CROWD_MEMBER_KILLED

This callback will be called when a member of the crowd is killed.
- The callback character is the crowd member that was killed.
- The callback impact contains the impact information.

CALLBACK_ID_CROWD_MEMBER_IMPACT

Similar to diguyCharacter::CALLBACK_ID_IMPACT, this callback will be called when a member of the crowd is hit by a detonation. If this callback isn't present the impacted character automatically will die. Handling this callback allows for the implementation of custom damage models at a crowd level.

- The callback character is the crowd member that was hit.
- The callback impact contains the impact information.

CALLBACK_ID_NEARBY_SCENE_OBJECT_IMPACT

This callback will be called when a detonation occurs within the awareness radius (as set by set_awareness_radius()) of the crowd's current bounds.

- The callback character is the character that caused the detonation.
- The callback impact contains the impact information.

CALLBACK_ID_NEARBY_WEAPON_FIRED

This callback will be called when a weapon is fired within the awareness radius (as set by set_awareness_radius()) of the crowd's current bounds.

- The callback character is the character that fired the weapon.

DI-Guy AI Objects Tab
Control autonomous, terrain-aware characters with this tab. Documentation is in the DI-Guy AI manual. Requires purchase of the DI-Guy AI option to DI-Guy Scenario.

5.5 Specialized Objects Tab

This section describes a second tier of features of DI-Guy Scenario.

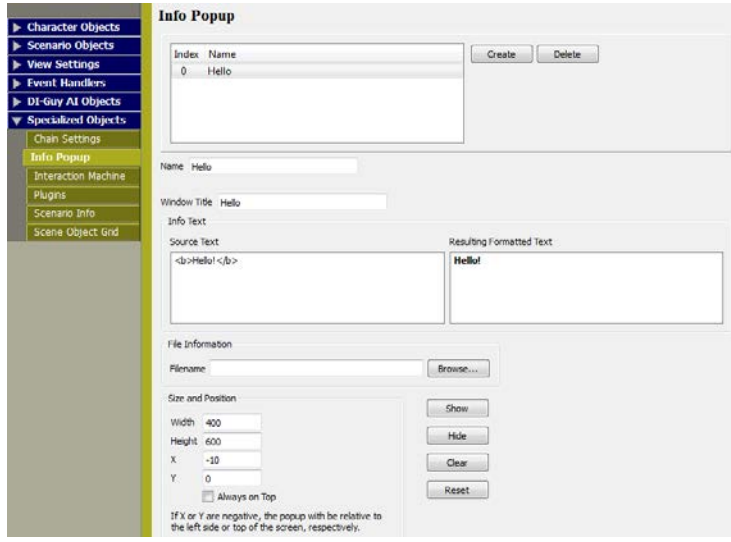
5.5.1 Specialized Objects - Chain Settings

Special DI-Guy characters use the physics described here to model chains, ropes, and cables.

Documentation pending.

5.5.2 Specialized Objects - Info Popup

Info Popups are windows used to display text and images as messages during a scenario. The display of 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. It is on your DI-Guy Scenario CD.

To create a new Info Popup window, click the *create* button on the Info Popup tab of the Elements palette. Then you can use the browse function to assign an HTML file to the window. The window can be referenced by name in decision logic.

Other fields in the Info Popup tab allow you to specify the size and location of the Info Popup window, when it is displayed, the title displayed in the header of the window, and the duration of display if you want it to disappear on its own. You can preview the display using the *show* button.

Decision control of Info Popup

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, using a script bead, or using 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
```

5.5.3 Specialized Objects - 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 users with simple yes-or-no questions, and also "dialog tree" type conversations with characters in a scenario.

It is called a "machine" because it is implemented as a straight-forward state machine. A state machine has some number of "states" it can be in. Each state responds to some number of inputs, which can advance the machine to a new state or end the interaction.

Graphically, interaction machines are represented as dialog boxes in DI-Guy Scenario. There are three areas to the dialog: the top, called the header, which provides a one-line title for the machine; the middle, which shows informational text of the current state; and the bottom, which presents the user with options they can click to proceed to a new state.

Example Scenario

An example of a straight-forward interaction machine is provided with DI-Guy Scenario, called “Interaction_Machine_Example.dss”. Open that scenario. This example has an interaction machine that is a simple conversation tree in which the scenario user interacts with an injured accident victim.

(Insert screenshot showing initial open interaction UI here.)

In the scenario there is an interaction machine called "converse_with_victim". To see the interaction machine, open the Scenario Objects window, open the Specialized Objects tab group if it isn’t already open, and select the Interaction Machine tab.

The interaction machine has four states:

- "begin" – the beginning state of the interaction
- "ask_about_injury" – an intermediate state
- "goodbye" – last exchange before interaction ends
- "end" – final state that ends the interaction

(Insert screenshot of State Settings area of Interaction Machine page here.)

When the interaction starts the machine is in the state "begin". The user is presented with a dialog that has the header "Talking to Accident Victim". The info portion of the dialog tells what the user sees. The inputs presented are:

1. "Are you injured?"
2. " I'm sorry, but I don't have time to stop."

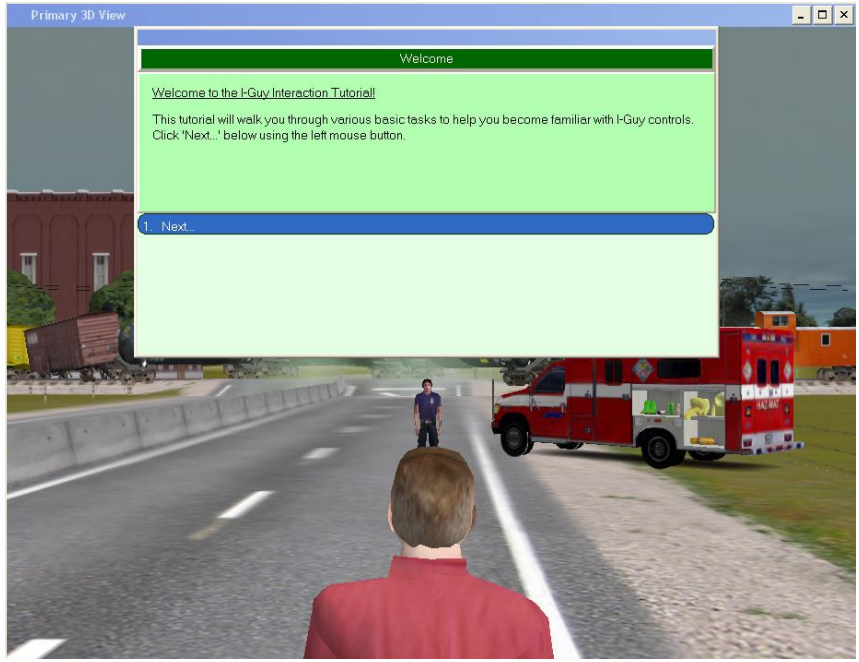
If the user selects input 1, the machine proceeds to the state "ask_about_injury", where new info is presented telling the user about any injuries the victim may have, and new inputs that present further questions the user can ask of the victim or actions the user can take. (Note that not all inputs need to be conversation or questions; inputs can also describe actions that the user may take.)

If the user selects input 2, the machine proceeds to the state "goodbye", where new info tells that user that the victim is angry at being ignored. There is one input, "(Move on.)", that ends the interaction and closes the interaction machine.

Note that the “goodbye” state can be arrived at from two other states: from “begin” by not asking about injuries, or from “ask_about_injuries”. The entry script of the “goodbye” script shows how to do different things depending on how the state was entered.

Advanced Example featuring I-Guy

A more advanced scenario is provided with DI-Guy Scenario for showing how interaction machines work and how they are authored. Open the scenario “Iguy_Interaction_Demo.dss”.



The scenario starts with a series of “click-through” screens explaining how to control the avatar. Let’s understand how to set this up. We’ll start by looking at the “im_tutorial_move” script from the Event Handlers Scripts page. This script starts up almost immediately at 0.1 seconds. The key commands in it are

`im = this_scenario:find_interaction_machine ("im_tutorial_move");` which gets a pointer to the previously created im_tutorial_move interaction machine, and

```
im:begin_interaction();
```

which makes the first call into it and brings up the interaction UI.

The im_tutorial_move interaction machine is defined in the Specialized Objects->Interaction Machines page. This scenario, in fact, defines 6 separate interaction machines, but we'll focus solely on this first interaction machine:

```
im = this_scenario:get_active_interaction_machine();
input_id = im:get_last_selected_input();

if (input_id eq "begin") then
       im:set_heading("Welcome");

       my info_text =
              "<u>Welcome to the I-Guy Interaction Tutorial!</u>" ..
              "<p>" ..
              "This tutorial will walk you through various basic tasks to help
you " ..
              "become familiar with I-Guy controls.  " ..
              "Click 'Next...' below using the left mouse button.";

       im:set_info(info_text);

       im:clear_inputs();
       im:add_input("movement", "Next...");

       im:update_ui();
elseif (input_id eq "movement") then
```

The first two lines are very common to most interaction machines scripts. They retrieve the current interaction machine and the last input into the machine. If there has been no input yet, the input will be returned as "begin".

An interaction machine dialog is divided into three basic parts: the heading, the information text, and the input section. The information text can be in plain text or simple HTML-style format. 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 don't need to put a number in when you have multiple choices. Notice that later when the user selects an input, the interaction machine is called with the new input_id.

For more information about the DI-Guy Interaction Machine, consult the SDK reference manual for the class diguyInteractionMachine.

Hot Objects

The IGuy_Interaction_Demo also features clickable "hot objects". How do they work? It is really quite simple now that you are becoming familiar with the Labels and Interaction Machine examples we've already discussed.

In the above picture, you can see that we have modified the label of the paramedic to indicate that the user should click on him. If we look at the Event Handler->Event Mapper page, and select "character" in the upper left box and "Paramedic" in the upper right box, you can see that the callback "im_paramedic_talk" is invoked when the event "IGUY_INTERACT" happens. In other words, if you are in IGuy input mode in DI-Guy Scenario, and you click anywhere on the character "Paramedic", a callback will occur 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.

5.5.4 Specialized Objects - Plugins Tab

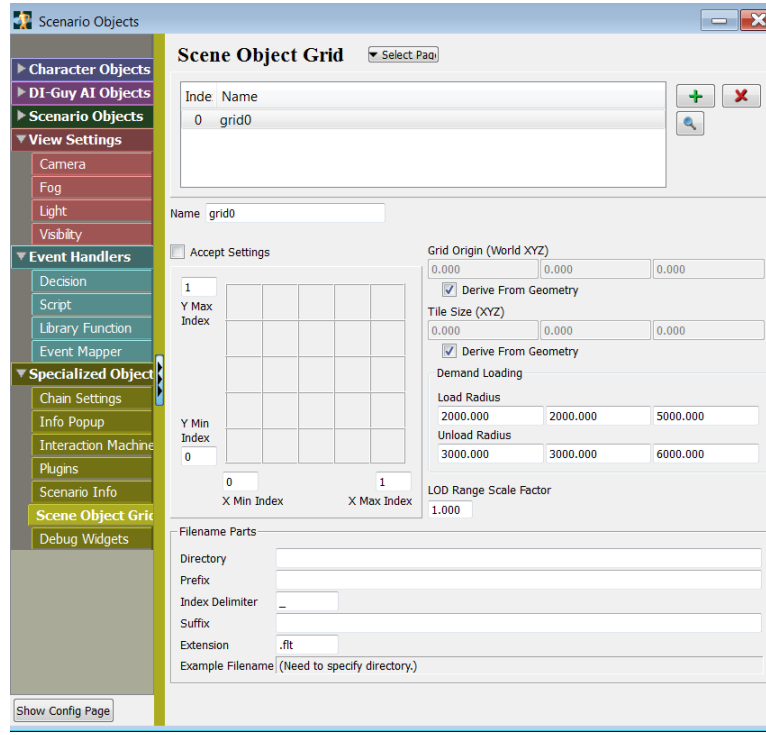
The Plugin tab of the Scenario Objects Palette will lists any plugins that are either currently loaded or have been selected previously as being required by this scenario. Any time a user makes a plugin that is required by his scenario to run properly, he should go this tab and highlight that plugin. Any subsequent run of the scenario where the plugin has not been loaded will issue a warning.

5.5.5 Specialized Objects - Scenario Info Tab

The Scenario Info tab is a convenience that lets scenario developers write notes about the current scenario.

5.5.6 Specialized Objects - Scene Object Grid Tab

DI-Guy Scenario supports very large terrain databases that are composed of tiles arranged in regular rectangular grids. Tiles are loaded and unloaded 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 Scene Object Grids tab of the Scenario Objects palette lets you specify a set of terrain tiles and the parameters needed to control their loading and use within DI-Guy Scenario.

All the tiles that compose a grid must be named systematically in the form of <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, '_' is the index separator, no suffix is used, and '.flt' is the extension.

You can reposition the Scene Object Grid by specifying an XYZ offset in the *Grid Origin* fields. If the check box *Derive From Geometry* is checked, 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 check box *Derive from Geometry* is checked, then the size of the tiles is determined automatically from the terrain model.

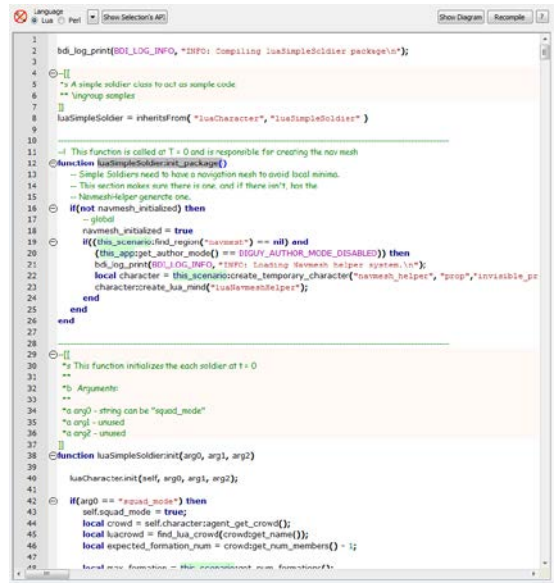
DI-Guy Scenario tabs tiles as needed, focusing on those tiles located near the camera's fixation point. As the fixation point moves throughout the terrain, tiles load and unloaded as necessary. Parameters that control the loading and unloading process are *Demand Load Radius* and *Demand Unload Radius.* The Demand Load Radius forces tiles to be loaded if they are within the specified distance of the fixation point. The Demand Unload Radius forces tiles to be unloaded if they are further from the fixation point than the specified distance.

Note: DI-Guy Scenario does not attempt to hide the loading and unloading of tiles, as would be required in an image generator. The purpose here is to allow a user to populate a very large space, working in just one region at a time.

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.

5.6 The DI-Guy Lua Code Editor



Recent versions of DI-Guy Scenario include a powerful Lua Script Code Editing system that supports code coloring, syntax highlighting, autocomplete and automatic display of DI-Guy function calls. The Code Editor is used for script bead editing and AI Minds. Some of the features of the Editor are described below.

Simple auto complete based on document inspection

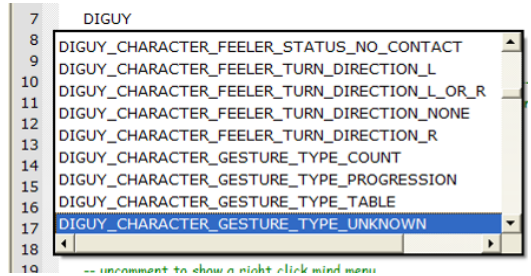
Whenever the user begins to enter code in the Code Editor, it will attempt to match other words in the code to the current string being entered. If a potential match(es) is found, the Code Editor brings up a multiple choice box showing those autocomplete words. Note in the image below how sldr07_0 is understood from the earlier context to be a potential match.

```
--sldr07_0
-- this block of code is only called the first time the sldr07_0 is created
if(sldr07_0 == nil) then
    local character = this_scenario:find_character("sldr07-0");
    sldr07_0 = luaSimpleFriendlySoldier:new(character);

    if(sldr07_0.one_time_init) then
        sldr07_0:one_time_init();
    end
    -- your one time init code here

    sl
```

```
sldr07
sldr07_0
sleep
```

Autocomplete for DI-Guy Constants

The autocomplete 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.



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.

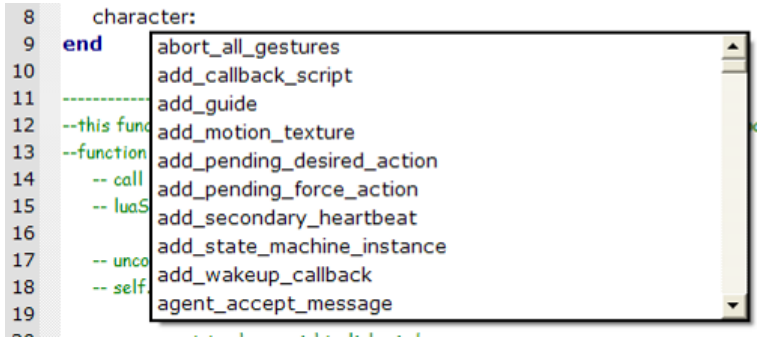
- API Based on Known Objects: If an object name is one that is explicitly set by our script engine e.g.: "this_scenario", "this_character", "this_app" or "callback_object" the list of functions that that object has should pop up. The callback_object type is derived from the type of callback that is being 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've implement a simple rule system for determining what type an object is based on the name of the object.

NOTE: The list below shows what conventions the Code Editor will recognize. 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", etc. USE THE 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

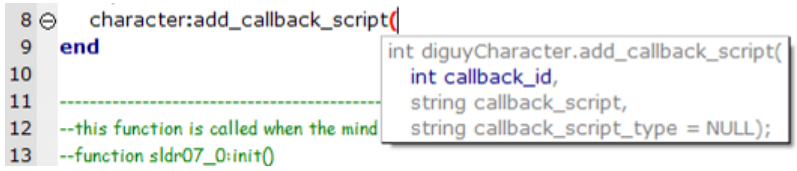
These conventions match code automatically generated by our decision bead to script bead converter.



Example of diguyCharacter object autocomplete list

Calltips

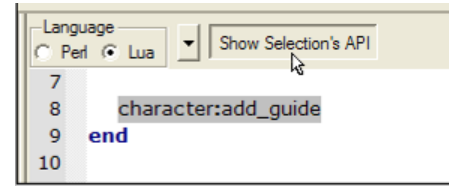
Whenever "(" is entered, the code editor will show *calltips* for the function in question:



The calltip information is found in ($DIGUY)\doc\diguy_sdk_reference\api. The ESC key can be used at any time to hide the calltip.

Full Documentation in the API Viewer

The Code Editor includes a simple DI-Guy API documentation viewer. The documentation is accessible directly via Help->API Docs or indirectly via the **Show Selection's API** button.

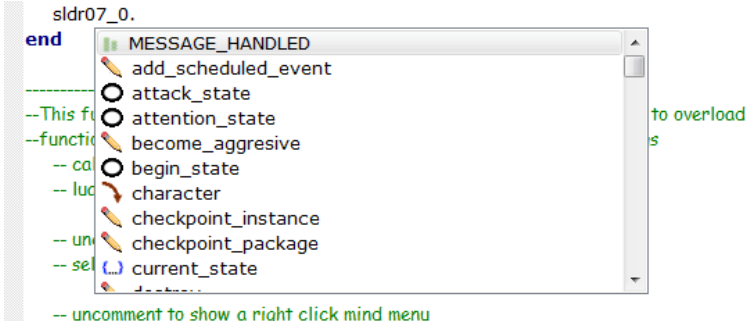


EXAMPLE: If *character:add_guide* is selected and then S**how Selection's API** is clicked, DI-Guy Scenario will launch the DI-Guy API viewer and scroll to the object/function entry. Note this will only work with objects that match the naming rules above.

Querying Lua for related objects and fields

The Lua language has introspection functionality, allowing users to query what objects and fields a given object might have. If it is not possible to guess an objects type, the Lua interpreter is queried for an object's existence.

EXAMPLE: If "*luaCharacter:"* is typed 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 "*luaCharacter."* is typed in, the autocomplete list will include all fields including strings, booleans, numbers, tables, as well as the functions that the object has.

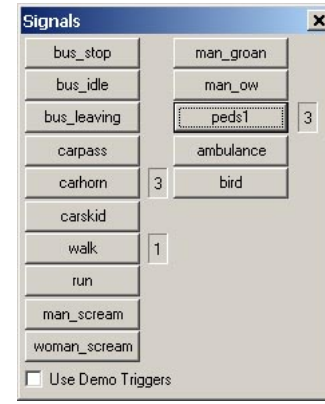


The parser will attempt to identify nested luaObjects as well: e.g. *luaSmartObjects.registry.socialize_men*

Nested luaObject autocomplete also work on any built-in lua library. Therefore, if the user types "*math."*, "*string."*, or "*table."* , the editor will bring up these built-in library contents. Calltips for built-in lua library should appear as well. Note that unfortunately a local variable cannot be analyzed in this fashion.

6. The Signals, Movie, and Log Palettes

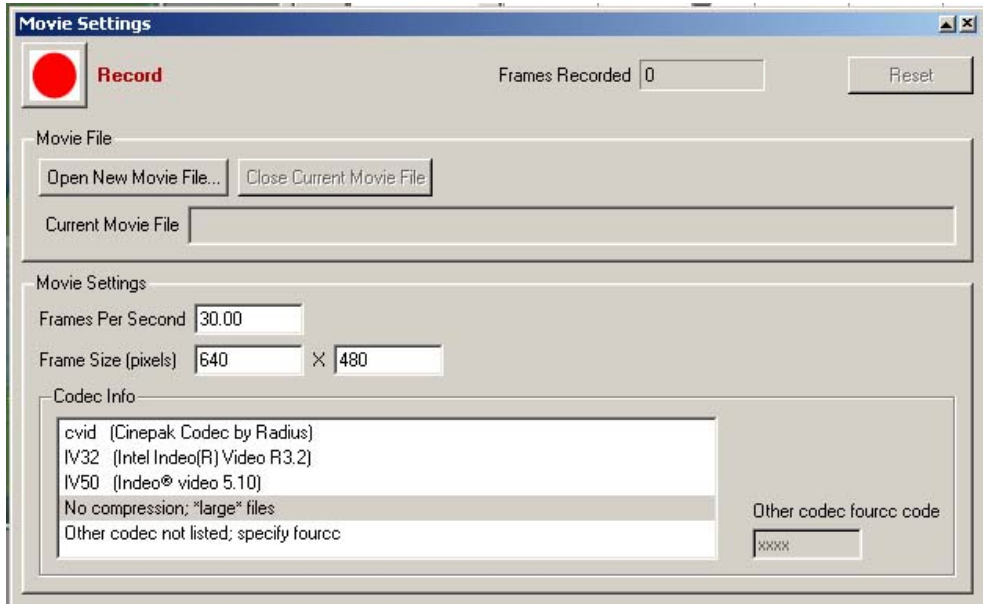
6.1 Signals Palette



The Interactive Signals palette allows you to to trigger a signal during the scenario and display the status of selected signals. Because signals are available to the decision logic, the Signals palette gives you a way to control the execution and flow of scenarios interactively.

There is a check box on the Signal tab of the Scenario Objects palette that selects a signal for display on the Interactive Signals palette. The number just to the right of a signal's button indicates how many times a signal has been activated or triggered since the beginning of the scenario. You can click on the name of any signal showing on the Interactive Signals palette to trigger the signal. The signal trigger counts are set to zero upon reset.

6.2 Movie Settings Palette

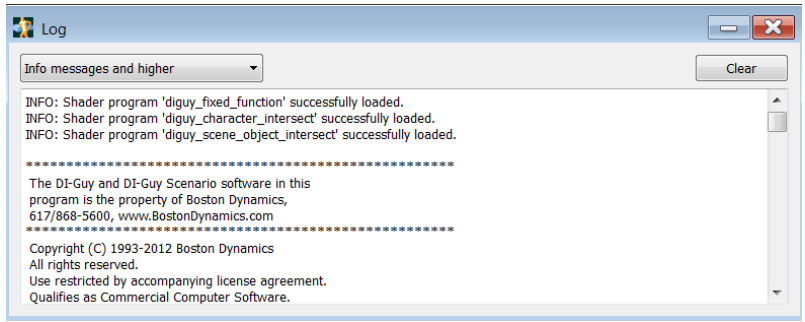


The Movie Settings palette supports the easy creation of high-quality digital videos straight from DI-Guy Scenario! Movies are created in a number of user-selectable formats. The only downside is that DI-Guy movies do not include audio. Users wishing to add audio will need to import DI-Guy movies and/or the underlying sequence of images into a video editing tool, such as Adobe Premiere, where they can add the audio, perhaps recorded separately from DI-Guy Scenario using 3rdparty audio software such as Audacity.

6.3 Log Palette

The Log Palette prints runtime log information. You can adjust the level of detail of information printed by setting the menu at the top of the palette.

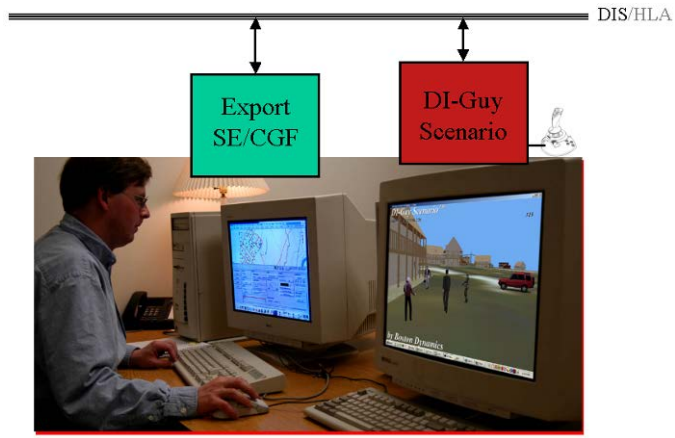
The Clear button clears out all previous log information and refreshes the window. Future log information will continue to be recorded, but previous log information is no longer available for viewing. The pull down tab in the center top of the palette allows you to set the amount and kind of data collected by the logging function.



Networking in DI-Guy Scenario

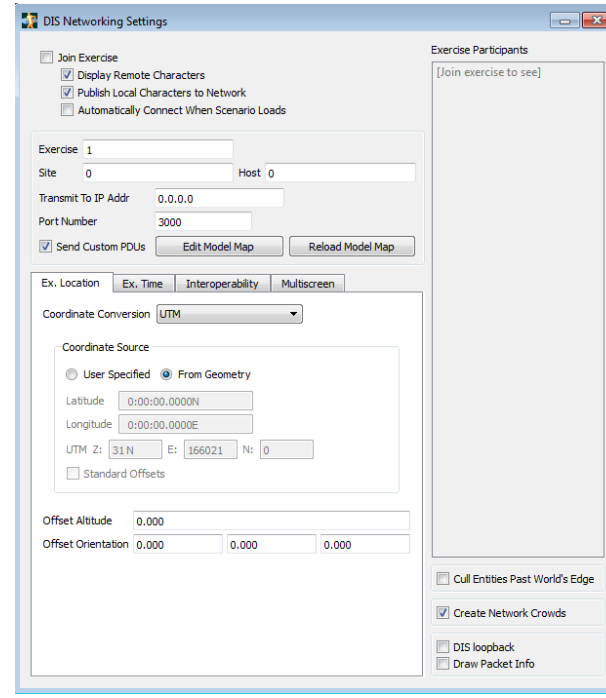
This section outlines the networking functions available in DI-Guy Scenario. Networking is an extra-cost option for DI-Guy Scenario. With the networking option, DI-Guy Scenario can use DIS or HLA to interoperate with other computers running DI-Guy Scenario and with third party applications such as OneSAF, VR-Forces, VBS2, etc.

DI-Guy Scenario interoperating with Export SE/CGF



PLEASE NOTE: Consult the DI-Guy SDK User Manual's Networking chapter for technical details of DI-Guy networking, custom PDUs, FOMS, etc.

To start networking, activate the Network Settings palette from DI-Guy Scenario's main pulldown menu or push the F11 key.



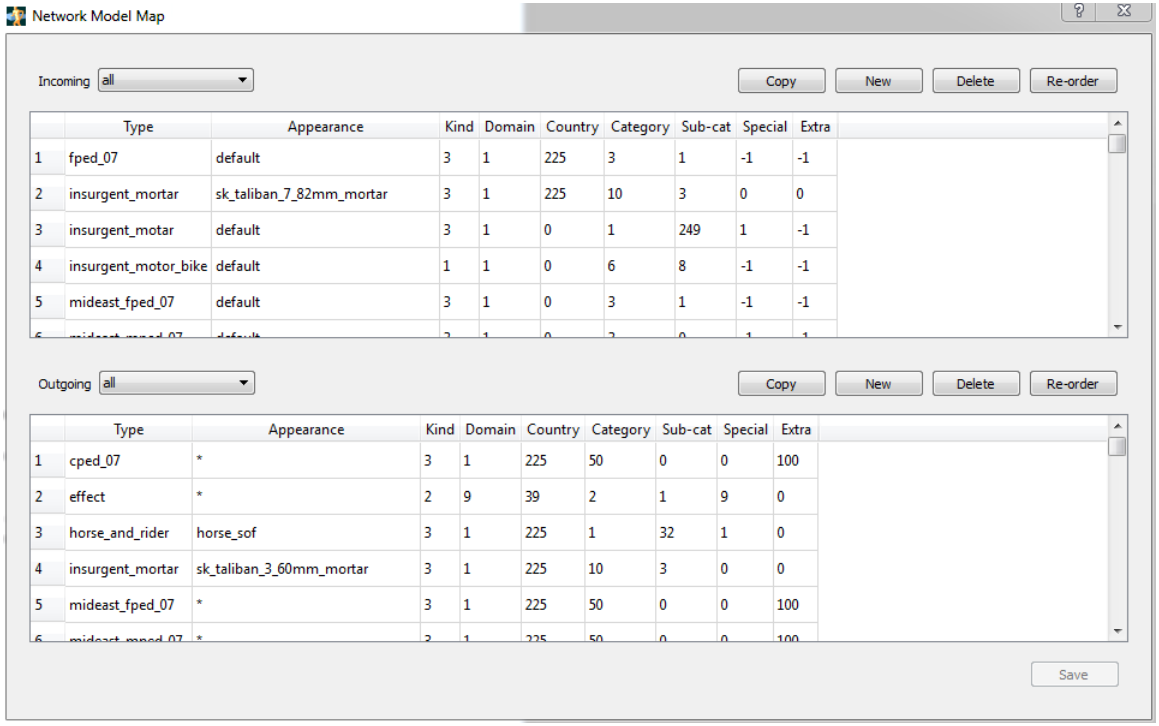
6.3.1 Enabling DIS Networking

Select an exercise by entering an exercise number in the "Exercise" field of the GUI. To join the exercise, check the "Join Exercise" box at the upper left. If you check the *Display Remote Characters* box, DI-Guy Scenario will display remotely simulated entities. If you check the *Publish Local Characters to Network* box, DI-Guy Scenario will broadcast its own locally simulated entities. 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 the network.cfg file found in the %DIGUY%/config/diguy/scenario directory. See the SDK Manual for more information

6.3.2 DI-Guy Scenario Entity Mapping

DIS uses “septets”, that is, an array of seven integer values, to describe entities. The DI-Guy network model map describes how DI-Guy character types and appearances are mapped to/from the DIS septet. Select “Edit Model Map” to bring up the GUI shown below:



The Network Model Map Editor (accessible from Network Settings) allows you to configure these mapping.The default settings can be found in the configuration file network_model_map.cfg, located in $DIGUY\config\diguy\. Any changes made will be saved to network_model_map.cfg in $DIGUY_CUSTOM\config\diguy\ (note distinction from above), which overrides the defaults.

In the map of remotely simulated entity types to DI-Guy 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.

6.3.3 DI-Guy Custom PDUs

Please consult the DI-Guy SDK User Guide for detailed information on how DI-Guy Custom PDUs work and how they can be constructed or read. Note that DI-Guy Custom PDUs can be used to control DI-Guy Scenario, for example, they can cause DI-Guy Scenario to load a scenario, start, stop, play a script, etc.

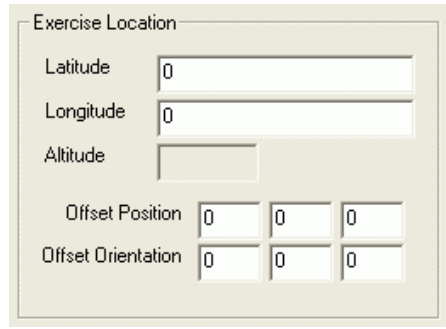
6.4 HLA 1.3 and 1516 Support in DI-Guy Scenario

The user interface for HLA is similar to the DIS interface described above. For information on configuring your RTI, RID, FED, and other HLA information, please consult the DI-Guy SDK User manual's networking chapter. The command-line argument, -net_hla, must be given to DI-Guy Scenario at startup, by default, DI-Guy Scenario networking uses DIS.

6.4.1 RPR-FOM elements and DI-Guy FOM Extensions

Please consult the DI-Guy SDK User Guide for information on attributes and elements used by DI-Guy Scenario, as well as information on the DI-Guy FOM Extensions for HLA.

6.5 Positioning the Terrain Database on Earth's Surface



DI-Guy Scenarios works in database coordinates. When DI-Guy Scenario communicates with a simulation via DIS or HLA, the locations of entities are specified to the network in geocentric coordinates. Using DI-Guy Scenario, you will not need to deal directly with geocentric coordinates, but you need to know how to situate your database on the surface of the Earth. You do so using the Exercise Location panel on the Networking property tab. DI-Guy Scenario supports UTM, flat earth, and geocentric.

6.5.1 The Exercise Location Panel

The Exercise Location panel allows you to specify where on the Earth your database is positioned. By default, the origin of your database is placed at zero degrees longitude, zero degrees latitude and at sea-level, according to the WGS1984 ellipsoid representation of the earth. Specify other latitude and longitude values to move the origin of your terrain database accordingly. 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.

Fine positioning of your database relative to the specified latitude and longitude is done using the Offset Position boxes. Offset Position box values are expressed in meters. To move your database to the east, increase the value in the first Offset Position box. To move your database north, increase the value in the second box. To move it to higher altitude, increase the value in the third box.

Once the location of the origin of your database is fully specified as above, you can then rotate the database about its origin, if desired. To rotate about the positive Z axis, specify the rotation in degrees in the first Offset Orientation box. To rotate about the positive X axis, specify the rotation in degrees in the second Offset Orientation box. To rotate the database about the positive Y axis, specify the rotation in degrees in the third Offset Orientation box.

6.5.2 UTM and the Exercise Location Panel

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.

6.5.3 Displaying Terrain Coordinates in DI-Guy Scenario

You may find it useful to quickly determine the coordinates of a visible object in the terrain database. DI-Guy Scenario allows you to display the coordinates of the object located under the cursor. To turn on display of coordinates, select Preferences from the Edit menu and check the box labeled "Show cursor position", then click the OK button. The heading of the camera will be displayed in the upper right corner of the display. The coordinates of the object the mouse cursor points at will be displayed in the upper-left corner of the 3D window. You can change the format displayed by pressing the G key. Each time you press G the format changes. 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 Exercise Location panel must identify the center of the UTM zone. UTM zones are numbered from 1 to 60. You can compute the latitude and longitude of the center of 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 database origin is in northern hemisphere.
Y offset = Northing – 10000000	IF database origin is in 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.

8. Miscellaneous

8.1 Complete Vehicles

Support for vehicle characters in DI-Guy has increased through each DI-Guy version. Some vehicles had articulated wheels, others doors that open and close, and yet others with aim-able turrets. No single vehicle character had all of these capabilities. DI-Guy 11 introduces more complete vehicle characters that address this. The first complete vehicle character is "vehicle_09".

Complete vehicle character types have the following capabilities:

- Rolling and turning wheels.
- Smooth acceleration and deceleration.
- Doors that open and close.
- Body that can rock.
- Aim-able turret.

Rolling and Turning Wheels

Wheels on complete vehicles roll and turn appropriately as the vehicle moves through a scenario, whether the vehicle on a path or in free action mode; nothing special needs to be done to enable this.

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's 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

Previous DI-Guy vehicles had very discrete speeds available to them. They could move slowly and move fast, but the transition between these speeds was abrupt. Complete vehicles add the ability to have smooth accelerations and decelerations between speeds.

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 will result in faster accelerations, while having them further away will result in more gradual accelerations.

Note that if change_speed actions are not inserted between moving and still actions, the vehicle will do abrupt speed changes as was the case for earlier vehicle characters.

Doors That Open and Close

Complete vehicles have doors that can open and close. They can do this independently of the vehicle's current action (moving forward, sitting still, etc.). Each door can be independently controlled.

Doors are controlled through the use of gestures. Gestures in DI-Guy 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, 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 via the API, scripting, and decision beads.

Some currently available vehicle gestures:

- ”open_doors_all”
- ”open_door_front_left”
- ”open_door_front_right”
- ”open_door_rear_left”
- ”open_door_front_right”
- ”rock_body_left_right” (described below)

Body Rocking
Similar to the opening of its doors, the vehicle can also be rocked on its chassis through the use of a vehicle gesture. This can be used, for example, to show a crowd attacking a vehicle.

Aim-able Turret
For vehicle appearances that have turrets (technicals, turreted humvees, etc.), 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 via API and scripting.

Example Scenario
There is an example scenario, Vehicle_Drop_Off.dss, that demonstrates all of the above capabilities.

8.2 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 your application using an embedded DI-Guy.

Maximum Display Frame Rate – *Frame dt*
DI-Guy Scenario allows you to specify an upper bound on the frequency the 3D window is redrawn. The parameter, *Frame_dt*, is set on the Preferences menu. The default setting is 30 fps. If you want more CPU time to go to other processes on your computer, reduce the value.

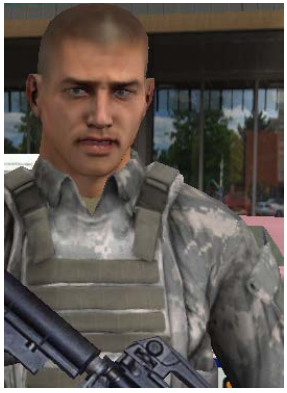
Simulation Update Rate

DI-Guy Scenario calculates behavior at regular intervals for each character. The interval is called *Tick_dt*. Every Tick_dt, DI-Guy Scenario calculates the motion of the characters and vehicles, evaluates active decision beads, and executes script events. Larger values of tick_dt result in less computing. The default value of tick_dt is 0.03125 seconds. You can adjust tick_dt by setting its value on the Performance tab of the Scenario Objects palette.

Bead Update Rate

Another way you can reduce the amount of calculation performed by DI-Guy Scenario is to evaluate decision, script, aim and gaze beads less frequently than every tick of the DI-Guy Scenario clock. Decision and script beads are particularly expensive. The fields *Default Ticks Between Checks* on the Performance tab of the Scenario Objects palette allows you to specify how many ticks to skip between evaluations for each type of bead.

8.3 Expressive Faces



Expressive Faces are an optional module available for DI-Guy. 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 to initialize and deinitialize the expressive face module and have a valid expressive faces license.

DI-Guy Expresive Faces has recently been upgraded to FaceFX technology, documentation is pending. For more information, contact diguy@diguy.com.

8.4 Customizing DI-Guy Scenario (Content)

DI-Guy Scenario 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, users can add their own motions, models, and textures DI-Guy Scenario. Consult the DI-Guy Motion Editor and/or DI-Guy Content Guide for information on how to add new content to DI-Guy Scenario.

8.5 Customizing DI-Guy Scenario (Plug-Ins)

Users can create plug-in DLLs that can work with DI-Guy Scenario at runtime. These can be useful for creating interfaces to custom hardware or adding intelligence to scenarios. Plugins require a separate license. Plugins should be compiled using the same compiler as DI-Guy Scenario. There is a detailed programming example found in `%DIGUY%/programming_examples/diguy_plugin.`

Loading a DI-Guy Scenario Plug-in

```
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
```

Plug-in DLLs should be loaded by specifying the plugin using command-line argument when launching DI-Guy Scenario. Plugins can also be loaded using an initialization script. Use the Script tab on the Event Handlers palette to initialize a plug-in this way. The following Lua code shows how one could then initialize the DLL:

- 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 plugins.

Supported Plug-in functions

The following list of functions can be exported from a DLL plugin and will be recognized by DI-Guy. You do not need to implement all the functions, just the ones you need. DI-Guy Scenario will automatically call these functions at the appropriate time. Plug-in functions should be declared with __declspec(dllexport) to make sure that the symbols are properly exported from the DLL. See the plugin example in the DI-Guy OpenGL programming examples folder for an example of how to build a plugin that will work with DI-Guy Scenario.

- void initialize_plugin(diguyApp *app)
- void deinitialize_plugin(diguyApp *app)
- void initialize(diguyScenario *s)
- void deinitialize(diguyScenario *s)
- void pre_draw(diguyScenario *s)
- void post_draw(diguyScenario *s)
- void pre_update(diguyScenario *s)
- void post_update(diguyScenario *s)
- void reset(diguyScenario *s)
- void save_review_data(const char* filename)
- void load_review_data(const char* filename)
- void mouse_func(diguyScenario * s, int x, int y,int down);

void initialize_plugin(diguyApp *app)

This function is called once when the plugin is being loaded. The call will happen while DI−Guy Scenario is starting up if the plugin was specified on the command line, or during the `load_plugin()` call if the plugin is being explicitly loaded by function call.

void deinitialize_plugin(diguyApp *app)

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

void initialize(diguyScenario* scenario)

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

void deinitialize(diguyScenario* scenario)

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

pre_draw(diguyScenario *scenario)

This code is called once per frame before objects are drawn. Only code related to drawing should be here. Use Update for any other code you wish to be called frequently.

post_draw(diguyScenario *scenario)

This code is called once per frame after all other objects have been drawn, it's a good place to draw overlays or transparent objects.

NOTE: Both draw functions work because they are called in a thread with a valid OpenGL context. Any OpenGL call should work, and querying and modifying the current OpenGL state should also work. It's the user's responsibility to make sure the OpenGL state is restored to what it was at the beginning of the function.

pre_update(diguyScenario *scenario)

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

post_update(diguyScenario *scenario)

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

> You can use functions such as: scenario->get_replaying_history() and scenario->get_t() to branch whether you are replaying history or you are generating new data at runtime. It's up to the user to decide what to do for each case.

pdate is called once per frame update.

reset(diguyApp *app)

Reset is called whenever the scenario is reset, if the plug in has its own history this may be an appropriate time to clear it.

diguyPluginMouseCursor get_mouse_cursor(diguyScenario* s)

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

void mouse_press(diguyScenario* s, diguyPluginMouseEvent* mouse_event)

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

void mouse_drag(diguyScenario* s, diguyPluginMouseEvent* mouse_event)

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

void mouse_release(diguyScenario* s, diguyPluginMouseEvent* mouse_event)

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

void mouse_move(diguyScenario* s, diguyPluginMouseEvent* mouse_event)

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

A key feature of DI-Guy Scenario is the ability to save and restore the history of a scenario. Plug-ins can supplement this feature using the review_data functions. The filename argument is the base name of the history review files. It is the programmer's job to write the serialization and deserialization code.

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

This function will be called when after−action review data for the scenario is saved by a call to `diguyScenario::save_review_data()`.

Example:

```
save_review_data(const char* filename)
{
char buffer[MAX_PATH];
sprintf(buffer, "%s.myplugin", filename);
FILE * file = fopen(buffer,"w+b");
assert(file);
.
.
.
fclose(file);
}
```

See the diguyApp documention in the DI-Guy SDK Reference Manual for more detailed information on plugin functions.

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

This function will be called when after−action review data for the scenario is loaded by a call to `diguyScenario::load()`.

8.6 Activating Switches and DOFs in Scene Objects

DI-Guy Scenario supports *Scene Object Behavior* files that operate switches and DOF that were 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 your scenario, you need to edit your DI-Guy Scenario scenario file (.dss). Add the following lines under the relevant scene object:

```
behavior_enabled = 1
behavior_filename = <my_behavior_filename>
```

Format for Scene Object Behavior File

Lines in the Scene Object Behavior File that start with the key words:

```
switch
dof
dof_flt
```

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.

Switches

The switch data format is as follows: time switch_name active_state.

Example: Here are possible lines in the behavior file which represent a switch named *s1* that is turned off at at 7 seconds. We first change the data format to switch mode, and then specify a switch change.

```
switch
7.0 s1 0
```

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 will enter 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 will be 0 and 1, respectively.

DOFs

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]`. `Time` and `dof_name` are the same as for switches. Arguments in square brackets are optional. If any of these arguments is specified, it must be labeled with `x`, `y`, etc.

Example: Here are possible lines in the behavior file which represent a dof displaced 3 units in the y direction and rotated 30 degrees around the yaw-axis (at 7 seconds). We first change the data format to dof mode.

```
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 thus: yaw represents rotation around the z-axis, roll around the x-axis, and pitch around the y-axis. All rotations of the DOF will be done in zxy (Boston Dynamics) order.

Using the keyword `dof_flt` indicates that the data following will be understood thus: yaw represents rotation around the z-axis, roll around the y-axis, and pitch around the x-axis. All rotations of the DOF will be done in xyz (MultiGen) 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.

Example File

```
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
```

8.7 Hotkeys

Many DI-Guy Scenario functions can be invoked using keyboard hotkeys. The following table lists the hotkey functions.

Hotkey	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
<CTL>	Temporarily switch to Insert Mode
<SHFT>	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
a	Toggle script control of camera
1 thru 0	Load camera settings
<SHFT> 1 thru 0	Save camera to camera settings slot
Palettes	
q	Toggle display of Character Elements palette
w	Toggle display of Scenario Objects palette
e	Toggle display of Camera View palette
r	Toggle display of Signal palette
t	Toggle display of Action Spreadsheet
l	Toggle display of Log window
i	Toggle display of Mode palette
Misc	
g	Cycle through units displayed by pointing 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 Hotkeys	
<CTRL>x	cut
<CTRL>c	copy
<CTRL>v	paste
<CTRL>z	undo
<CTRL>q	quit
<CTRL>s	file save
<CTRL>a	file save as
<CTRL>o	file open
<CTRL>w	file close

8.8 Troubleshooting

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 Waypoint tab of the Character Elements palette 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?

If there is no current path, neither waypoints nor action beads can be created, modified, or deleted. Make sure there is a current path on the Path tab.

Why does my character disappear when it reaches the end of its path?

Check the dead space for the current path on the Path tab of the Character Elements palette. 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 next path?

1) Check the dead space for the current path on the Path tab. If the dead space value is positive, the character will jump to the beginning of the next path.
2) Check to make sure the first waypoint of the next path is in the same position as the last waypoint of the preceding path.

8.9 Glossary

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. An action bead can tell a person to start walking, to stand for a while, or to put its hands in the air.

Aim Bead	An aim bead sets the azimuth and elevation of the weapon of a person who is aiming.
Bead	A bead is a graphical symbol marker associated with a person and a path. Beads can be positioned anywhere along a path, but can not leave the path. There are 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.
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.
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.
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.
Path	People travel along paths to move through scenarios. Paths are spline-shaped functions defined by waypoints. You can define, edit, and shape paths using the DI-Guy Scenario 3D user interface.
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.
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.
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.
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.