DI-Guy Content

Reference Manual

VT MÄK
A company of VT Systems

DI-Guy Content

Reference Manual

VT MÄK
A company of VT Systems

For third-party license information, please see “Third Party Licenses,” on page xi.

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

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

info@mak.com

www.mak.com

Revision DIG-13.0-3-140523

Contents

Preface

Chapter 1. DI-Guy Content

Chapter 2. Tutorials

Chapter 3. Creating a Custom Character Type

Chapter 4. Creating An Expressive Faces Head Appearance

Chapter 5. Using the Character Viewer

Index

Preface

This manual is written for persons who want to add new characters to use with DI-Guy.

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

How This Manual is Organized

The chapters are organized as follows:

Chapter 1, *DI-Guy Content*, briefly describes DI-Guy and its features.

Chapter 2, *Tutorials*, guides you through several ways to add content to DI-Guy.

Chapter 3, *Creating a Custom Character Type*, explains how to add a character that does not have a similar character already in DI-Guy.

Chapter 4, *Creating An Expressive Faces Head Appearance*, explains how to add an Expressive Faces head using FaceFX.

Chapter 5, *Using the Character Viewer*, explains how to view characters, appearances, and actions in the Character Viewer. It also describes the Geometry Viewer.

Documentation

DI-Guy comes with the following documentation:

- *DI-Guy SDK Developers Guide*. Explains how to program using the DI-Guy SDK.
- *DI-Guy SDK Reference Manual*. Online documentation that lists all the functions and classes of DI-Guy SDK.
- *DI-Guy Scenario Users Guide*. Explains how to use DI-Guy Scenario to create scenarios with realistic human activity.
- *DI-Guy Content Reference Manual*. Explains how to customize DI-Guy characters and add new character models.
- *DI-Guy AI Users Guide*. Explains how to use DI-Guy AI to add intelligent behavior to DI-Guy characters.
- *DI-Guy Motion Editor Users Guide*. Explains how to import, edit, and customize DI-Guy motions.
- *DI-Guy Expressive Faces Users Guide*. Provides a brief introduction to using the Expressive Faces module.
- *DI-Guy Master Index*. A combined index for all of the manuals. It does not index the HTML documentation.
- DI-Guy Digest. An XML file that explains DI-Guy content to non-DI-Guy applications.

These documents are available from the start menu under DI-Guy, or by browsing to *./doc*. Of particular interest is the DI-Guy Documentation Master Index, which lets you quickly access all the documents above along with additional technical information about DI-Guy. (This is the entry point to HTML documentation. It is not the same things as *DI-Guy Master Index*.

Derivative Work

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

MÄK Products

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

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

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

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

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

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

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

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

How to Contact Us

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

Telephone

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

E-mail

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

Internet

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

Post

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

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

Document Conventions

This manual uses the following typographic conventions:

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

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

DI-Guy Conventions

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

Table 2-1: Coordinate system conventions

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

Mouse Button Naming Conventions

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

Third Party Licenses

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

Boost License

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

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

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

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

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

libXML and libICONV

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

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

Lua

Some MÄK products use the Lua programming language (www.lua.org). Its license is as follows:

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

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

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

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

1. DI-Guy Content

This chapter is an introduction to customizing DI-Guy content.

1.1. Customizing DI-Guy Content

DI-Guy is highly customizable. Motions, geometry, textures, characters, and special effects (that is, DI-Guy content) can all be customized. Be sure to use the *./custom* directory, as described in this manual, when modifying DI-Guy. This will safeguard your changes so that you will be able to install future releases without losing your customizations.

When editing configuration files, do not use Notepad. Use a text editor that preserves end-of-line information.

1.1.1. Encrypt Your Final Customized Geometry

When customizing *.flt* and *.dae* geometry files, always encrypt your geometry to protect DI-Guy intellectual property. This is required by the DI-Guy Model Use Agreement. To encrypt your customized geometry, open up a command prompt and type:

```
diguyEncryptGeometryFile filename
```

1.1.2. Where to Place Customized Files

When DI-Guy needs to access data, it looks in the *./custom* directory before looking in the default location for the file. The *./custom* directory acts as a shadow or mask directory to the default DI-Guy content directories. Therefore the files in it should use the exact same directory layout as the default directories. For example, if DI-Guy normally loads *./geometry/flt/model.flt*, it will first search for the file *./custom/geometry/flt/model.flt* before loading *./geometry/flt/model.flt*.

It is strongly recommended that you leave all files installed by DI-Guy unchanged and place all your modified and new files in *./custom/*. This will allow your customizations to be portable to other DI-Guy installs and allow you to upgrade to new versions of DI-Guy with minimal or no changes to your custom content.

1.1.3. The DI-Guy Data Directory Structure

Here is a list of custom directories whose files mask identically named files in the standard directory:

- Geometry *./custom/geometry/*
  - Textures *./custom/geometry/rgb/*, *./custom/geometry/png/*
  - Scene Objects *./custom/geometry/sets/*
  - OpenFlight Models *./custom/geometry/flt/*
  - Collada Files *./custom/geometry/dae/*
- Configuration Files *./custom/config/*
- Motions *./custom/motions/*
- Sound Effects *./custom/sfx/*
- Scenarios *./custom/scenarios/*

1.1.4. Understanding DI-Guy Configuration Files

The content available in DI-Guy is declared in and loaded using configuration files, allowing DI-Guy to be easily customized. By default, the configuration file *./config/diguy.cfg* is used by DI-Guy to define content. This file `includes` other configurations files, which also may `include` more nested configuration files, and so on.

If you want to better understand how DI-Guy organizes its data, we encourage you to examine, trace through, and modify files using the custom directory shadow technique. All of the configuration files are in ASCII text format, so they are easy to read and understand.

The following concepts and terminology are used in the DI-Guy configuration file system:

- Actor. The flesh-and-blood actor whose performance was captured using motion capture. Motion files reference the original actor. The DI-Guy motion engine often performs mathematics to scale motions to fit the end dimensions of the virtual character's appearance.
- Appearance. The specific geometry and textures that describe the visual look of the character. DI-Guy appearances can be skinned (deformable mesh) or segmented (rigid interpenetrated polyhedral). Many skinned appearances include equipment that may be segmented.
- Character_type. The combination of the set of actions a character can perform with the set of visual appearances the character can use.
- Joint. Describes the way two rigid links are joined together. Popular joints include pin, frozen, 6DOF (degrees-of-freedom), ball, 3DOF, and slider.
- Link. A link is a joint plus an offset. Often shapes are associated with links. Links can be traversed and drawn. Sometimes links are referred to as "bones".
- Shape. A mesh or polyhedral associated with a link. Unlike the link, which is conceptual in nature, shapes are actual drawables.
- Shapeset. A collection of one or more shapes.

For definitions of other terms used by DI-Guy, please see *DI-Guy Glossary*, in *DI-Guy Scenario Users Guide*.

The entries in *diguy.cfg* can be organized into the following sections:

- Preface. This section defines very basic things about the content.
- Main Section. This section defines the bulk of DI-Guy content, often by pointing at lists of Level Two include file content.
- Level Two File. Common files using common syntax to describe DI-Guy content.

Paths in *diguy.cfg* are relative to the *./config* directory.

The Preface Section

This section of *diguy.cfg* defines very basic things. It is not usually modified except by very advanced DI-Guy users.

- geometry_set_priority_list. Order of operation for geometry file search.
- data_version. Specifies the DI-Guy version of the data, include the configuration files themselves.
- required_sdk_version. Specifies the version of DI-Guy software that this data is compatible with.
- *common/loaders.cfg*. Motion data loaders. Specifies the exact name of every variable for a particular DI-Guy character_type. Each loader has a name, and that name is referenced in the character_type definitions.
- *common/shader_matrix_index_tables.cfg*. A list of different object types and bones that a DAE file has, needed to map a skinned character to a DI-Guy skeleton.
- *common/common_links.cfg*. The variables for popular links, as well as the joint type and some skeletal hierarchical information.
- *common/shape_and_joint_data.cfg*. The name of a shape is joined to the name of the joint. Therefore you can determine what shapes will be drawn when a joint is drawn.
- *common/common_actors.cfg*. The actor object is described by its shapesets and links.
- *common/common_characters.cfg*. Various DI-Guy primitive characters, such as lines, are described in this file. Character definitions usually have a graphics_class, actor, link list, and default_equipment.
- *diguy/required_characters.cfg*. Defines the character waypoint.

The Main Section

This section of *diguy.cfg* defines most of the DI-Guy content. If you want to customize DI-Guy content, you need to understand this section and the Level Two content it references.

- *diguy/diguy_actors.cfg*. Lists individual actor configuration files. See *diguy/actor_<actor_name>.cfg* for more information.
- *diguy/diguy_characters.cfg*. Contains a list of character_type definitions. See the *diguy/char_<character_type>.cfg* definition for more information. The file also includes *diguy/diguy_character_type_maps.cfg*, which associates a single word (for example, soldier) with a character_type and appearance combination to support generic requests for a type of character where DI-Guy responds with the best match.
- *diguy/character_types_list.cfg*. Defines the set of character_types that match simple descriptions. For example, more than a dozen character types are deemed relevant when querying which character_types are available as male pedestrians. Like *diguy/diguy_character_type_maps.cfg*, this list supports wizard-type queries into the DI-Guy content database for best matches to user search criteria.
- *diguy/exface_shapesets.cfg*. Defines expressive face appearances and geometry available to particular actors.
- *diguy/gestures.cfg*. Defines lists of gesture motions available.
- *diguy/motex_arcs.cfg*. Defines a list of available motion textures, used to add visual noise to character motions.
- *diguy/default_particle_descriptions.cfg*. Defines default particles and the parameterizations that define them.
- *diguy/appearances.cfg*. Lists of DI-Guy appearance category include files. See *diguy/appearances_<appearance_category>.cfg* for more information.
- *diguy/default_appearance_effects.cfg*. Appearance effects are particle effects that augment appearances, particularly of DI-Guy vehicles. Appearance effects also are used to support incoming DIS/HLA designations such as “smoking” or “on fire”.

The Level Two Files Section

- *diguy/actor_<actor_name>.cfg*. Each actor configuration file defines the shapesets available to the character, the links that compose the actor, and an actor definition. The actor definition includes the standing_hip_height used to scale actor motions to end appearance dimensions. The actor definition also includes *common/links_<link category>.cfg*, which tells DI-Guy what variables will be found in the motion file. Finally, the actor definition includes a shader_matrix_index_table assignment.
- *diguy/actor_<actor_name>_shapesets.cfg*. Defines the geometry and textures that will be made available to character appearances that are supported by the actor.
- *diguy/actor_<actor_name>_links.cfg*. Defines the skeletal hierarchy of the actor as a set of links, each with a name, offset, associated joint, and DOF variable names.
- *diguy/appearances_<appearance_category>.cfg*. Defines an appearance. An appearance consists of its name, the actor with which is associated, a number of "items" which designate shapesets used by this appearance, and information that helps categorize the appearance for wizard-like queries.

1.1.5. Autoload Configuration Files

Any files placed in the *./custom/config* directory will mask their counterpart in the *./config* directory.

In addition, any configuration file in *./custom/config/* that begins with autoload_ will automatically be read during DI-Guy configuration initialization.

1.2. Customizing Scene Objects

To add custom scene objects (terrains), copy the OpenFlight format database to *./custom/geometry/sets*.

1.3. Customizing Motion

You can customize motion using the DI-Guy Motion Editor. Please see *DI-Guy Motion Editor Users Guide* for details.

1.4. Customizing Texture

To customize textures, edit an existing texture and save it with the same name in *./custom/geometry/rgb* or *./custom/geometry/png*.

Textures use the RGB, RGBA, and PNG format.

Alternatively, you may want to create a new appearance that uses your new texture instead of overriding the existing appearance.

1.5. Customizing Sound

To customize sounds, edit an existing sound file and save it with the same name in *./custom/sfx*. You can also add new sounds to this directory.

2. Tutorials

This chapter has examples of customizing DI-Guy content.

2.1. Customizing Appearances Using Texture Swaps

Perhaps the easiest and most popular customization is to simply take an existing DI-Guy appearance and tweak its texture file without modifying geometry to create a new appearance. Uniforms can be changed to be specific to a country; clothing can match a target culture, sensorized appearances can be created to simulate UV or infrared visualization, and so on.

DI-Guy uses texture swapping to add variety to its character appearances while keeping the memory footprint small. Here are two mped_07 appearances, male_suit_1 and male_suit_2. Note how the polygons of the bodies are the same but the texture maps make it appear that the men are wearing different jackets, ties, and pants. The following sections guide you through the process of creating a new appearance.

Figure 2-1. Texture swapping to create different appearances

2.1.1. Select an Appearance to Modify

This tutorial references *.dae* files. The *.dae* files are in the DI-Guy CD3 installer. If you did not install CD3, please do so before trying out this tutorial.

1. On the Start menu, choose **All Programs** → **DI-Guy Applications 13** → **DI-Guy Character Viewer**. The Character Viewer opens.
2. In the Character Type list, select mped_07.
3. In the Appearance list, select sk_male_suit_1.
4. Hold down the left and right mouse buttons, and rotate the character so that it faces you (Figure 2-2).

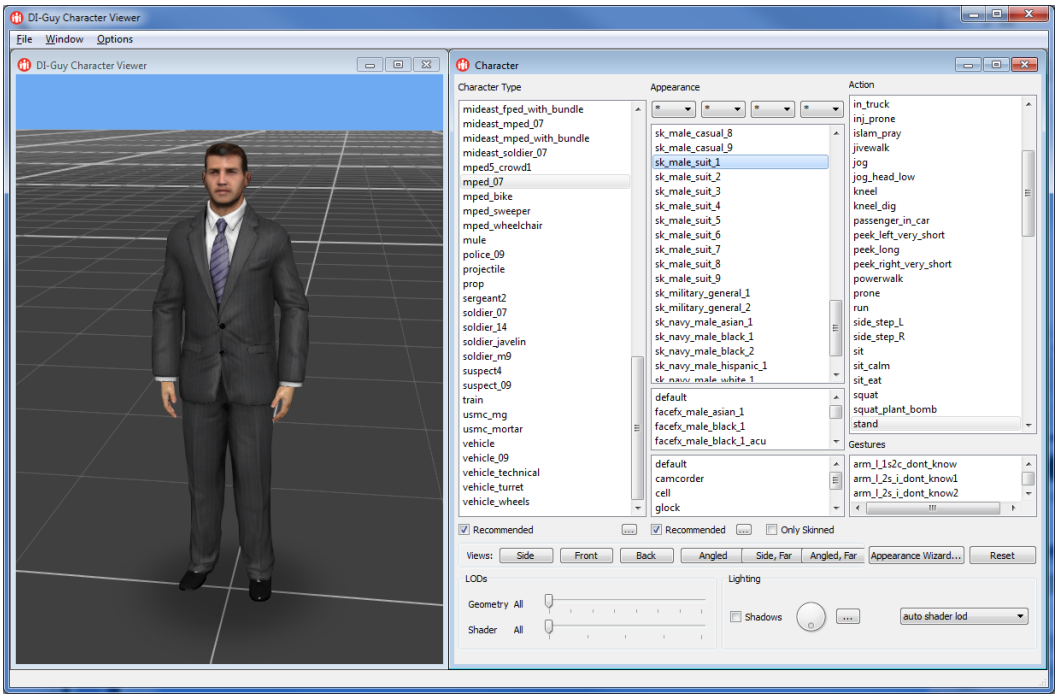

Figure 2-2. Character viewer

5. Select the sk_male_suit_2 appearance to see the differences.
6. Choose **Window** → **Geometry Viewer**. The Geometry Overview window opens (Figure 2-3).

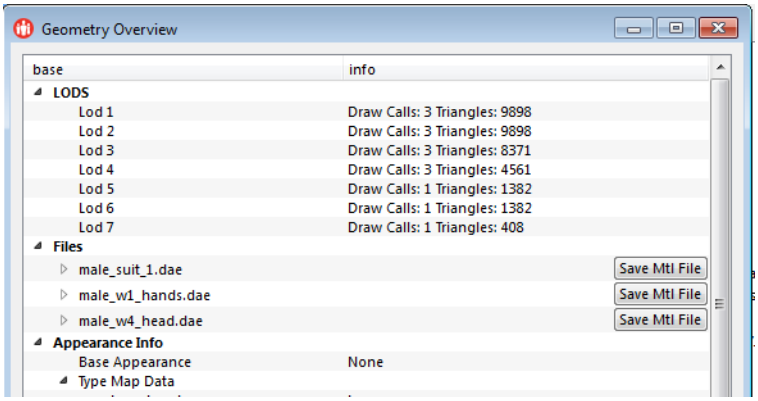

Figure 2-3. Geometry Overview window

The Files section lists three files. Their names indicate that they are for the suit, hands, and head. For our customization, we will make a simple change to male_suit_1. First we need to find the texture map that male_suit_1 uses.

7. Expand the *male_suit_1.dae* entry and then expand the textures entry. You can see there are many textures associated with the character. The textures in bold indicate textures already loaded into memory. In the image, *male_suit_2_diffuse.png* and *male_suit_5_diffuse.png* are also highlighted because they had been visualized when you switched appearances. The naming conventions indicate how the textures are used. The convention _diffuse is used for the diffuse texture, _normals for the normal map, and _specular for the specular map.

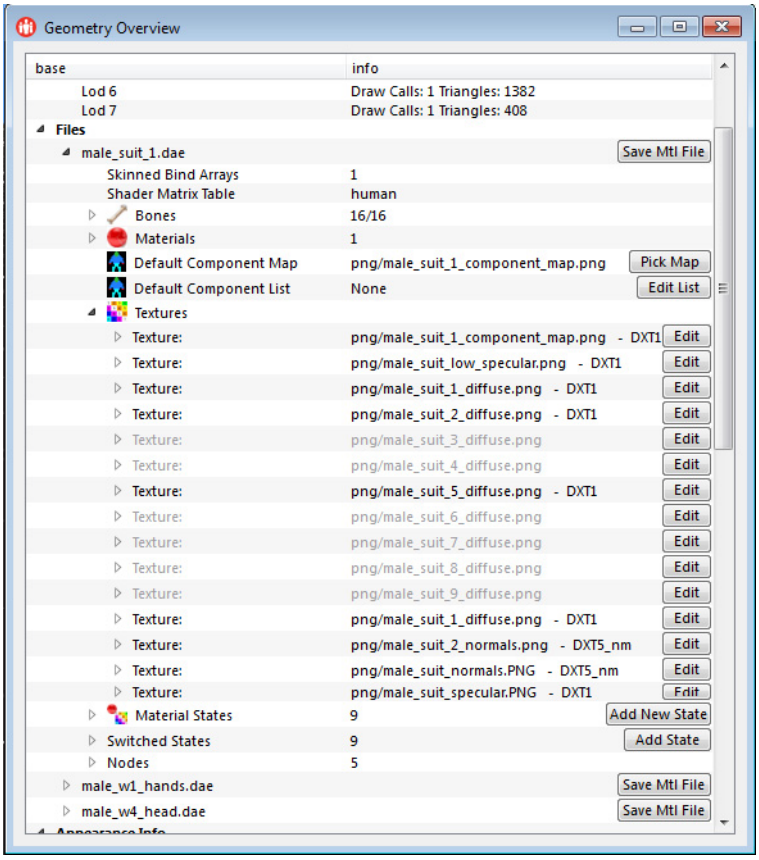

Figure 2-4. male_suit_1 textures

8. We need to make a copy of the male_suit_1 configuration file. However, to do that, DI-Guy requires that the *.dae* file be in the custom directory.

   a. In *./custom/geometry*, create a new directory called *dae*.

   b. Copy *./geometry/dae/male_suit_1.dae.enc* to *./custom/geometry/dae/male_suit_1.dae.enc*.

9. In the Geometry Overview window, on the *male_sui_1.dae* line, click Save Mtl File. You are prompted to save to the custom directory and may receive informational prompts.

10. Click Yes. The file *./custom/geometry/dae/male_suit_1_mtl.cfg* gets created.

11. Use a text editor to open the file *male_suit_1_mtl.cfg*. Near the top of the file, are some texture entries. One of them is *male_suit_1_diffuse.png*. We will edit this file.

2.1.2. Change the Texture Map

To change the texture map, you need a paint program, such as Photoshop or Gimp.

1. In your paint program, open *./geometry/png/male_suit_1_diffuse.png* (Figure 2-5). The tie is in the upper right quadrant of the file.

Figure 2-5. *male_suit_1_diffuse.png*

2. Change the tie's stripes to yellow.

   Be careful not to change the topology. That is, do not move any of the individual images. The x,y coordinates of the image map to the u,v of the target mesh. Changing the topology would require a re-mapping.

3. Create a new directory *./custom /geometry/png*.

4. Save the texture as *male_suit_1_g_diffuse.png* in *./custom /geometry/png*.

2.1.3. Create Your New Texture Swap and Declare Your New Appearance

1. In the Geometry Overview, in the Material States line for male_suit_1.dae, click Add New State. The State Editor opens (Figure 2-6).

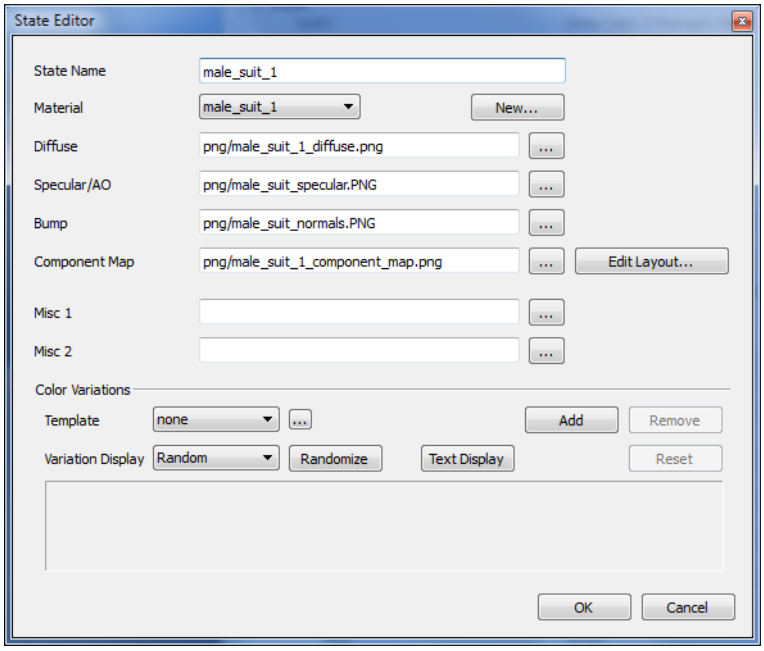

Figure 2-6. State Editor

2. Set the State Name to "male_suit_1_g" and browse *./custom /geometry/png/male_suit_1_g_diffuse.png*.
3. Click OK. The *male_suit_1.dae* is highlighted and has an asterisk.
4. Click Save Mtl File. You are prompted to save to the custom directory.
5. Click Yes.
6. Create a new appearance configuration entry for your character.
   a. In a text editor, open *./config/diguy/appearances_skinned.cfg*.
   b. Copy the multi-line entry for appearance sk_male_suit_2 to a new file.
   c. Save the new file as *./custom/config/autoload_my_custom_appearances.cfg*.
   d. Change the name of that entry to sk_male_suit_1_g.
   e. Change the head back to sk_male_w1_head (or leave it as is if you like).
   f. Change the graphics_state_switch_map1 to use male_suit_1_g.
   g. Delete the graphics_state_switch_map2 line.
7. Restart the Character Viewer. The new appearance should now be available with a beautiful new tie (Figure 2-7). Compare it to Figure 2-2.

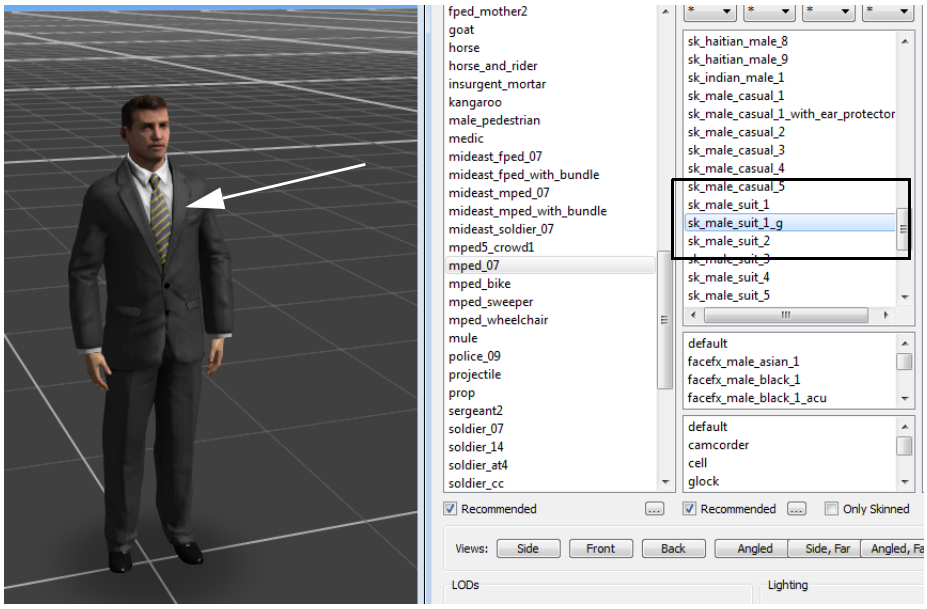

Figure 2-7. Character with modified tie

2.2. Customizing Appearances by Modifying Geometry

Swapping textures is an easy way to customize DI-Guy characters. However, often geometry needs to be changed to create the exact new appearance desired. The following three examples show, in increasing complexity, how to customize the geometry files of a prop, a vehicle, and a human. All the examples are done in the same basic way. Since they build in complexity, it is important to read how to customize a prop and vehicle if you want to customize a human.

The key to customizing is to find the declarations for a character's appearance that are closest to the new appearance you want to implement and then copy the form and structure for your custom declaration.

To complete this tutorial, you must have an application that can edit OpenFlight and MAX files and be familiar with editing them.

2.2.1. Creating a Custom Prop

A prop is defined as a DI-Guy character that cannot move. Prop appearances include:

- Road barrier
- Barbed wire fence
- Sofa
- Vending machine
- House.

Props in DI-Guy are built using OpenFlight or Collada (DAE) geometry. (DI-Guy developers use DAE for all new models.) This example assumes OpenFlight geometry.

Prepare Your Geometry File

An OpenFlight model defines each prop's geometry.

To add a new prop model to the prop library:

1. If necessary, convert your model from its existing format into the OpenFlight format. (Meshlab, http://meshlab.sourceforge.net/, is a good tool for doing this.)
2. Write out each LOD (up to 7) as a separate FLT file.
3. Edit your model so that there is a group at the root of the model hierarchy named "base".
4. Edit your model so that the positive Z axis points up and the positive X axis points forward. (Your prop should "look" down the X axis.)
5. Make sure your OpenFlight model specifies its texture files using relative addressing. Specify the location of the texture files as *../rgb/<texture_name>.rgb*.
6. Make sure the units in the OpenFlight model are correctly specified. (DI-Guy reads the OpenFlight units label to scale the props correctly. The preferred unit is meters.)
7. Set the OpenFlight LOD switch settings to: switch in = 1000, switch out = 0.
8. Save the file in *./custom/geometry/flt/*.
9. Save the texture files (.rgb, .rgba, .rgb.attr, .rgba.attr, .int) in: *./custom/geometry/rgb/*.

Customize the Configuration Entry

Now that the geometry is ready, it is time to customize the configuration entries so that DI-Guy knows about the new prop appearance. The system requires a configuration file that points to the geometry that defines the new prop at each level of detail.

1. Specify this file name as *./custom/config/autoload_actor_still_shapesets.cfg*.

   If this file already exists, modify it to add a shape_set entry for the new prop appearance. If the file does not exist, create a new one, using *./config/diguy/actor_still_shapesets.cfg* as a model of what to include.

   For example, to add a prop called "pink_flamingo" to the system, place the following block of text in the configuration file:

```
shape_set pink_flamingo
  actor = still
  is_base_shape = 1
  filename = pink_flamingo_most_detailed_LOD1.flt
  filename = pink_flamingo_LOD2.flt
  filename = pink_flamingo_LOD3.flt
  filename = pink_flamingo_LOD4.flt
  filename = pink_flamingo_LOD5.flt
  filename = pink_flamingo_LOD6.flt
 filename = pink_flamingo_least_detailed_LOD7.flt
  shape_and_joint = base position
```

DI-Guy expects an OpenFlight or Collada (DAE) file designation for each LOD. If you have fewer than seven LODs for your model, just use the same file for multiple LODS. LOD1 is the most detailed.

2. Add or edit *./custom/config/autoload_custom_appearances.cfg*.
3. Add an appearance entry for the new prop. If this file already exists, modify it to add an appearance entry for the new appearance. If the file does not exist, create a new one, using *./config/diguy/appearances_base.cfg* as a model of what to include:

```
appearance pink_flamingo
item = pink_flamingo
preload = false
character_type = prop
```

Your customization should now work.

1. In DI-Guy Character Viewer, select the character prop. Your new appearance should appear as a choice.
2. Select it and you should see your new prop appearance.

2.2.2. Creating a Custom Vehicle

DI-Guy comes with a variety of vehicle appearances that you can use in the scenarios you create. You can create custom vehicles using OpenFlight files or Collada DAE files. The preferred method, and the one used by DI-Guy for new models is to use DAE files. Each vehicle appearance is defined by a DAE file.

To add a new skinned vehicle appearance to DI-Guy:

1. Import/Merge the skinned model (*chevy_caprice_rigged.max*) into your new vehicle model.
2. Move/Scale/Rotate the new model into place. Be sure to Reset X Form so the model will not be offset when you export.
3. Add Skin Wrap Modifier to the new model.
4. Click Add.
5. Click the previously skinned model.
6. Click Convert To Skin.
7. Move bones into proper locations. DI-Guy vehicle skeletons support four doors and four wheels
8. Use Skin/Parameters/Edit Envelopes to skin wheels and doors.
9. Create LODs.
10. Export the model as a DAE.
11. Do appropriate configuration editing.
12. Load the new model in DI-Guy.

Creating a Custom Vehicle Using an OpenFlight Model

Each vehicle appearance is defined by an OpenFlight model or an encrypted Open-Flight model.

To add a new vehicle appearance to DI-Guy, perform the same steps as the prop example, and do the following:

1. Edit your model so that the positive Z axis points up and positive X points toward the front of the vehicle.
2. Place the origin of the vehicle at ground level in the center of all the wheels.

If you have a pre-existing model that faces in the y-forward direction, you can use that model "as is" by:

- Having your shape_and_joint entry in the shapeset definition use y_forward instead of position.
- Adding the entry `item = invisible_vehicle` as an additional item in your appearance definition.

3. Put your custom shapeset in *./custom/config/autoload_custom_vehicle_shapesets.cfg*.

   If this file already exists, modify it to add a shape_set entry for the new vehicle. If the file does not exist, create a new one, using *./config/diguy/actor_vehicle_shapesets.cfg* as a model of what to include. For example, to add a vehicle called "my_bmw" to the system, place the following block of text in the configuration file:

```
shape_set my_bmw
  actor = vehicle
  is_base_shape = 1
  filename = my_bmw_LOD1.flt
  filename = my_bmw_LOD2.flt
  filename = my_bmw_LOD3.flt
  filename = my_bmw_LOD4.flt
  filename = my_bmw_LOD5.flt
  filename = my_bmw_LOD6.flt
  filename = my_bmw_LOD7.flt
  shape_and_joint = base position
```

4. Create a custom appearance entry in *./custom/config/autoload_custom_appearances.cfg* as follows:

```
appearance my_bmw
  item = my_bmw
  preload = false
  character_type = vehicle
```

Your customization should now work. You can test it in DI-Guy Character Viewer.

1. In DI-Guy Character Viewer, select the character vehicle. Your new appearance should appear as a choice.
2. Select it and you should see your new prop appearance.

Any skeleton embedded in your OpenFlight character model is ignored by DI-Guy. DI-Guy will use the skeleton of the associated actor to piece together the model at runtime. So do not be confused if the character you view in Creator or another OpenFlight editing tool looks different from the assembled character shown by DI-Guy.

2.2.3. Creating a Custom Human: Skinned Characters Using Collada

DI-Guy's best looking and fastest performing characters are skinned. These characters have a flexible mesh, eliminating the seams seen in segmented characters while also reducing draw calls to the CPU. The following sections describe the process we recommend for creating a skinned custom character in DI-Guy. (However, it is not the only way to add skinned models to DI-Guy.)

Create (or Better Yet Copy) a Model that has a DI-Guy Compatible Skeleton

All DI-Guy actors are built Z up and X forward. Because the standard bones and biped in 3D Studio Max do not follow this convention, you need to be careful that your model uses a DI-Guy compatible skeleton. A sample skeleton is provided in *./geometry/max/greg_object_bones_skeleton.max*. Further information on the skeleton is provided in the next few sections to explain what is found in the MAX file.

Defining Links for a DI-Guy Compatible Skeleton

Links for a DI-Guy compatible skeleton should be positioned in world space based on the offsets listed in *./geometry/config/diguy/actor_greg_links.cfg*, except that all shapes are positioned directly from the origin (in other words, in world space) rather than starting from their parents' local coordinate systems. Figure 2-8 shows how the shapes should look in 3D Studio Max or another 3D editing program prior to setting up the hierarchy and locally rotating the arms and legs.

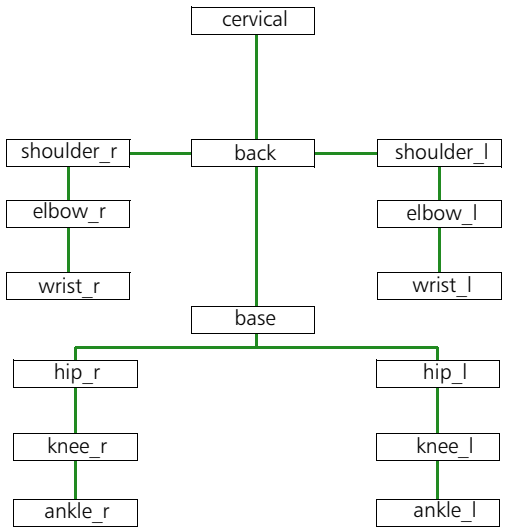

Figure 2-8. Shapes in 3D Studio Max

```
Actor Greg Skeleton in World Offset XYZ
  base          0, 0, 0
  back          0,0, 0.2254250
  cervical      0, 0, 0.650875
  shoulder_l    0, 0.1619250, 0.504825
  shoulder_r    0, -0.1619250, 0.504825
  elbow_l       0, 0.1619250, 0.2032
  elbow_r       0, -0.1619250, 0.2032
  wrist_l       0, 0.1619250, -0.053975
  wrist_r       0, 0.-1619250, -0.053975
  hip_l         0, 0.085725, 0
  hip_r         0, -0.085725, 0
  knee_l        0, 0.085725, -0.454025
  knee_r        0, -0.085725, -0.454025
  ankle_l       0, 0.085725, -0.885825
  ankle_r       0, -0.085725, -0.885825
```

Setting Up the Hierarchy of a DI-Guy Compatible Skeleton

The hierarchy links are set up in the Schematic View. Make sure the links are set starting from the child to the parent. DI-Guy requires the object names to match exactly the joint name convention listed in the previous section, so be sure to use those names and double-check your spelling.

Skin the Model in 3D Editing Software

Skinning refers to the process of associating polygonal vertices with one or more bones. DI-Guy characters are driven almost entirely by motion capture, so a full animation rig is not necessary. Skin your new character using the same bones from the DI-Guy skeleton (preferably copied from an existing model.)

LODs

1. To optimize performance, decimate all models to four levels-of-detail. Table 2-1 lists an estimate of the polygon count per LOD.

Table 2-1: Polygon count per LOD for a typical body

LOD #	Polygon Count:
LOD1	5000
LOD2	2500
LOD3	800
LOD4	200

Be sure to name material using DI-Guy naming conventions.

Remember the material will be looked up later in DI-Guy configuration files. DI-Guy supports geometry with one material which may mean that the *head.max* file needs to be cleaned of any multi-materials.

A Note on Files

A typical set of output files looks like:

- *head.dae*
- *body.dae*
- *hands.dae*
- *head_mtl.cfg*
- *body_mtl.cfg*
- *hands_mtl.cfg*
- *head_diffuse.png*
- *head_normal.png*
- *head_specular.png*
- *body_diffuse.png*
- *body_normal.png*
- *body_specular.png*.

The skinned head requires LOD1 and LOD2. For LOD2, decimate the model by 50% of its polygons. The names of the LODs should be “head_lod1” and “head_lod2” respectively. The skinned hands have the same requirements as the head, but with the LOD names “hands_lod1” and “hands_lod2”.

Copy the Collada (*.dae*) File into the DI-Guy Custom Directory

1. Copy the new Collada model into *./custom/geometry/dae*.
2. Copy the RGBs and PNGs this model uses into *./custom/geometry/rgb* and *./custom/geometry/png*.

Create a New DI-Guy Shape Set and Appearance

A Collada model is loaded into DI-Guy the same way an OpenFlight file is loaded. After exporting and processing the Collada file you must create the proper DI-Guy configuration entries to load it. Two files need to be created (or have new entries appended within them) and placed in the *./custom/config* directory.

The file *autoload_appearance.cfg* designates a new DI-Guy appearance that uses the shape set and makes it available to a DI-Guy character type. In the example code below, the new appearance sk_arab_male_1 is defined, and will now be available to the mideast_mped_07 character_type.

```
appearance sk_arab_male_1
  actor = greg
  item = no_body
  item = sk_arab_male_1
  character_type = mideast_mped_07
```

The file *autoload_shape_set.cfg* loads the shape set that stores the Collada *.dae* file and assigns it to an actor. In the example code below, a new shape set sk_arab_male_1 (our convention for all skinned characters is to preface them with 'sk_'.) is defined that uses "greg" as its actor and assigns the dae file to all seven LODs.

```
shape_set sk_arab_male_1
  actor = greg
  is_base_shape = 1
multi_lod_filename = arab_male_1.dae
  shape_attachment_data = body position
  shape_suffix_lod1 = _lod1Mesh
  shape_suffix_lod2 = _lod1Mesh
  shape_suffix_lod3 = _lod2Mesh
  shape_suffix_lod4 = _lod2Mesh
  shape_suffix_lod5 = _lod3Mesh
  shape_suffix_lod6 = _lod3Mesh
  shape_suffix_lod7 = _lod4Mesh
  shader_program = diguy_skin
```

Test New Content and Create Material Files in DI-Guy Character Viewer

To test your new appearance:

1. Open the DI-Guy Character Viewer.
2. Select the character type and appearance corresponding to your new model. You should be able to see your new model, and you can then see how it looks when performing Actions.
3. Create the *mtl.cfg* file that will point all of the various *.dae* files to their respective *.png* files:
   a. Choose **Window** → **Geometry Viewer**. The Geometry Overview window opens (Figure 2-3).
   b. Under the Material States, click Edit.
   c. Browse for the appropriate files.
   d. Click Save Mtl File to save the *mtl.cfg* file. It gets created in *./geometry/dae*.

2.2.4. Creating New Weapon/Equipment Combinations

It is really easy to create new appearances from existing characters and weapons. For example, the weapon pk_machine_gun, is used by appearance taliban_2. What if you would like a soldier to have an appearance where he wears a bdu (battle dress uniform) and is holding the machine gun?

To make that appearance available, create or append the following appearance to *./custom/config/autoload_custom_appearances.cfg*:

```
appearance bdu_pk
  actor = greg
  item = bdu
  item = pk_machine_gun
  item = pk_machine_gun_flash
  item = ak47_ammo_pack
  item = bdu_trigger_hands
  parameters
         weapon_supplemental_data = pk_machine_gun
  character_type = afghan_fighter
  character_type = soldier_07
  quality_bias = 5
  appearance_map = male/adult/white / usa/army/pk74
```

The file *./config/diguy/appearances_human_military_soldier.cfg* contains the taliban_2 appearance. To create the pdu_bk appearance, it was as simple as copying the taliban_2 appearance and then replacing the first item, taliban_2, with bdu.

If you want to add your own weapon model into DI-Guy, create a shapeset that points to your geometry based on the pk_machine_gun and pk_machine_gun_flash entries in *./config/diguy/actor_greg_shapesets.cfg*. Now list your shapeset as an item in your new appearance.

It is also possible to base a new appearance on an existing appearance with the base_appearance line, many soldiers have a base appearance with no weapon so a new appearance can be made to inherit from an existing one:

```
appearance sk_acu_1_SMAW
  base_appearance = sk_acu_1_no_weapon
  item = SMAW_rocket_launcher
  item = SMAW_flash
  character_type = usmc_smaw
  appearance_map = male/adult/white/usa/army/smaw
  is_skinned_appearance = 1
  parameters
    weapon_supplemental_data = m240_cctt
```

3. Creating a Custom Character Type

This chapter explains how to create a new character type.

3.1. Creating a Custom Character Type

Sometimes being able to customize the appearance of an existing character is not enough. This is most often the case when you want to animate a joint hierarchy that does not match any of the existing DI-Guy characters.

3.1.1. A Custom Character Type Example: Vehicle with Turret

In this section, we will create a new character_type for use in DI-Guy called vehicle_turret. It will use the vehicle character_type as a template, but include an extra 3 DOF link to allow a turret to also be animated.

Do not create a new character_type if you can find a joint hierarchy match in an existing DI-Guy character. It is much easier to create a new appearance for an existing character_type than it is to create a new character_type.

The basic steps for creating the character_type are:

1. Create the configuration files that define the new character_type.
2. Test it using the DI-Guy Character Viewer. Keep refining until all load errors disappear and you see your character in the viewer.
3. Verify that your new joints work by modifying the OpenGL pose example in programming examples.

Remember that a secondary goal when customizing DI-Guy configuration files is to avoid masking future changes to DI-Guy data. Always copy a standard DI-Guy configuration file into your custom directory, add more content, and then save it with the same name.

Creating the Configuration Files

The following files are created in the custom directory to define the vehicle_turret character:

```
./custom/config/autoload_vehicle_turret.cfg
./custom/config/autoload_vehicle_turret_appearances.cfg
./custom/config/autoload_vehicle_turret_loader.cfg
./custom/config/common/variables_vehicle_turret.cfg
./custom/config/diguy/char_vehicle_turret.cfg
./custom/config/diguy/actor_vehicle_turret.cfg
./custom/config/diguy/actor_vehicle_turret_shapesets.cfg
./custom/config/diguy/actor_vehicle_turret_links.cfg
```

The following sections examine each file separately.

autoload_vehicle_turret.cfg

This is the top level file for the vehicle turret. It tells DI-Guy to load the associated actor and vehicle.

```
include_optional_once diguy/actor_vehicle_turret.cfg
include_optional_once diguy/char_vehicle_turret.cfg
```

autoload_vehicle_turret_appearances.cfg

This file defines a new appearance for our vehicle_turret, called silly_test_vehicle_turret. The item it references is the silly_test_vehicle_turret shapeset. A good example of an appearance file that was used as a template for this file is *./config/common/appearances_base.cfg*.

```
appearance silly_test_vehicle_turret
  item = silly_test_vehicle_turret
  preload = false
  character_type = vehicle_turret
```

autoload_vehicle_turret_loader.cfg

This file defines the new data loader required to load the new vehicle_turret. Remember, this vehicle has a new joint never loaded by DI-Guy before. A good example of a loader file that was used as a template for this file is *./config/common/loaders.cfg*.

```
data_loader ideal_and_turret
  include common/variables_ideal.cfg
  include common/variables_offset.cfg
  motion_lod_break = here
  include common/variables_vehicle_turret.cfg
  motion_lod_break = here
```

variables_vehicle_turret.cfg

This file references the three new variables for our turret. A good example of a variables file that was used as a template for this file is *./config/common/variables_vehicle_wheels.cfg*.

```
variable = q.turret_rz
  variable = q.turret_rx
  variable = q.turret_ry
```

char_vehicle_turret.cfg

This file defines the vehicle_turret character. The file *./config/diguy/char_vehicle.cfg* was used as the template for this file. We will use the standard DI-Guy vehicle's action table (we will drive the turret programmatically in "Modify the Pose Programming Example to Exercise Your New Joint," on page 3-6).

```
include_once diguy/char_vehicle_action_table.cfg

character_definition vehicle_turret

  abbreviation = vehicle_turret
  actor = vehicle_turret
  default_appearance = silly_test_vehicle_turret

# data_loader = ideal_only
  data_loader = ideal_and_turret

  action_table = vehicle
```

actor_vehicle_turret.cfg

This file defines the vehicle_turret actor. It references the new shapeset and link file, as well as adding a link called "turret". The file *./config/diguy/actor_vehicle.cfg* was used as the template for this file.

```
include_once diguy/actor_vehicle_turret_shapesets.cfg
include_once diguy/actor_vehicle_turret_links.cfg

actor_definition vehicle_turret
  #
  #  standing_hip_height measurement not needed
  #

  include common/links_ideal.cfg
  include common/links_y_forward.cfg
  link = turret
```

actor_vehicle_turret_shapesets.cfg

We define a single shapeset for our vehicle_turret in this file. To avoid the issue of geometry file creation, we are using a human firefighter as our geometry model. That is why we have named the shapeset silly_test. The shapeset references the firefighter OpenFlight files. The group base will be associated with the joint position, while the group head will be associated with the joint turret. See various subsections in "Customizing Appearances by Modifying Geometry," on page 2-7 for information about shapeset files.

```
shape_set silly_test_vehicle_turret
  actor = vehicle_turret
  is_base_shape = 1
  filename = firefighter_LOD1.flt
  filename = firefighter_LOD2.flt
  filename = firefighter_LOD3.flt
  filename = firefighter_LOD4.flt
  filename = firefighter_LOD5.flt
  filename = firefighter_LOD5.flt
  filename = firefighter_LOD5.flt
  shape_and_joint = base position
  shape_and_joint = head turret
```

actor_vehicle_turret_links.cfg

This file defines the link/joint hierarchy of vehicle_turret. The file is the same as *./config/diguy/actor_vehicle_links.cfg*, except that it appends an extra link called "vehicle_turret_turret". This link is placed one meter above the position link.

```
link vehicle_turret_parent
   name = parent
   parent = position
   type = BDI_JOINT_TYPE_6DOF
   name_tx = constant
   name_ty = constant
   name_tz = constant
   name_rz = constant
   name_rx = constant
   name_ry = constant

link vehicle_turret_y_forward
  name = y_forward
  parent = position
  type = BDI_JOINT_TYPE_FROZEN
  offset_rz = -1.5707
  name_tx = constant
  name_ty = constant
  name_tz = constant
  name_rz = constant
  name_rx = constant
  name_ry = constant

link vehicle_turret_turret
  name = turret
  parent = base
  type = BDI_JOINT_TYPE_BALL
  offset_tx = 0
  offset_ty = 0
```

```
offset_tz = 1.0
name_tx = constant
name_ty = constant
name_tz = constant
name_rz = default
name_rx = default
name_ry = default
```

Try to view your new custom character in the DI-Guy Character Viewer. Look at its log to make sure that no unusual warnings or errors were caused by your changes. Most of the messages are quite helpful and a little debugging can often quickly locate the typo that caused your problem.

Modify the Pose Programming Example to Exercise Your New Joint

Edit *./programming_example/diguy_ogl/pose/pose.cpp*. Make the following changes:

1. Change the three static back index variables to be turret index variables.
2. Change the bdi_log_stdout_enable setting to BDI_LOG_INFO.
3. Change the character type in the create_character call from `soldier` to `vehicle_turret`.
4. Change the action in the force_action call from `stand_ready` to `still`.
5. Change the three get_var_index calls to retrieve your turret variables instead of back variables.
6. Initialize the three turret variables to zero.
7. Change the three weight variable indices from back indices to turret indices.
8. In the glut_idle_callback() function, set all three turret variables equal to sin(t). Remove any reference to the back variables.

Recompile and run the example. You should see a fireman with an extra head that rotates in an unusual fashion. If you have problems, check that during initialization it successfully lists the three turret variables, and that your indices are not -1.

4. Creating An Expressive Faces Head Appearance

This chapter explains how to use FaceFX to create an Expressive Faces head appearance.

4.1. Creating and Exporting Data from 3DS Max

1. Start with a sample Expressive Faces model in *./geometry/facefx*.
2. Modify geometry, textures and so on. We recommend using the skin wrap modifier if you need to do any significant changes to vertex topology. The wrap modifier allows you to reference a second model and copy its weights by proximity.
3. Verify that the character can go through all the emotions and poses in the animation data.
4. Pick the FaceFX utility plug-in.
5. Click Create Actor and give the character a reasonable name that will match the associated head appearance.
6. On the Reference Bones panel, click Export and select all bones with the BN_ prefix.
7. Click Batch Export.
8. On the Nodes panel, select *./geometry /facefx/diguy_batch_export.txt*. It has a list of poses and the keyframes at which they happen.
9. Click Save Actor and place the new actor file in *./custom/geometry/facefx*.
10. Select the head of the character and execute the Export Selected command. The file should be exported as an Ogre *.Mesh* file. Give it the same name as the actor and save the file in *./custom/geometry /facefx/*.
11. Select the various mesh LODs of the character and all of the BN_joints and export them as a collada DAE file into *./custom/geometry/dae*.

Any change in the hierarchy of character bones or changes to the neutral pose will probably require that the actor be recreated. When in doubt go through the export process again.

4.2. Setting Up and Publishing a New FaceFX Actor

1. Load FaceFX Studio.
2. Open the actor file that was saved in *./custom/geometry/facefx*. You should now see a the face that was exported as an Ogre mesh file.
3. If you switch to the face graph view there will be an unstructured node graph that does not have constraints or connections. Select **Actor** → **Sync to Template**.
4. Import *./geometry/facefx/diguy_head_face_graph.fxt*. This should now give you a character that can animate and lip sync to audio.
5. Follow the FaceFX Studio manual on how to author animation and create animsets for use in DI-Guy.
6. Run the command **File** → **Publish to Game** and publish the new actor in *./custom/geometry/facefx/published*.

4.3. Setting Up an EF Head for use in DI-Guy

Example of an Expressive Faces shapeset:

```
shape_set facefx_male_w1
  actor = greg
  is_base_shape = 0
  class_type = head
  multi_lod_filename = facefx_male_w1.dae
  shape_attachment_data = head position facefx
  shape_suffix_lod1 = _lod1Mesh
  shape_suffix_lod2 = _lod1Mesh
  shape_suffix_lod3 = _lod1Mesh
  shape_suffix_lod4 = _lod2Mesh
  shape_suffix_lod5 = _lod2Mesh
  shape_suffix_lod6 = _lod2Mesh
  shape_suffix_lod7 =
  parameters
    facefx_actor = male_white_1
```

Special flags for an Expressive Faces head shape set:

- class_type = head. Indicates that this shape replaces other items with class type of head, without this the character will have two heads.
- shape_attachment_data = head position facefx. Indicates that the head mesh should be attached to the position link and be initialized with the FaceFX module shape callbacks.
- facefx_actor = male_white_1. Indicates which FaceFX actor should be used when this shapeset is loaded.

Example of an Expressive Faces head appearance

```
head_appearance facefx_male_white_1
  item = facefx_male_w1
  parameters
    head_type = male_large
    graphics_state_switch_map1 = sk_male_w1_hands:male_w1_head
```

Special flags for an Expressive Faces head appearance:

- head_type = male_large. Indicates which body appearances this head is compatible with. Body appearances are also tagged with a head_type field to allow the user interface to only present the appropriate heads for a given appearance.
- graphics_state_switch_map1 = sk_male_w1_hands:male_w1_head. This state switch applies to the character's hand item and keeps the color of the hands in sync with the color of the head. This is applied after any state switches in the body appearance.

The head appearance functionality isn't limited to Expressive Faces, non-Expressive Faces heads can be made swappable as well, the shape_attachment_data should just not specify `facefx`.

4.3.1. Considerations for Using FaceFX

- Shader lookup table. DI-Guy uses a shader lookup table to establish a mapping between 3ds max bones and DI-Guy Links. The lookup table is specified in *./config/common/shader_matrix_index_tables.cfg*. If you intend to use a different naming convention for your face bones you must add a new shader_matrix_index_table entry to a custom version of *shader_matrix_index_tables.cfg*.
- Number of joints. DI-Guy uses a maximum of 50 link matrixes in shaders. If you intend to build an expressive face with more than 50 joints, you must modify both the shaders and the shader_matrix_index_tables.
- Scale mismatch between DI-Guy and FaceFX. The animation data that FaceFX produces is scaled by .0254 when mapped into DI-Guy's animation system. A few models have required a different value. It has been unclear where in a three application/two file format export pipeline the problems arise. Figure 4-1 illustrates the problem.

Incorrect Correct

Figure 4-1. Scale mismatch

To fix this add a bit of meta data to *./custom/config/diguy/facefx.cfg*:

```
face_data female_white_1
  unit_conversion_factor = .01
```

Scaling of heads in 3ds Max is also very tricky. It may require:

1. Scaling the BN_Head node.
2. Disconnecting all children and scaling the bones back to a scale factor of <1.0,1.0,1.>
3. Renormalizing all scale keyframe data on the joints.
4. Reconnecting the hierarchy.

After a mesh is scaled the mesh will need a Reset XForm modifier added above the mesh and below the skin and collapsed. After changes to joint position it is usually necessary to reset the skin by checking and unchecking Always Deform in the advanced section of the Skin Modifier.

5. Using the Character Viewer

This chapter explains how to use the Character Viewer to view characters, appearances, and actions.

5.1. Introduction to the DI-Guy Character Viewer

The DI-Guy Character Viewer is a utility program that lets you view all of the characters provided with DI-Guy, their appearances, their gestures, and their actions. It also has advanced features for experimenting with lighting and geometry.

The Character Viewer is a Windows-only application.

➤ To open the Character Viewer, on the Start menu, choose **All Programs** → **DI-Guy Applications 13** → **DI-Guy Character Viewer**. The Character Viewer opens (Figure 5-1).

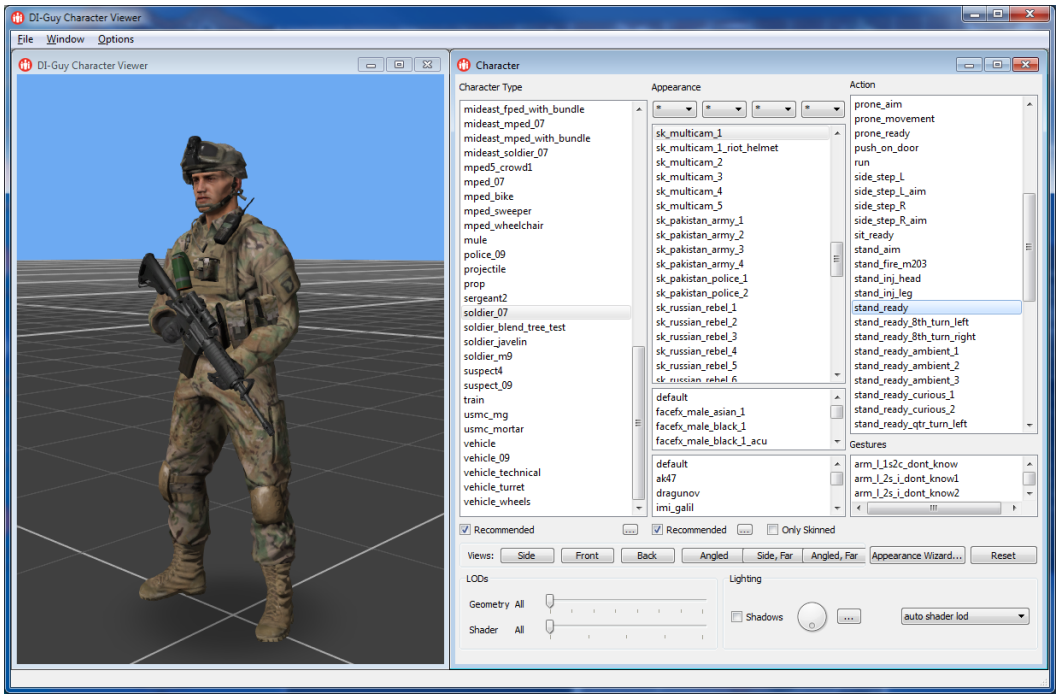

Figure 5-1. Character Viewer

The Character Viewer has three windows, the DI-Guy Character Viewer window, the Character window, and the Geometry Overview window. You can resize them just as you would windows in other applications.

5.2. Viewing Characters, Appearances, Gestures, and Actions

The Character Viewer lists all of the characters available in DI-Guy. For each character, it lists all of the appearances, gestures, and actions that are available for that character. You can view any combination of character, appearance, gesture, and action. You can filter the Appearance list and you can change the perspective from which characters are viewed.

To view a character and its appearances, actions, and gestures:

1. In the Character window, in the Character Type list, select the character that you want to view. It is displayed in the 3D window.
2. In the Appearance list, select the appearance that you want to see for this character. You can filter the list by (from left to right) country, gender, age, and ethnicity. The filter options change based on the character type.
3. To see only skinned characters in the list, select the Only Skinned check box (below the list).
4. In the Action list, select the action that you want to view for this character. The character starts the action. If it is a traveling action, the character repeats it continuously.
5. In the Gestures list, select the gesture that you want the character to use.

5.2.1. Demonstrating a Character's Aim and Locomotion Capability

When you select an action in the Action list, the Character Viewer displays the action in the 3D window. For some aim actions, the Character Viewer pops up a panel that lets you vary the aim azimuth and elevation (Figure 5-2). For some movement actions, you can vary the blend tree settings.

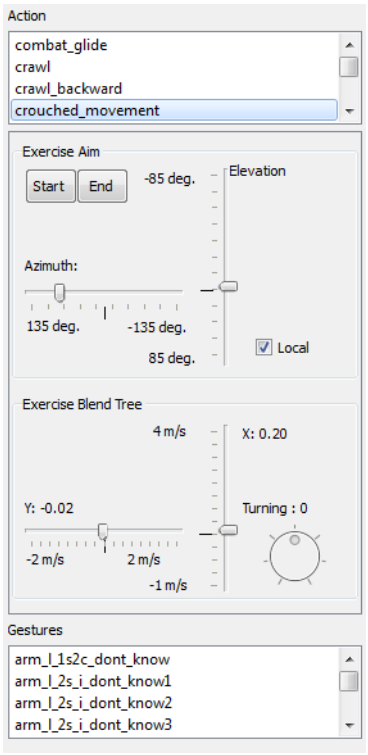

Figure 5-2. Exercise Aim panel

To demonstrate aim and locomotion capability:

1. Select an aim action or an action that uses a blend tree. A panel gets added to the Character window between the Action list and the Gestures list. You may need to resize the window as a whole or the list windows to see all of the panel.
2. Adjust any of the sliders in the panel to see how they affect the character's movement and aiming.

5.2.2. Changing the Viewing Angle for a Character

The default view of a character is an angled view of the right side. You quickly change to a predefined set of angles or move the character using your mouse.

To change the viewing angle of a character do one or more of the following:

- Click any of the Views buttons.
- Zoom in and out using the left and right mouse buttons.
- Change the angle of view by moving the mouse and holding down both mouse buttons.

5.2.3. Changing the Quality of the Character

To optimize performance, DI-Guy characters have several quality levels. The closer a character is to the camera, the higher the quality that is used. If a character is farther away from the camera, it can be rendered at a lower quality, thereby improving performance. You can view characters at their various quality levels.

To change the quality of the character in the viewer:

1. Click the browse button below the Character list. The Character Type Quality Threshold dialog box opens.
2. Select a quality level. The quality of the character in the viewer changes.
3. Click the browse button below the Appearance list. The Appearance Quality Threshold dialog box opens.
4. Select a quality level. The quality of the appearance in the viewer changes.

5.2.4. Viewing Different LOD Levels

You can view a character at the different LOD levels available and you can view different quality of shaders.

To change the LOD level:

1. In the LODs group box, adjust the Geometry slider to see the different LODs for visual quality.
2. In the LODs group box, adjust the Shader slider to see the different LODs for shaders.

5.2.5. Changing Lighting and Shadows

You can quickly make simple changes to the lighting and shadows applied to the character. You can also make more advanced changes.

To change lighting direction and shadows:

1. In the Lighting group box, select the Shadows check box to enable shadows.
2. Rotate the lighting widget to changes the direction from which light is applied to the character (and therefore, the direction of the shadow).

Configuring Advanced Lighting Options

To make advanced changes to the lighting:

1. Click the browse button next to the lighting widget. The Lighting dialog box opens (Figure 5-3).

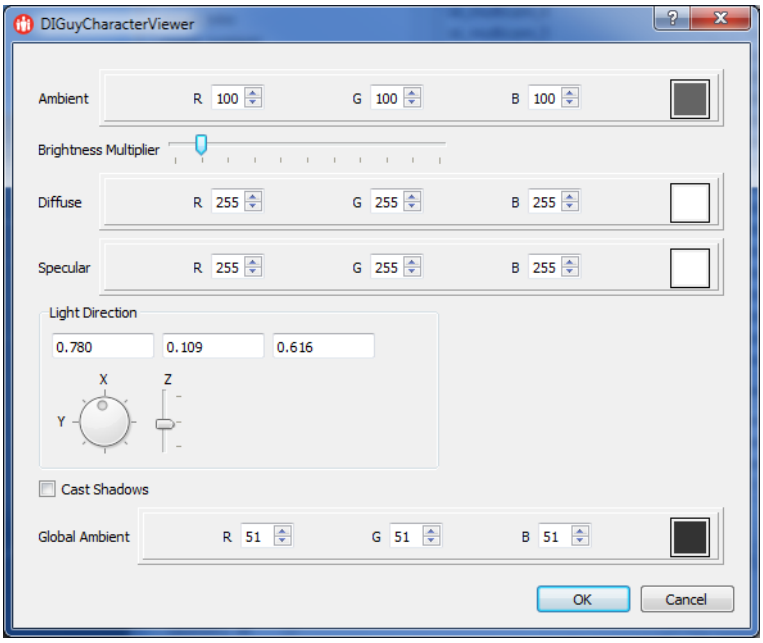

Figure 5-3. Lighting dialog box

2. Change the color of ambient, global ambient, diffuse, or specular light by changing any of the RGB values or by clicking the color swatch and selecting a different color.
3. In the Light Direction group change both the direction and altitude of the light source.
4. Adjust the Brightness Multiplier slider to increase the brightness of the light.
5. To enable or disable shadows, select or clear the Enable Shadows check box.

6. Click OK.

5.2.6. Viewing Multiple Appearances at One Time (Army Mode)

You can display many instances of a character at one time using different appearances.

To view multiple appearances for a character:

1. Choose **Options** → **Advanced**. Two tabs are added to the bottom of the Character window (Figure 5-4).

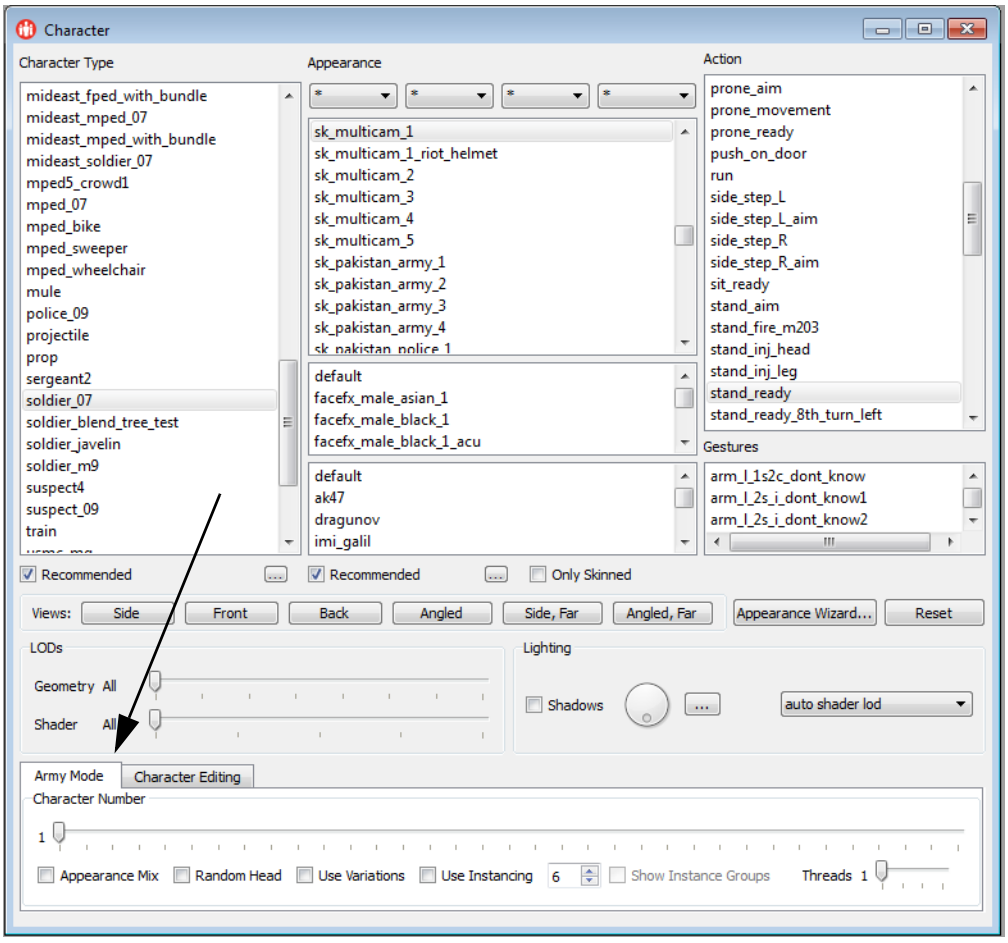

Figure 5-4. Army mode

2. Select the Army Mode tab.
3. Drag the Character Number slider to increase the number of characters displayed in the viewer window (Figure 5-5).

Figure 5-5. Multiple character instances (army mode)

4. Select the various check boxes to change the variations in appearance of the characters in the viewer.

5.2.7. Viewing a Character's Structure

You can view a characters wireframe, skeleton, and bounding volume.

To view a character's structure:

1. Choose **Options** → **Advanced**. Two tabs are added to the bottom of the Character window.
2. Select the Character Editing tab (Figure 5-6).

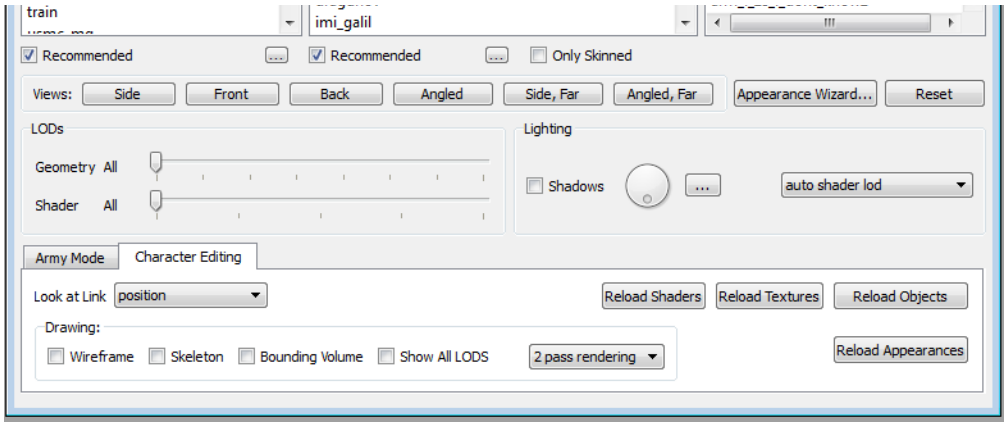

Figure 5-6. Character Editing tab

3. Select the appropriate check box to see a character's wireframe, skeleton, or bounding volume.
4. Select an option from the rendering list to see how the character would be rendered if you do not use 2 pass rendering (the default).
5. If you are editing some aspect of a character in another application, such as Photoshop, you can load the updated files without exiting the Character Viewer. Click any of the Reload buttons to reload the specific type of file you are editing.

5.3. Running a Lua Script

You can run a Lua scripts to debug characters. You can access majority of the *diguyCharacter* API as well as *diguyScenario* and *diguyApp* functions.

To run a Lua script:

1. Choose **Options** → **Run Script**. The Run Script dialog box opens.
2. Type the name of the script you want to run.
3. Click OK,

5.4. Viewing Character Geometry Details

DI-Guy characters are rendered based on information in many different configuration files. Trying to understand a characters structure and functioning by examining all these files can be a tedious process. The Geometry Overview window gathers all of the information for the selected character into one window so you can easily understand its details.

➤ To view character geometry, choose **Window** → **Geometry Viewer**. The Geometry Overview window opens (Figure 5-7).

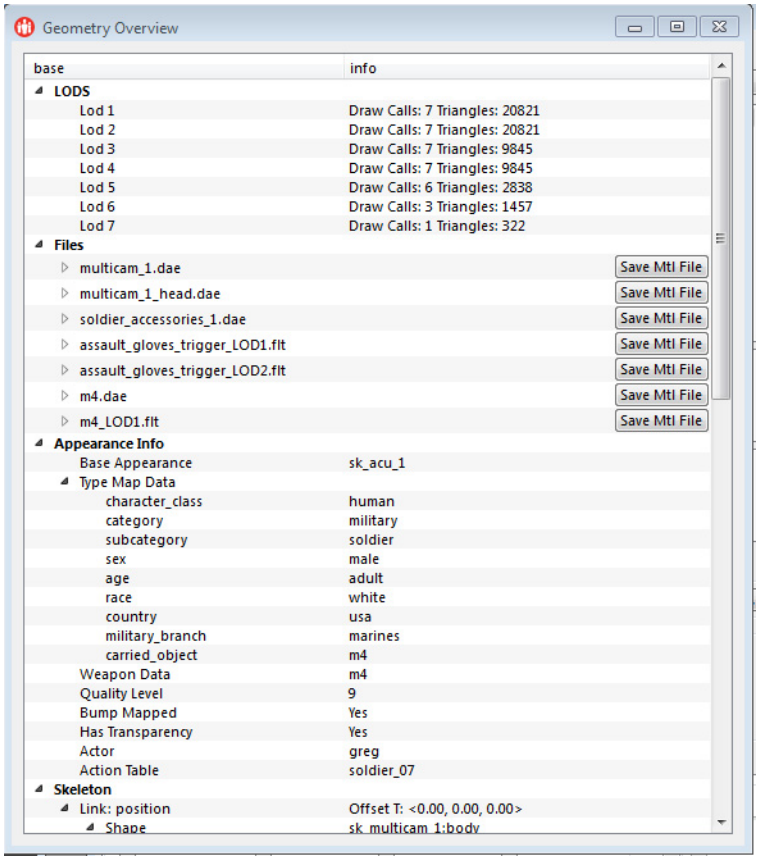

Figure 5-7. Geometry Overview

5.5. The Character Viewer Log Window

The Character Viewer has a log that records system messages while you are using it.

➤ To open the Character Viewer Log window, choose **Window** → **Log**.

Index

Numerics

A

B

C

D

E

F

G

H

J

L

M

P

Q

R

S

T

V

W

Link - Simulate - Visualize

DIG-13.0-3-140523

150 CAMBRIDGE PARK DRIVE, 3RD FLOOR CAMBRIDGE, MA 02140 617.876.8085 www.mak.com