DI-Guy Content

Reference Manual

VT MAK
A company of VT Systems

DI-Guy Content

Reference Manual

Version 13.4

VT MAK
A company of VT Systems

VT MAK
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-13.4-3-190507

Contents

Chapter 2. Tutorials

Chapter 3. Creating a Custom Character Type

Chapter 4. DI-Guy Expressive Faces

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, *DI-Guy Expressive Faces*, 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*. Online documentation that lists all the functions and classes of DI-Guy SDK and explains how to program using the DI-Guy SDK.
- *DI-Guy Documentation Center*. Contains tables of contents for all DI-Guy manuals and a master index to help you quickly locate the correct manual for the information you need.
- *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. It also describes how to use DI-Guy Expressive Faces.
- *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.

These documents are available from the start menu under DI-Guy, or by browsing to *./doc*.

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 MAK to discuss licensing for other uses of derivative models.

MAK Products

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

- MAK RTI. An RTI (Run-Time Infrastructure) is required to run applications using the High Level Architecture (HLA). The MAK 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 entities, aggregate them into hierarchical units, assign tasks, set state parameters, and create plans that have tasks, set statements, and conditional statements. You can simulate using entity-level modeling, which focuses on the actions of individual people and vehicles, and aggregate-level modeling, which focuses on the interaction of large hierarchical units.

  VR-Forces also functions as a plan view display for viewing remote simulation objects 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 simulation objects 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.
  - SensorFX. SensorFX is an enhanced version of VR-Vantage that uses physics based sensors to view terrain and simulation object models that have been materially classified. It is built in partnership with JRM Technologies.
  - The VR-Vantage Toolkit is a 3D visual application development toolkit. Use it to customize or extend MAK'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.
- MAK 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™. DI-Guy is an SDK that allows you to embed the DI-Guy library in your real-time application and populate your world with lifelike human characters. DI-Guy provides high-fidelity human characters that move realistically, respond to simple high-level commands, and travel throughout the environment as directed by the hosting visual application. It comes with thousands of ready-to-use characters, appearances, and motions. DI-Guy character simulation is an integral part of all MAK's visual applications, including VR-Engage, VR-Forces, and VR-Vantage.
- RadarFX. RadarFX is a client-server application that simulates synthetic-aperture radar (SAR). The server application, which is based on VR-Vantage and SensorFX, loads a terrain database and, optionally, connects to simulations. A client application requests SAR images from the server. RadarFX includes a sample client application.

- VR-Engage. VR-Engage is a multi-role virtual simulator that lets users play the role of a first person human character, a ground vehicle driver, gunner or commander, a sensor operator, or the pilot of a fixed wing aircraft or helicopter. It incorporates the VR-Force simulation engine and the VR-Vantage graphics rendering capabilities.
- WebLVC Server. WebLVC Server implements the server side of the WebLVC protocol so that web-based simulation federates can participate in a distributed simulation. It is part of the WebLVC Suite, which includes the server and several sample applications that work with VR-Forces and VR-Vantage.

How to Contact Us

For DI-Guy technical support, information about upgrades, and information about other MAK 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

Internet

MAK web site home page:	www.mak.com
License key requests:	www.mak.com/support/get-licenses
Product version and platform information:	www.mak.com/support/product-versions
For the free, unlicensed MAK RTI:	www.mak.com/support/bonus-material

Post

Send postal correspondence to:	VT MAK 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 MAK 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 *vrforces13.4/doc* appears in the text as *./doc*.

DI-Guy Conventions

The following table lists the conventions used for entity coordinates in DI-Guy documentation.

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

MAK software products may use code from third parties. This section contains summary license information and where required, specific license documentation required by these third parties.

The following libraries are covered by the GNU Lesser General Public License (LGPL) license:

- CIGI
- DevIL
- GEOS
- GStreamer
- LibIconV
- OPCODE
- OpenAL
- OSG
- OsgEarth
- Pthread
- Qt
- Qwt
- LibWebSockets.

The following libraries are covered by the ZLib license:

- Bullet Physics
- Zlib
- LibPNG
- LibSDL
- Minizip.

The following libraries are covered by the MIT license:

- Expat
- FreeGLUT
- GDAL
- GeoTiff
- HarfBuzz
- LibCURL
- LibSquish
- LibXML
- LUA
- LuaBind
- Proj.

The following libraries are covered by the BSD license:

- FreeType
- GLEW
- JPEG
- LibTiff
- Protobuf
- PCRE.

The following libraries are covered by the commercial licenses:

- SpeedTree
- Sundog Triton
- Sundog SilverLining
- FlexLM
- Gameware
- JRM Technologies SenSimRT and SigSimRT
- GLStudio
- OpenFlight
- CDB API.

The following libraries are covered by custom licenses:

- NVidia CG Toolkit
- FreeImage (FreeImage Public License)
- Boost (Boost Software License)
- Poco (Boost Software License).

COLLADA DOM is covered by the SCEA Shared Source license.

Thread Building Blocks is covered by the GPL license:

SQLLite is in the public domain.

Boost License

VR-Link, and all MAK 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 MAK software that uses VR-Link, links in libXML and libICONV. On some platforms the compiled libraries and header files are distributed with MAK Products. MAK 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 MAK products use the Lua programming language (www.lua.org). Its license is as follows:

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. Compressing Geometry Files

As an alternative to encrypting geometry files, you can compress them. This allows you to distribute them without distributing source files. The geometry processor compresses all uncompressed geometry in the custom directory.

➤ To compress your customized geometry,open up a command prompt and type:

```
DIGuyGeometryProcessor
```

1.1.3. 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.

1.1.4. 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/*

1.1.5. 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 *Preface*, 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.

1.1.6. Autoload Configuration Files

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

In addition, any configuration file in *./custom/config/* that begins with autoload_ is read automatically 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

Textures use the RGB, RGBA, and PNG format.

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

Alternatively, you can 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. Windows 7: On the Start menu, choose **All Programs** → **MAK Technologies** → **DI-Guy Applications 13.4** → **DI-Guy Character Viewer (64-bit)**.

   Windows 10: On the Start menu, choose **All apps** → **MAK Technologies** → **DI-Guy Character Viewer (64-bit)**. 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).

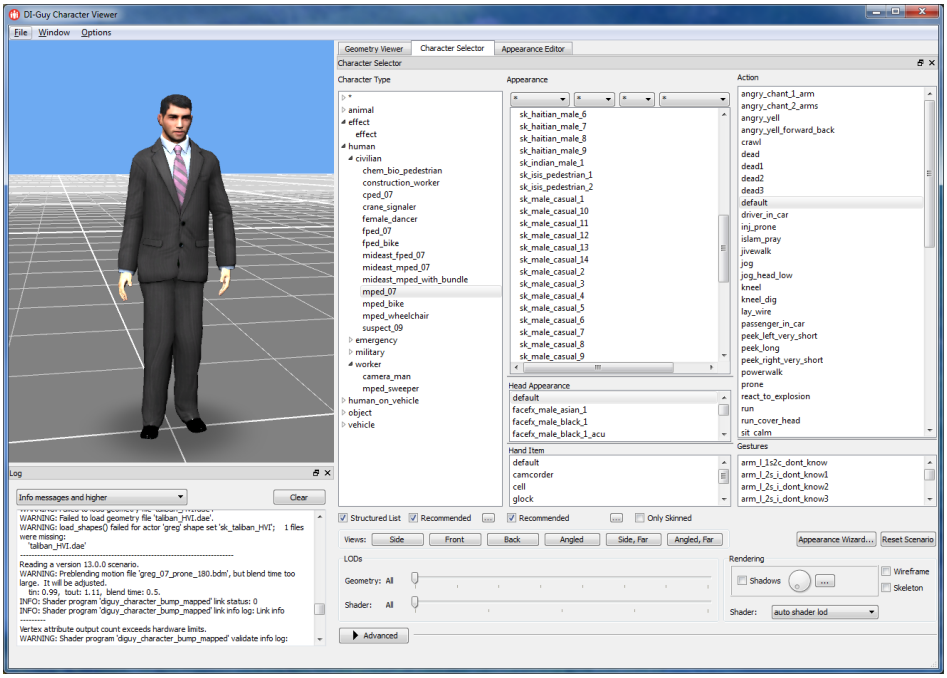


Figure 2-2. Character viewer

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

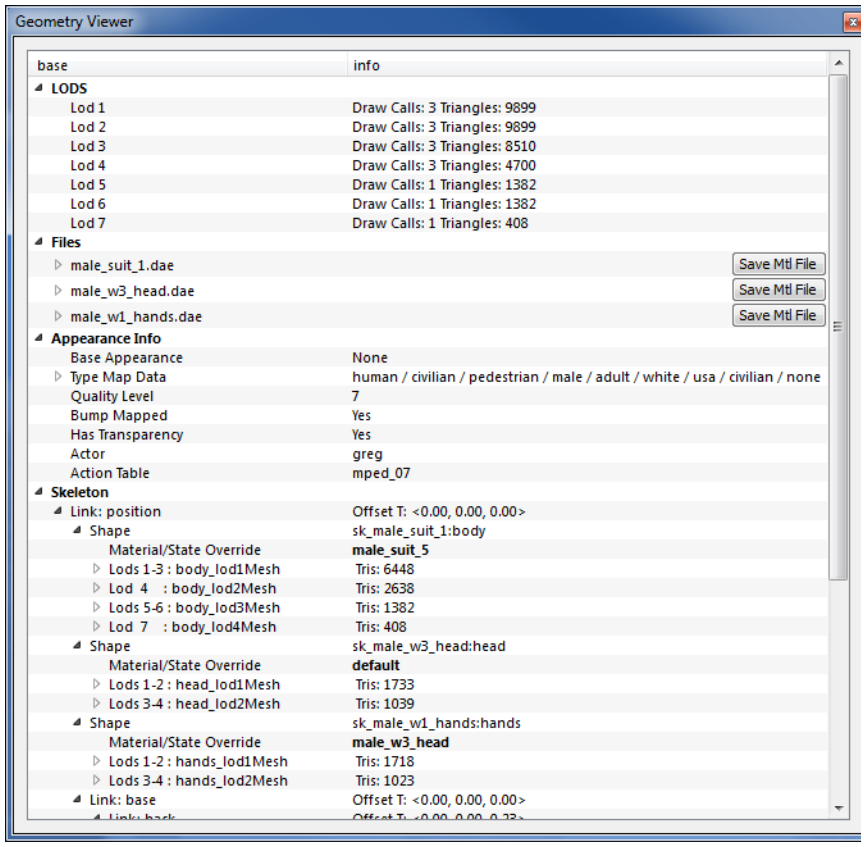


Figure 2-3. Geometry Overview window

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.

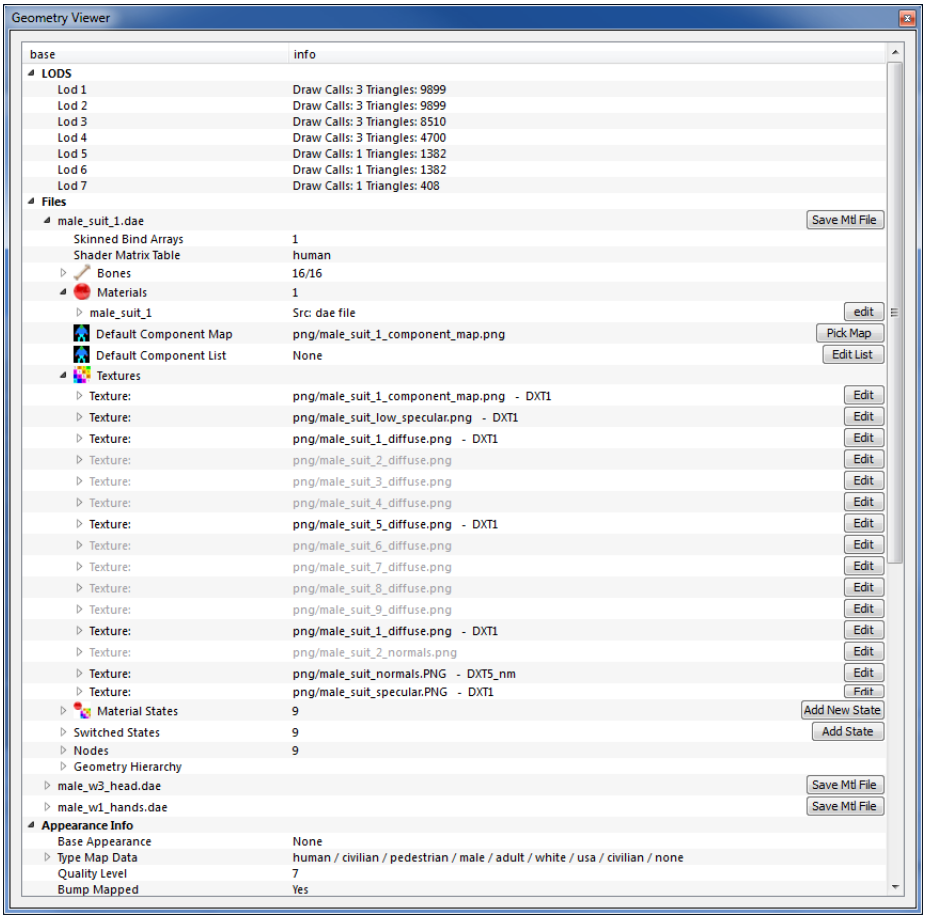


Figure 2-4. male_suit_1 textures

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.

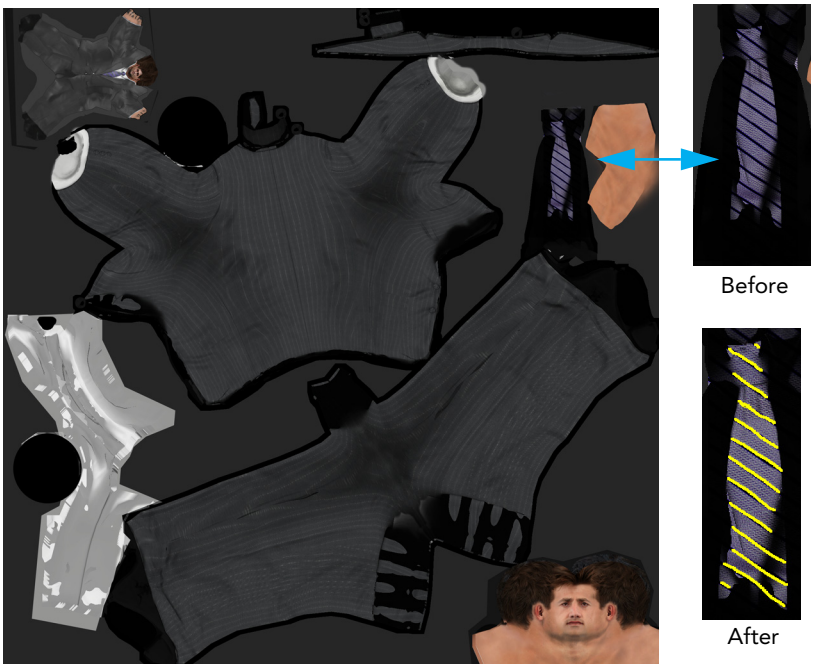


Figure 2-5. *male_suit_1_diffuse.png*

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).

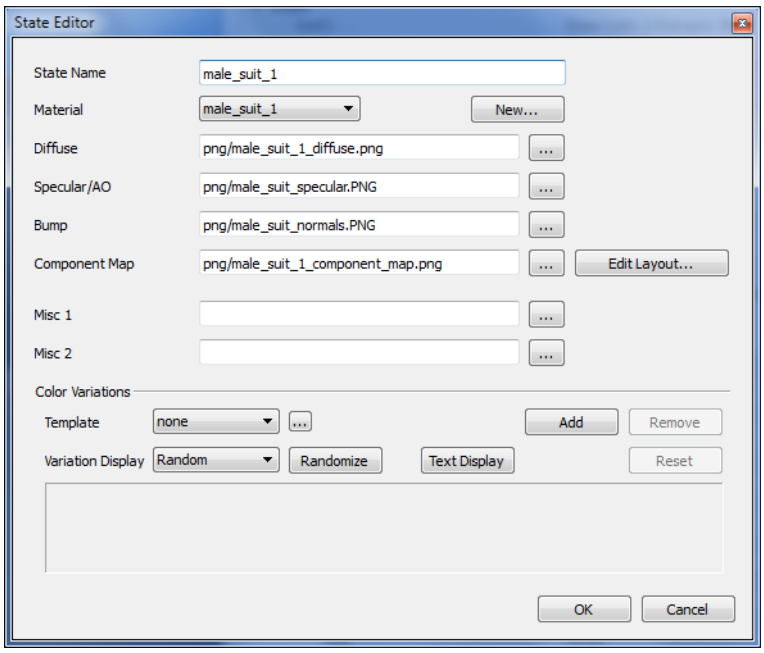


Figure 2-6. State Editor

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 line 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.

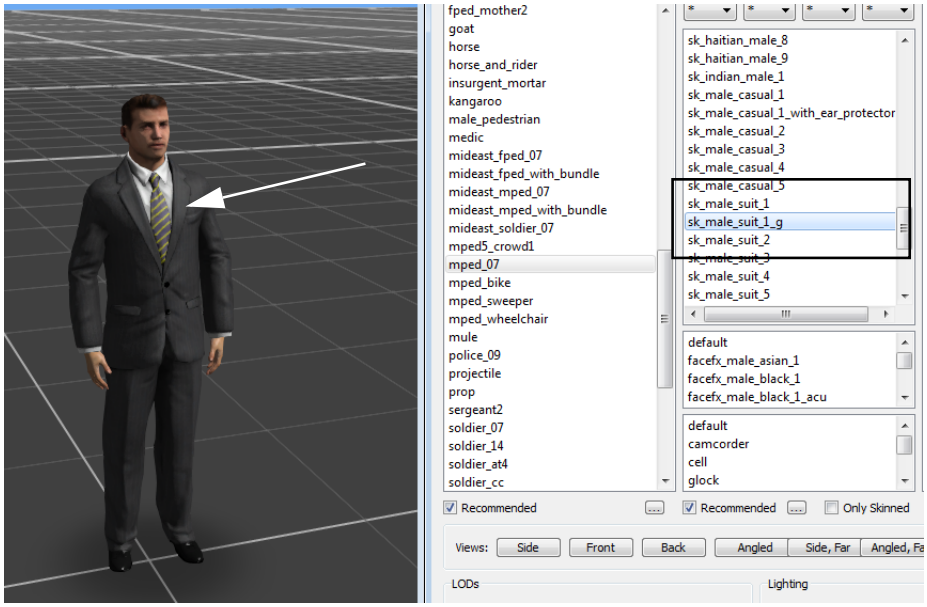


Figure 2-7. Character with modified tie

2.2. Customizing Appearances by Modifying Geometry

Swapping textures is an easy way to customize DI-Guy characters. However, you often need to change geometry to create the exact new appearance you want. 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 OpenFlight 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 uses 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 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. Copying a model similar to the one you want is usually the quickest and easiest way to create a new model.

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.

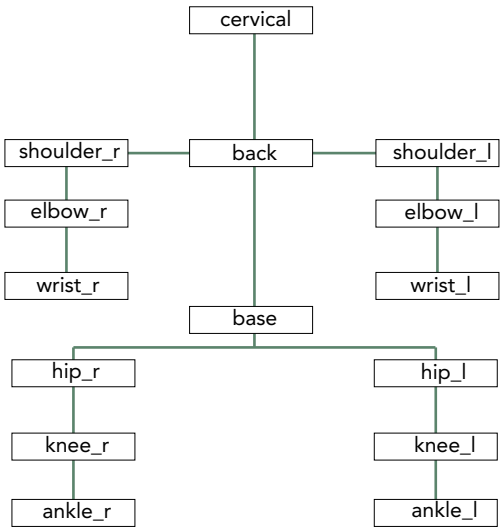


Figure 2-8. Shapes in 3D Studio Max

```
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. You must create or edit *autoload_appearance.cfg* and *autoload_shape_set.cfg* and place them in *./custom/config*.

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 following example code, 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 following example code, 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

In this step, you create the *mtl.cfg* file that points the *.dae* files to their respective *.png* files.

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 see how it looks when performing actions.
3. Choose **Window** → **Geometry Viewer**. The Geometry Viewer window opens (Figure 2-3).
4. Expand the Files section.
5. Expand the DAE file that you want to edit.
6. Expand Material States (Figure 2-9).

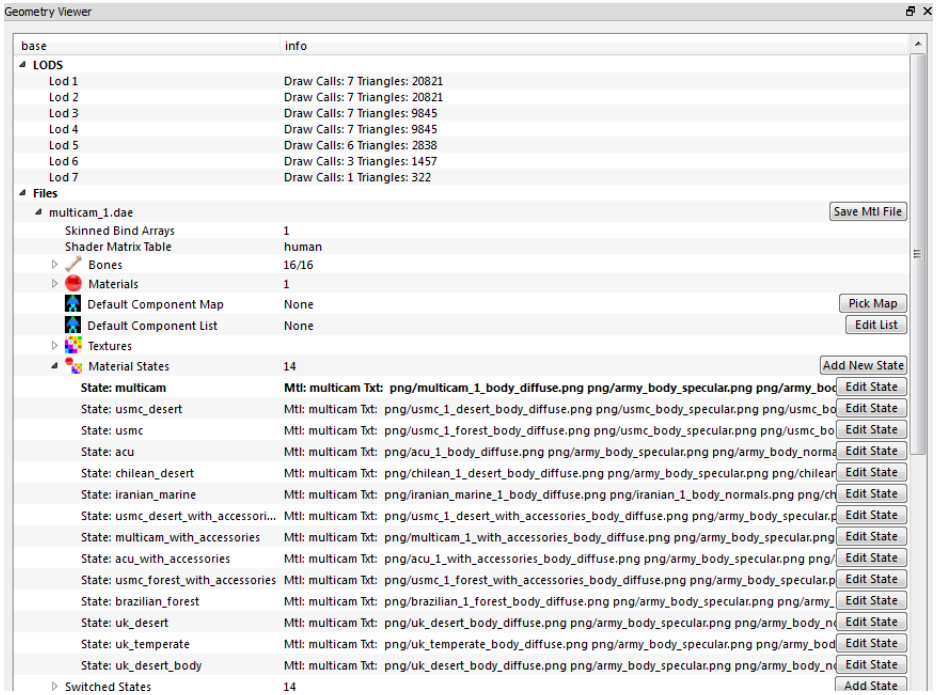


Figure 2-9. Material States

7. For the bolded state, click Edit State. The State Editor opens (Figure 2-10).

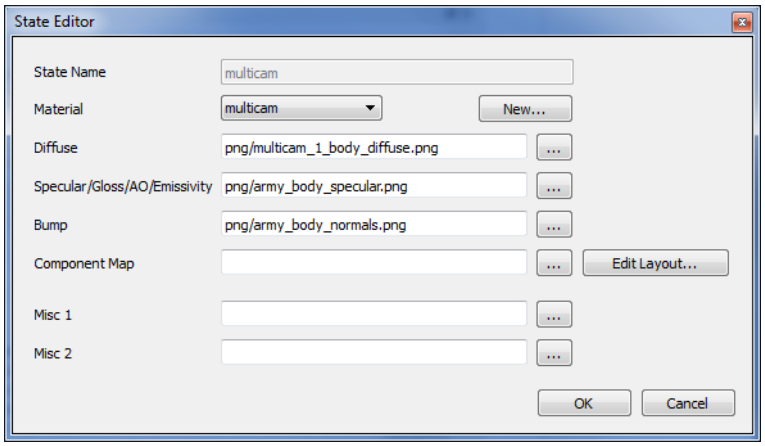


Figure 2-10. State Editor

8. Specify files for Specular/Gloss/AO/Emissivity, Bump, and Component Map entries.
9. Click Save Mtl File (next to the file name) to save the *mtl.cfg* file. It gets created in *./geometry/dae*.

2.2.4. Creating New Weapon/Equipment Combinations

It is 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 want a soldier to have an appearance in which he wears a BDU (battle dress uniform) and is holding a 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, we copy the taliban_2 appearance and replace the first item, taliban_2, with bdu.

If you want to add your own weapon model into DI-Guy:

1. 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*.
2. List your shapeset as an item in your new appearance.

You can also 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

If you want to animate a joint hierarchy that does not match any of the existing DI-Guy characters, you must create a custom character type.

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 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 a character_type are:

1. Create the configuration files that define the character_type.
2. Test it using the DI-Guy Character Viewer. Keep refining it 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 2-8 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. DI-Guy Expressive Faces

This chapter explains how to use the DI-Guy Expressive Faces plug-in.

4.1. Introduction to Expressive Faces

Standard DI-Guy characters have static faces and are often viewed from far enough away that this is a satisfactory visual result (and a win for high performance visualization). The DI-Guy Expressive Faces module enables you to enhance DI-Guy characters with expressive face geometry to create a more intimate performance with their characters including:

- Faces that emote.
- Lips that synchronize to audio.
- Eyes that blink and gaze.

DI-Guy Expressive Faces users also tend to make heavier use of some standard features of DI-Guy such as gesturing, pointing, and gazing.

DI-Guy Expressive Faces is the perfect solution for simulations where communicating with other human characters is critical and characters are close to the camera. DI-Guy have been used for:

- Interrogation training for criminal investigators.
- Cultural training.
- Use-of-force training.

Developers will often couple speech recognition technology with DI-Guy characters to communicate naturally and have conversations with their lifelike Expressive Faces avatars. The avatars in turn often have sophisticated branching logic and performances that allow the avatar to react intelligently to questions, building on previous questions, and so on.

OC3's FaceFX software underpins DI-Guy. Use of FaceFX provides the following benefits:

- The FaceFX Studio Professional application. This software enables you to create lip-synching and other expressive performances. The user interface lets you edit parameters for facial animation, such as blink, eye brow raise, head tilt, and so on, and has a timeline-based editor that lets you tune the performance. FaceFX Studio Professional comes as part of the DI-Guy development license.

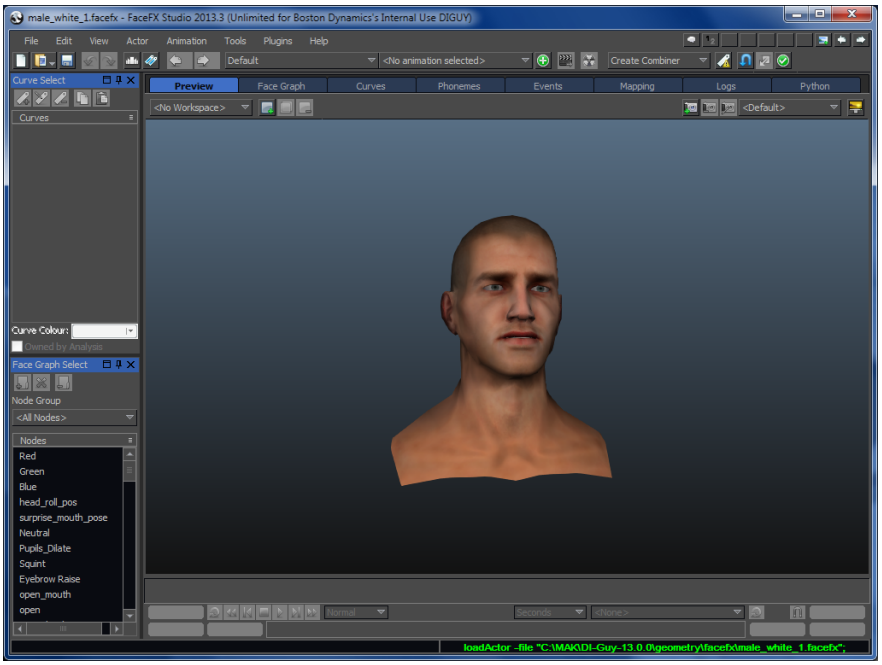


Figure 4-1. FaceFX

- FaceFX is a popular choice in the video game industry. Like DI-Guy, OC3 is investing in the continuous upgrade of FaceFX, so that users who choose DI-Guy can feel confident that their investment will keep pace with the ever-advancing changes in human and facial simulation.

4.2. How to Work with DI-Guy

There are many ways to work with DI-Guy, including:

- Using a DI-Guy AI hierarchical finite state machine (HFSM) to control logic.
- Have users (typically trainees) interact with an avatar
- Using speech.
- Using a button-pushing user interface.

This document does not try to address all possible modes of using DI-Guy. It describes a technique called the expressive local paths technique that has been successful with DI-Guy customers and DI-Guy developers. The goal is to quickly introduce you to using DI-Guy.

We recommend that you use simple nomenclature for files and other assets. If possible, avoid using non-alphanumeric characters, dashes, spaces, and so on.

The expressive local paths technique assumes:

- The scenario to be created will be interactive between a user (referred to as the trainee) and one or more DI-Guy characters equipped with DI-Guy (referred to as avatars).
- The scenario follows a question and answer format between the trainee and the avatar. Occasionally actions performed by the trainee, such as drawing a weapon, will also influence avatar behavior.
- Responses to a particular question or action may be variable and will be affected by what questions or actions have previously occurred in the simulation.

The following steps summarize the expressive local path technique:

1. Create a script.
2. Select an avatar.
3. Select a terrain, props, and background characters and prepare a base scenario.
4. Record voice performances.
5. Create lip-synching animations in FaceFX Studio Professional.
6. Create a simple path for each avatar response.
7. Create signals and a pushbutton interface for signaling questions to the avatar.
8. Create an expressive local path for each avatar response by adding gestures, gazes, and actions to the simple local path.
9. Build up logic for how the avatar chooses a response.
10. Create and train a vocabulary for accepting speech from the trainee and translating it into a signal.

In the following sections, we will examine each step in detail.

4.2.1. Create a Script

The first step to creating your performance is to think about your goals for the simulation and how they can best be achieved. Consider the following questions:

- Is your application training the trainee on how to ask certain questions, or perform a particular task?
- How can your scenario most effectively get at this core education?
- Is the scenario meant primarily to train, demonstrate, or test?
- How can you set boundaries on the simulation?

It will not be possible to have your avatar respond to every conceivable question. You will have to fit your development to the goals and the budget you have for authoring.

For this demonstration, we have created a simple interrogation scene. It tries to do the following:

- Create a simulation likely to be useful to DI-Guy customers.
- Demonstrate a wide range of emotion for the avatar.
- Demonstrate logic that shows that previous questions affect future responses.
- Create a simulation that can be run in 2-3 minutes.
- Choose neutral subject matter.

- The initial script differs from the demonstration scenario included in the distribution of DI-Guy (Exface_Sales_Demo). Like all scripts, it undergoes changes as voice actors and developers acting as directors make modifications during the process of simulation creation.
- This is a purely fictitious scenario intended for demonstration purposes. It does not claim to follow real-world interrogation practices.
- There are two answers to most of the questions; this is because these questions can be asked twice.

Sample Interrogation Script

0? Can we ask you some questions? [Miranda rights]

No, it's OK, I've got nothing to hide. I don't even know what this is about. (friendly)

1? What is your name?

My name's Timothy Johnson, you can call me Tim. (smile)

I already told you, my name's Tim…

2? Where were you last night?

Last night I was with my buddy Johnny, you know, having a few beers. (laughs) We watched TV at his house all night. (happy)

Yea, like I was telling you, I was with my pal Johnny. That guy's the greatest. We watched the baseball game all night at his place.

3? Really? We arrested your friend Pete Kenny last night for burglary, and he said you were driving the car… [establishing witness]

I don't believe you. (disgust) *I was with Johnny last night.*

Why would Pete say that? (disgust)

4? Do you know that an officer was shot last night? [establishing guilt]

What? Really? (confusion) *I….I didn't know that.*

I understand, that's terrible. (sympathy, empathy)

5? Listen Tim, we're here to help, we just want to know what happened. [establishing trust]

I dunno…I was with Johnny last night. (confusion, weakness)

Listen, I didn't do it, but if I did, it would have been for the money, not to hurt anybody!

(Once 3,4,5 have been asked at least once, and perhaps forcing 2 or 3 of the questions to be asked twice, he then confesses.)

Alright, listen, I did it, it was a stupid idea. Pete put me up to it. I didn't want anyone to get hurt. That was Pete!....I think I should get a lawyer.

4.2.2. Select an Avatar

Selecting the best starting avatar involves three things:

- Selecting the best DI-Guy character type.
- Selecting the best body appearance.
- Selecting the best facial appearance.

1. Open the DI-Guy Character Viewer. DI-Guy Character Viewer lets you audition all of the content that comes with DI-Guy. (For details about the Character Viewer, please see Chapter 5, *Using the Character Viewer*, in *DI-Guy Content Reference Manual*. If you have created any custom content, you can view it as well.)
2. Choose the character type that has the most actions similar to what you are looking for in your performance. The "suspect_09" character type is a particularly rich character type that has many interesting actions relevant to simulations that might need facial animation.

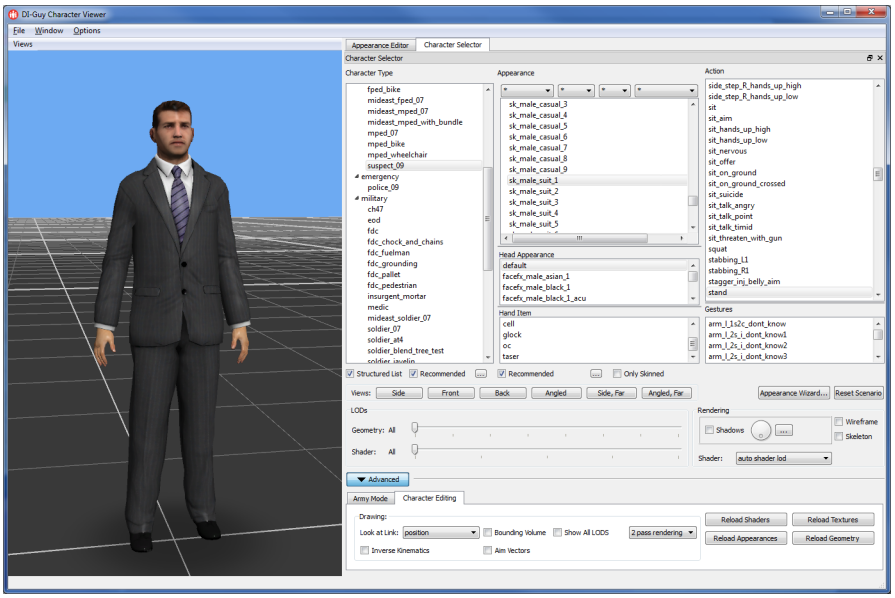


Figure 4-2. suspect_09

3. Find the best appearance for the body and the facial appearance. Use the Appearances column to select different body appearances. Not all appearances have corresponding expressive face appearances. Those that do have a secondary appearance list under the main appearance list in which one or more expressive face appearances are listed.

If you cannot find the right combination of character type, appearance, and facial appearance, please contact DI-Guy Support at support@mak.com for assistance. Because FaceFX-style expressive faces were introduced in DI-Guy 12.0, there are only a small number of sample faces available for certain appearances. Let us work with you to get you the character type, appearance, and facial appearance that will let you be successful.

DI-Guy is fully customizable, so if there are additional motions, character appearances, or facial appearances required, these can be added by the developer. Contact support@mak.com for assistance with facial customization. See *DI-Guy Content Reference Manual* for customizing character appearances and *DI-Guy Motion Editor Users Guide* for instructions about customizing actions for DI-Guy.

You can also contract with VT MAK to create customizations, as well as building a starting scenario or other assistance.

4.2.3. Prepare a Base Scenario

1. Start DI-Guy Scenario[1].
2. Choose a scene object (terrain). It can be one provided by DI-Guy or one that you supply.
3. Determine where you would like your avatar(s) to be located, create a character of the character type and appearances decided in "Select an Avatar," on page 4-7.
4. Choose one or more interesting camera angles to help bring the performance to life.
5. Place appropriate vehicles and props in your scene.
6. If there are any supporting characters, add them to the scenario.

1. The use of DI-Guy Scenario is recommended but not required for implementing Expressive Faces. Developers choosing not to use DI-Guy Scenario need to perform these operations using their own tools.

4.2.4. Record Voice Performances

The quality of the voice recordings will be a major factor in the believability of your end performance.[1] Two factors are important:

- Technical quality. Use a quality microphone. Have post-processing tools that, at a minimum, allow you to crop audio recordings, reduce noise, and output *.wav* files.
- Performance quality. Choose a voice actor that has a voice that matches your avatar and is not a distraction. Be sure that the voice actor not only has a fine voice, but can emote and act convincingly. When producing the vocal performance, give strong direction to the actor to assure you get the voice capture that will make your scenario successful.

Consider video recording the performance to assist with later steps where the facial animation will be emotively tuned in FaceFX Studio Professional and DI-Guy Scenario to help bring the performance to life. Larger budget programs can even consider performing a motion capture session to capture skilled actor performances for reproduction in the simulation.

Final *.wav* files should be placed in *./custom/sfx/<recording session name>*. Consider using descriptive filenames and perhaps including the actor's name in the filename.

1. Advanced Developers may consider using speech generation for their avatar. Contact support@mak.com for support and advice.

4.2.5. Create Lip-Synching Animations in FaceFX Studio Professional

1. Start FaceFX Studio Professional.
2. Choose **File** → **Open Actor**. Available actors are in *./geometry/facefx*. These files have a *.facefx* suffix. You do not need to create a custom file. The face is displayed in the window
3. Press **`Alt`** and move the mouse to rotate the face to a frontal view (Figure 4-3).

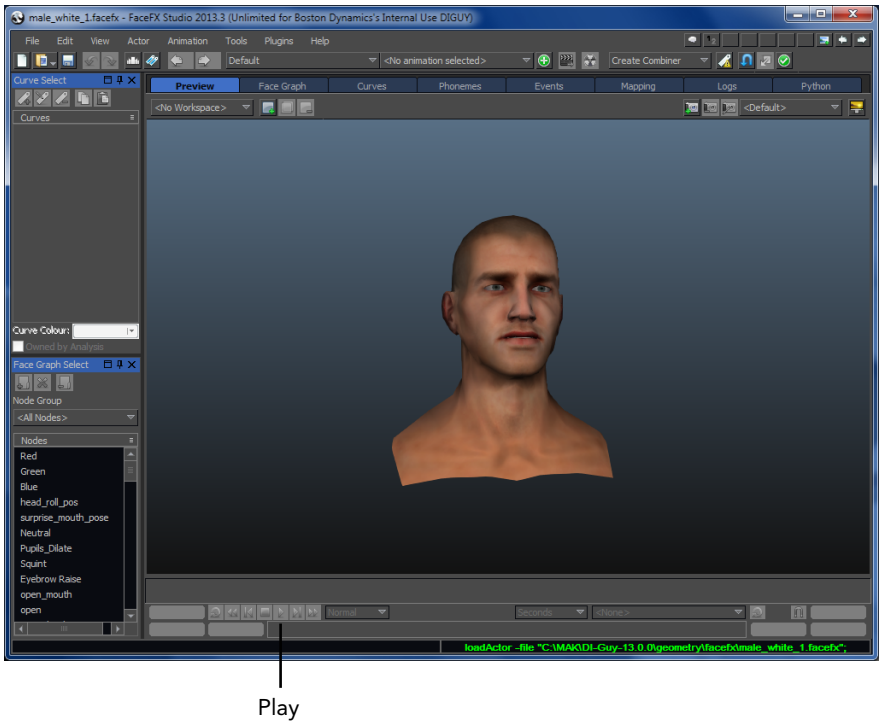


Figure 4-3. FaceFX Studio Professional

4. Generate a new animation group for the actor. Choose **Actor** → **Animation Manager**. The Animation Manager opens (Figure 4-4).

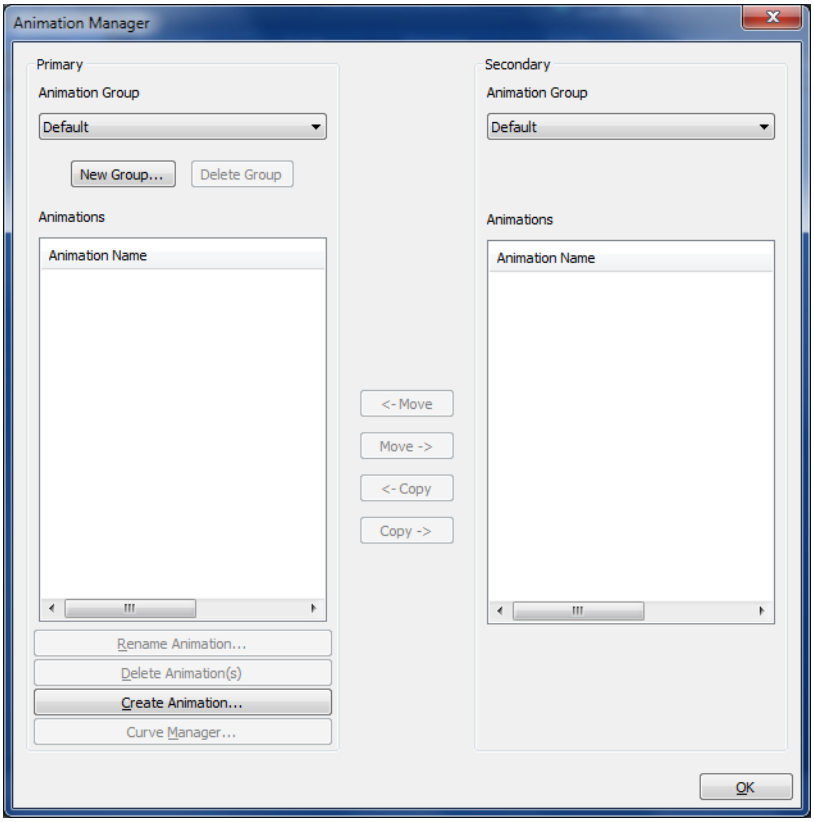


Figure 4-4. Animation Manager

5. Click **New Group**. The Create Animation Group dialog box opens.
6. Type a name for your animation group. We will use “interrogation”.
7. Click OK.
8. Click Create Animation. The Create Animation wizard opens.
9. Select Create an Animation from an Audio File and click Next.
10. Select a *.wav* file and choose Next. The wizard prompts you for the text of the sound file. (If you do not have your own wave file and just want to try this out, select *./sfx/exface_sales_demo/bob_now_were_talkin.wav*. The dialog text is “Now we’re talking.”)
11. Enter the dialog text and select Next.
12. Click Next to accept the language and default analysis actor file. The wizard prompts you with an animation name derived from the wave file name.
13. Click Finish. The animation name is added to the list (Figure 4-5).

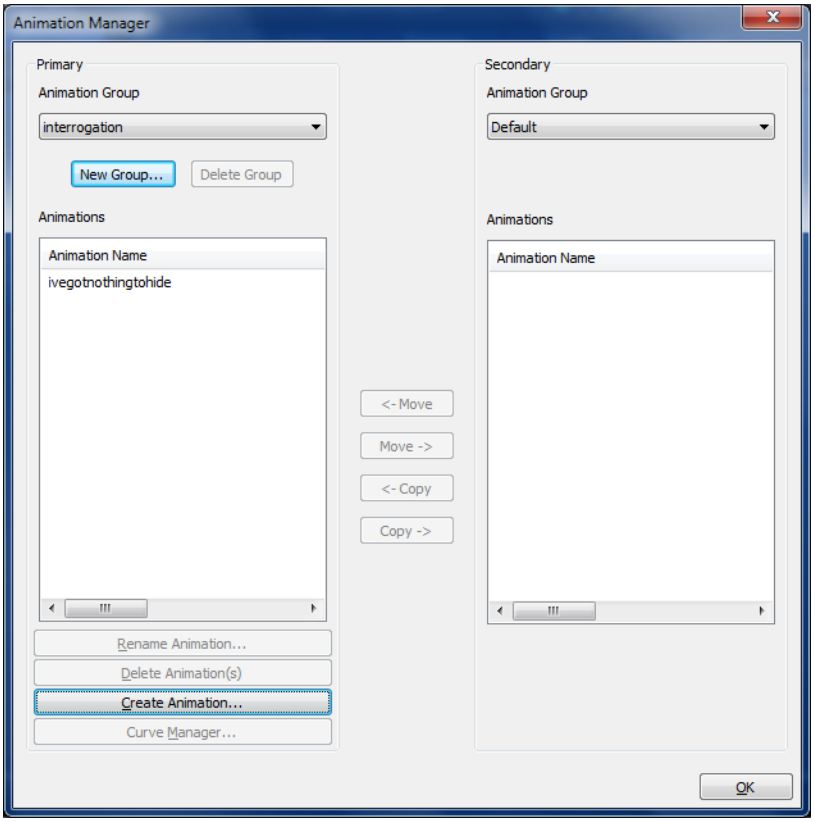


Figure 4-5. Animation Manager with new animation

14. Click OK.
15. In the main window, click Play (Figure 4-3). You should see the character speak the text you have entered.

The character will not only perform the lip-synching, but other facial animation will be created by the system, attempting to get you started on a convincing performance. These may include various head tilts, eye brow motions, and blinks. If you are lucky, the performance will be satisfactory and you can consider publishing the animation and testing it in DI-Guy.

However, you will usually want to improve the performance. FaceFX Studio Professional has a wealth of features for directly editing the performance. The following sections demonstrate some basic tools you can use to refine the performance. See FaceFX Studio documentation for more details.

Refine the Lip-Synching

Examine the lip-synching in detail. If you do not believe that the character's mouth is truly speaking the phonemes you hear, then the quality of the rest of the performance does not matter.

1. Select the Phonemes tab. Figure 4-6 illustrates the phoneme tab for the text, "No. OK. I've got nothing to hide."

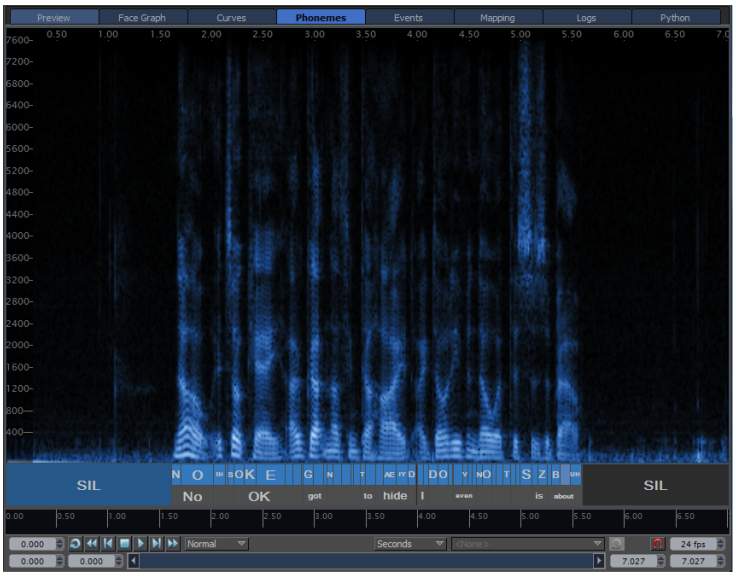


Figure 4-6. Phonemes tab

   The display shows the audio wave. Beneath it is the phoneme breakdown of the sentence as matched to the actual audio wave. Beneath the phoneme breakdown is the exact text that you entered when you created the animation. Timing, in seconds, is shown along the top and bottom of the screen.

2. Adjust the location and width of the phonemes to match the lip movements.
3. Go back and forth between this window and the Preview window. Try playing the performance at quarter time by changing the Normal pulldown on the playback controller to Very Slow. While the voice will be much deeper, it is much easier at this speed to see and correct lip-synching timing issues.

Besides editing the phonemes, you can insert or delete them. One thing you might want to do is close the mouth during pauses. We will see how to do this in the next section.

Curve Editing

One way to edit an animation is to edit curves for the different facial movements, such as opening and closing the mouth, blinking, head movement, and so on.

1. Select the Curves tab (Figure 4-7).

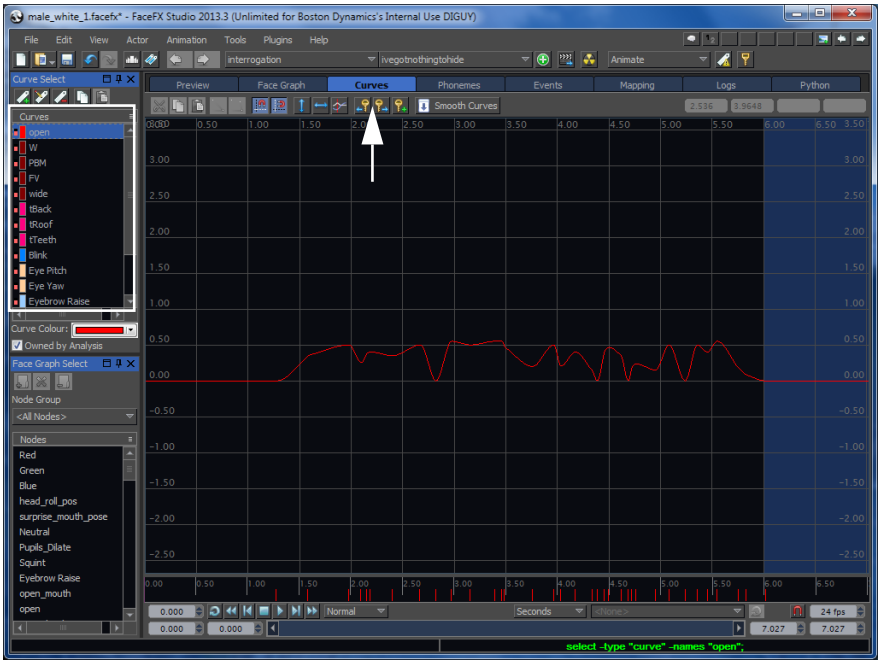


Figure 4-7. Curve editing

2. On the Curves palette, select open. Open controls how far open the mouth is.
3. To edit the curve, clear the Owned by Analysis check box. Points are added to the curve (Figure 4-8). You can move the control points up, down, or laterally in the curve editor. You can add new control points.

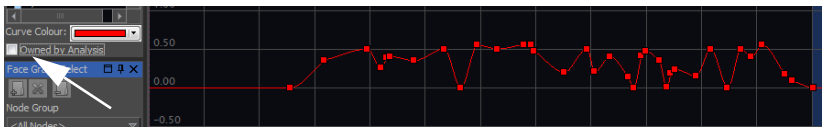


Figure 4-8. Curve points

4. Try editing other curves. Look at some of the other curve types, such as Blink, Squint, and Head Roll.

FaceFX adds head movements based on its analysis of the sound file and text. If the voice recording is fairly short, such as "Now we're talking," it may not add many of the possible types of curves. You might want to add curves that it did not insert or try a longer recording to see if you get more movement added automatically.

To add a new curve type:

1. On the Curve Select palette, click the Add button (). The Select Node to Drive or Type New Curve Name dialog box opens.
2. Select a curve in the list.
3. Click OK. The curve is added to the list. In the window it is a straight line.
4. Edit the curve to add the motion you want.

Working with these parameters in FaceFX is where technology meets art. The variation you can add to your performance is infinite. If you have made video recordings of your actor, you can watch those videos and duplicate the facial animations you liked the best.

Saving and Publishing Your Work

When you are satisfied with the new animation, save your actor and animation set. If this is the first time you are saving the animation set, you also need to save (export) it.

1. Choose **Actor** → **Export Animation Set**. A file browser opens.
2. Save the Animation set to *./custom/sfx/<your project name>/<character name>*. For this demo we called the project name *interrogation* and the character name *suspect*. Once you have done this, the Export Animation Set will be unavailable and the update of the animation set happens automatically during actor save or game publishing.
3. Select **File** → **Publish to Game**. The Publishing Options dialog box opens.
4. Set the Output directory to *./custom/sfx/<your project_name>* and accept the default checkboxes.

Reopening Your Work

When you want to do some more work:

1. Start FaceFX Studio Professional.
2. Choose **File** → **Open Actor** and select the same actor as before.
3. Choose **Actor** → **Mount Animset** to retrieve your older animations.

4.2.6. Create a Simple Local Path for Each Response (Animation)

1. Start DI-Guy Scenario and load the scenario you created in "Prepare a Base Scenario," on page 4-8.
2. Open the Character Objects:Character page and select the character for which you made the animations.
3. In the Head Appearance list, select facefx_male_white_1. This gives the character the head appearance you created.
4. Select the Character Objects:Face Exp. page (Figure 4-9).
5. Select the FaceFx Assets tab.
6. Select your FaceFx Actor (male_white_1).
7. Click Add Animset. The Select Animset to Mount on Actor window opens.
8. Select the animset you saved (interrogation).
9. Click OK. The animation set is added to the Animation Groups section of the Assets list Figure 4-9.
10. Expand the interrogation section. It lists the wave files that you saved in FaceFX.
11. Double-click the ivegotnothingtohide line. The character executes the animation.

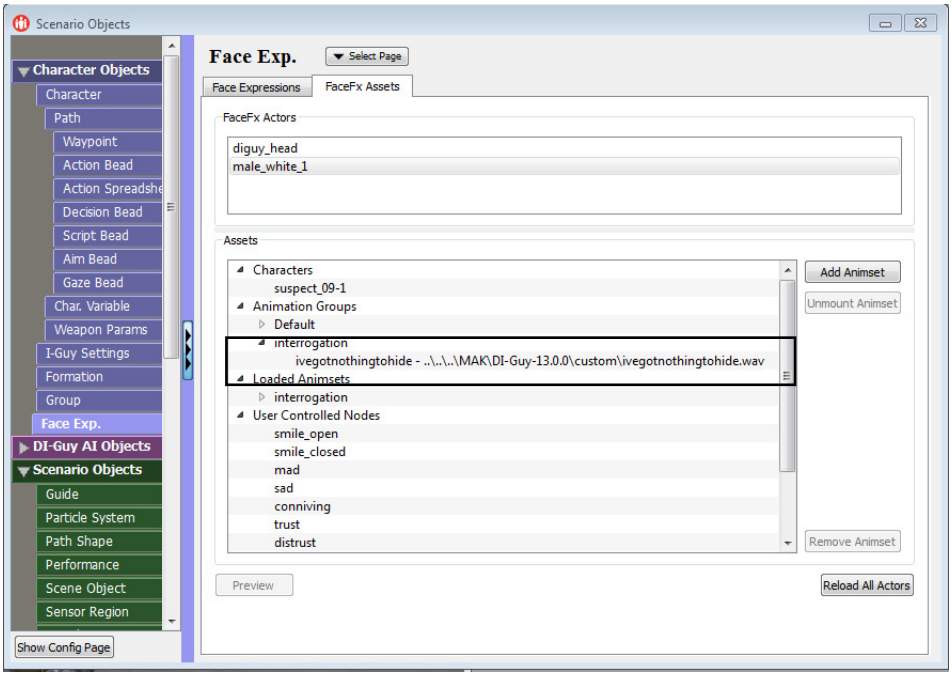


Figure 4-9. Face Expression page

At the bottom of the FaceFx Assets tab there is the Reload All Actors button. This button is very useful. You can keep your FaceFX Studio Professional session running, and as you modify or add animations, once you have saved those changes, you do not need to restart DI-Guy Scenario to use those changes, just click the button.

Create a path for the animation.

1. Since the animation is called ivegotnothingtohide, select the Character Objects:Path page.
2. Add a path.
3. Change the name from path0 to ivegotnothingtohide.
4. Select the Character Objects:Decision Bead page.
5. Add a decision bead and logic to play your animation (Figure 4-10).

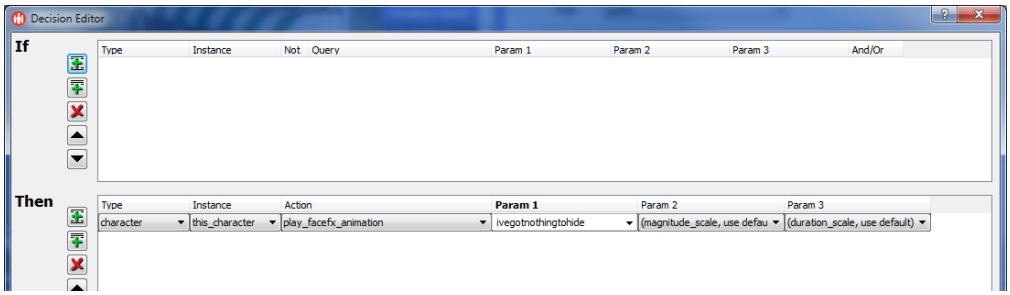


Figure 4-10. Animation setup

6. To audition the path, go to the Character Objects:Character page and make this path be the Initial Path.
7. Click Play. You may want to change the character's action to sit or stand, because the default is to walk.
8. On the Path page, click the Copy button for each animation you created. It is recommended that you set the "Path Paste Offset" to 0, 0, 0 in the Preferences dialog box so that each new copy is in the exact same place.
9. Update the new paths with better names and point the decision bead entries at the appropriate animation.

We now have one or more simple paths, but they are not local. Making these paths be local means that when the path starts, it will begin where the last path left off. This is useful when stringing paths together, the last thing you want is a discontinuity when a new path is activated.

4.3. Creating a FaceFX Head Appearance

This section explains how to create and export data from 3ds Max and how to set up and publish a new FaceFX actor.

4.3.1. Creating and Exporting Data from 3ds Max

1. Open a sample FaceFX model from *./geometry/facefx*. There should be a number of *.max* files to use as a base.
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 (Figure 4-11).

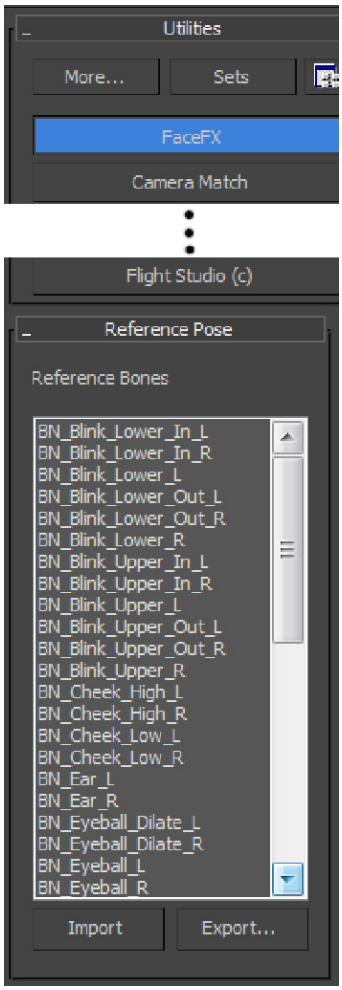


Figure 4-11. FaceFX plug-in

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. On the Nodes Panel, click Batch Export.
8. Pick *./geometry /facefx/diguy_batch_export.txt*. This script has a list of poses and the keyframes they happen at.
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 a Ogre *.Mesh* file. Give it the same name as the actor and save the file in *./custom/geometry /facefx/*.

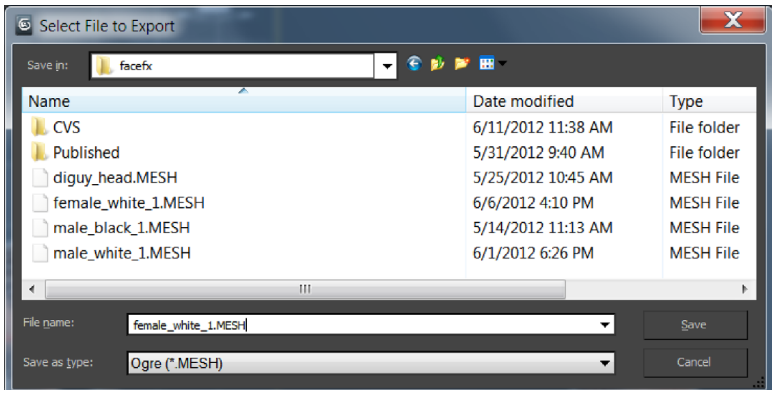


Figure 4-12. Exporting a file

Any change in the hierarchy of character bones or changes to the neutral pose will probably require that the actor is recreated. When in doubt go through the export process again. Select the various mesh LODs of the character and all of the BN_joints and export them as a collada DAE file in *./custom/geometry /dae*.

4.3.2. Setting Up and Publishing a New FaceFX Actor

1. Start FaceFX Studio.
2. Open the actor file that was saved in *./custom/geometry/facefx*. You should now see the face that was exported as an Ogre mesh file.
3. If you switch to the face graph view there will be a unstructured node graph that does not have constraints or connections.
4. Choose **Actor** → **Sync to Template** and import *./geometry/facefx/diguy_head_-face_graph.fxt*. This should now give you a character that can animate and lip sync to audio.
5. Please see the FaceFX Studio documentation for the procedures to author animation and create animsets for use in DI-Guy.
6. Choose **File** → **Publish to Game** and publish the new actor in *./custom/geom-etry/facefx/published*.

4.3.3. Setting Up a Head for Use in DI-Guy

Example of a FaceFX shape set:

```
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 a FaceFX 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 a FaceFX 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 a FaceFX head appearance:

- head_type = male_large. Indicates what 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 synchronized with the color of the head. This is applied after any state switches in the body appearance.

The head appearance functionality is not limited to FaceFX. Non-FaceFX heads can be made swappable as well. The shape_attachment_data should just not specify FaceFX.

4.4. Considerations for Using Expressive Faces

- 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 plan 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 plan 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 3 application/2 file format export pipeline the problems arise. Figure 4-13 illustrates this problem.

Figure 4-13. Scale mismatch

To fix this, add the following 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 scaling the BN_Head node, disconnecting all children and scaling the bones back to a scale factor of <1.0,1.0,1.>, renormalizing all scale keyframe data on the joints, and then 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 and edit all of the characters provided with DI-Guy. It is also a testing tool for DI-Guy customers who create their own models. The Character Viewer has the following windows:

- Character Viewer. Displays the selected character, appearance, and action.
- Character Selector. The Character Selector lets you select the character appearances, gestures, and actions you want to view. It also has advanced features for experimenting with lighting and geometry.
- Appearance Editor. The Appearance Editor is a tool for advanced users who want to edit the parameters of character models.
- Geometry Viewer. The Geometry Viewer gathers all of the information for the selected character into one window so you can easily understand its details.
- Log. Displays errors, warning, and informational messages.

You can undock and resize the windows just as you would windows in other applications.

The Character Viewer is a Windows-only application.

➤ To open the Character Viewer, on the Start menu, choose (Windows 7) **All Programs** → **MAK Technologies** → **DI-Guy Applications 13.4** → **DI-Guy Character Viewer** or (Windows 10) **All apps** → **MAK Technologies** → **DI-Guy Character Viewer**. The Character Viewer opens (Figure 5-1).

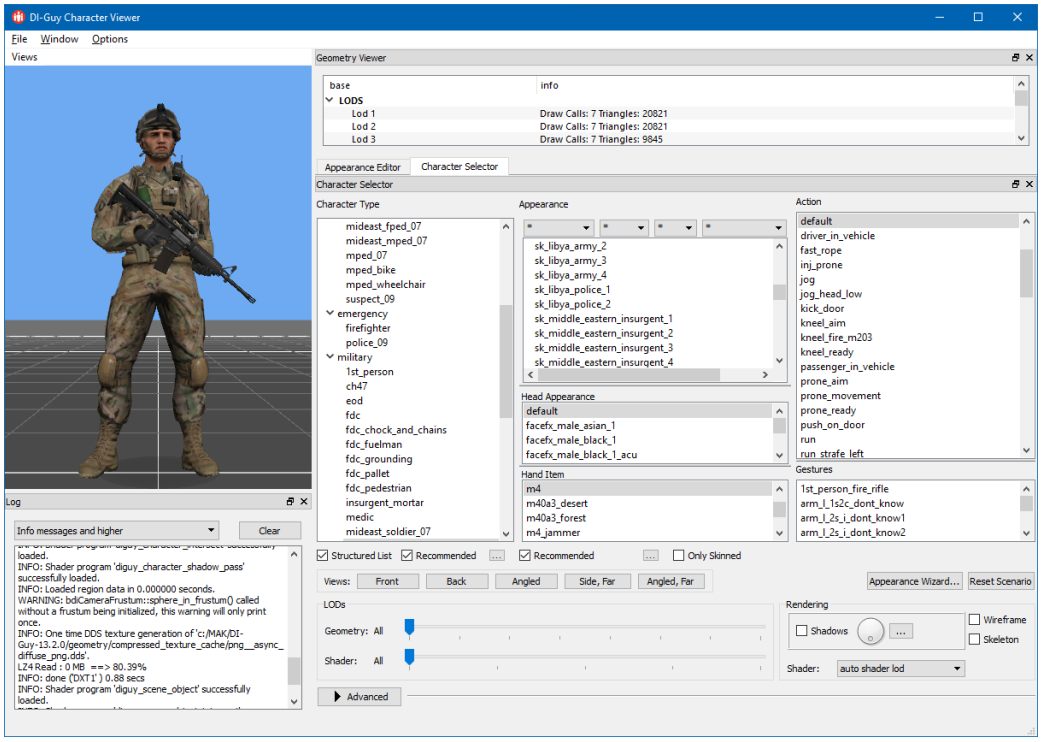


Figure 5-1. Character Viewer

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. Select the Character Selector tab.
2. In the Character Type list, expand the list for the type of character you want to view.
3. Select the character that you want to view. It is displayed in the 3D window.
4. Optionally, to see only skinned characters in the Appearance list, select the Only Skinned check box (below the list).
5. In the Appearance list, select the appearance that you want to see for this character. If the selected character and appearance has head appearances, they are listed in the Head Appearance list. If the selected character and appearance has hand items, they are listed in the Hand Item list.
6. Optionally, select a head appearance.
7. Optionally, select a hand item.
8. 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.
9. In the Gestures list, select the gesture that you want the character to use.

5.2.1. Filtering the Character Type and Appearance Lists

You can filter the Character Type and Appearance Lists in the following ways:

- Organize the Character Type list by type of character or alphabetically.
- Display character types and appearances at or above a specified level of quality.

The quality of a character or appearance is specified in its quality_bias parameter. The quality is a somewhat arbitrary value assigned by the model developer based on their opinion of the quality of the character relative to other characters. The characters provided with DI-Guy have quality values ranging from 1 to 10.

- Display only skinned appearances.
- Filter appearances by country, gender, age, and ethnicity. The filter options change based on the character type.

➤ To organize the Character Type list by category of character, select the Structured List check box. To organize it alphabetically, clear the check box.

➤ To display character types, appearances, or both that are at or above a specified quality level threshold, select the Recommended check box under the list.

➤ To filter the Appearance list by country, gender, age, or ethnicity, select an option from one of the drop-down lists at the top of the Appearance column.

To specify the quality level threshold to use when the Recommended check box is selected:

1. Click the Browse button under the list for which you want to specify a recommended level of quality. A Quality Threshold dialog box opens for the list (Set Character Type Quality Threshold or Set Appearance Quality Threshold.)
2. Set a quality threshold value. The higher the value the higher the quality.
3. Click OK.

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 can 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 (Figure 5-2).

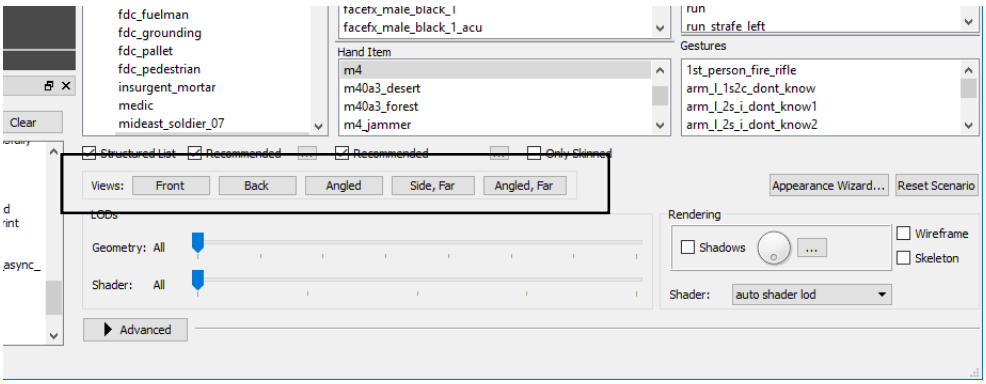


Figure 5-2. Views buttons

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

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. 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 5-3). For some movement actions, you can vary the blend tree settings.

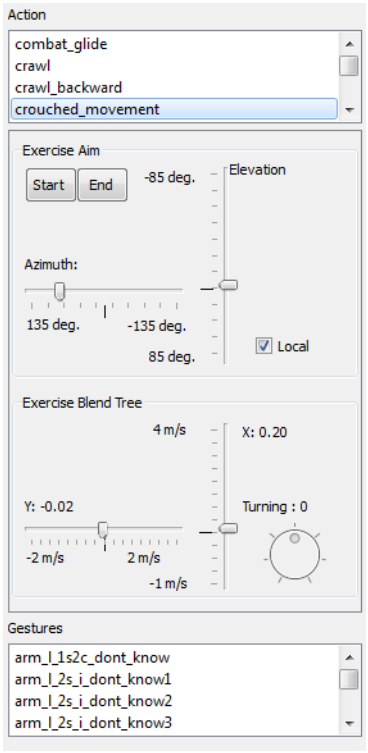


Figure 5-3. Exercise Aim panel

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 Selector 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.

5.2.5. 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.

5.2.6. Changing how a Character is Rendered

The Rendering group box has options for lighting, shadows, and shaders.

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 Rendering 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. In the Rendering group box, click the Browse button next to the lighting widget. The Lighting dialog box opens (Figure 5-4).

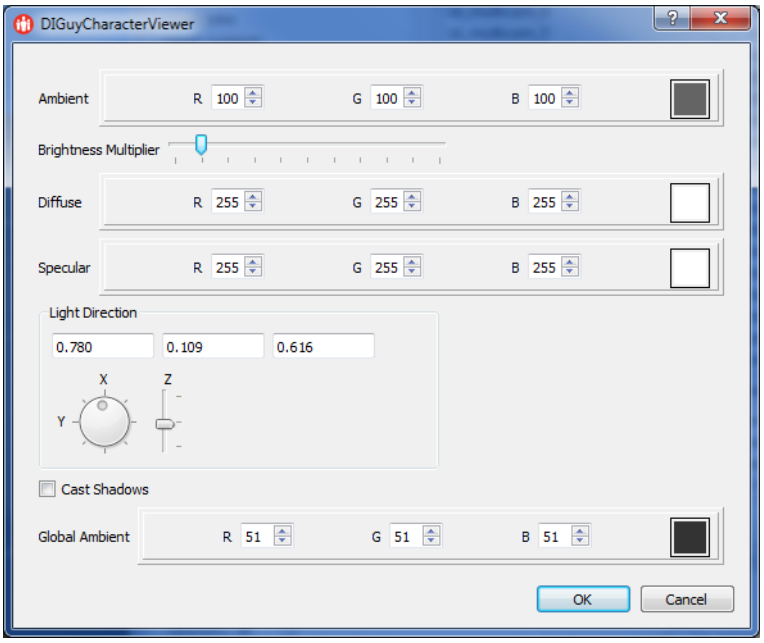


Figure 5-4. Lighting dialog box

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 diffuse and specular light.
5. To enable or disable shadows, select or clear the Cast Shadows check box. This check box is linked to the Shadows check box in the Rendering group box.
6. Click OK.

Viewing a Character's Structure

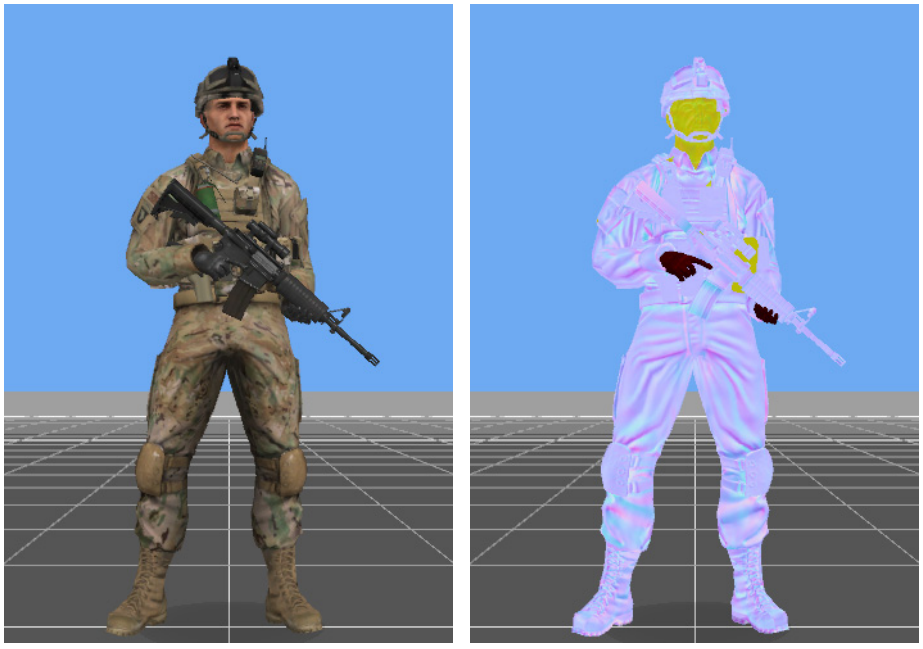
You can view a characters wireframe and skeleton.

To view a character's structure:

1. To view a character's wireframe, in the Rendering group box, select the Wireframe check box.
2. To view a character's skeleton, in the Rendering group box, select the Wireframe check box.

Using the Shader Debugger

The shader debugger allows you to visualize the different texture maps that are used as source data. DI-Guy provides some shader debugging capabilities within its lighting shaders. The shaders are instrumented such that setting different flags (through uniforms) changes the behavior of the shader. Typically it changes fragment colors based on some parameter or parameters. Figure 5-5 shows two different shader settings.



auto shader lod normal map

Figure 5-5. Shader debugging

Shader debugging is also provided in DI-Guy Scenario on the Scenario Objects:Performance page.

➤ To change the shader being used, in the Rendering group box, select an option from the Shader list.

5.2.7. 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. Click the Advanced button at the bottom of the page. Two tabs are added to the bottom of the Character Selector (Figure 5-6).

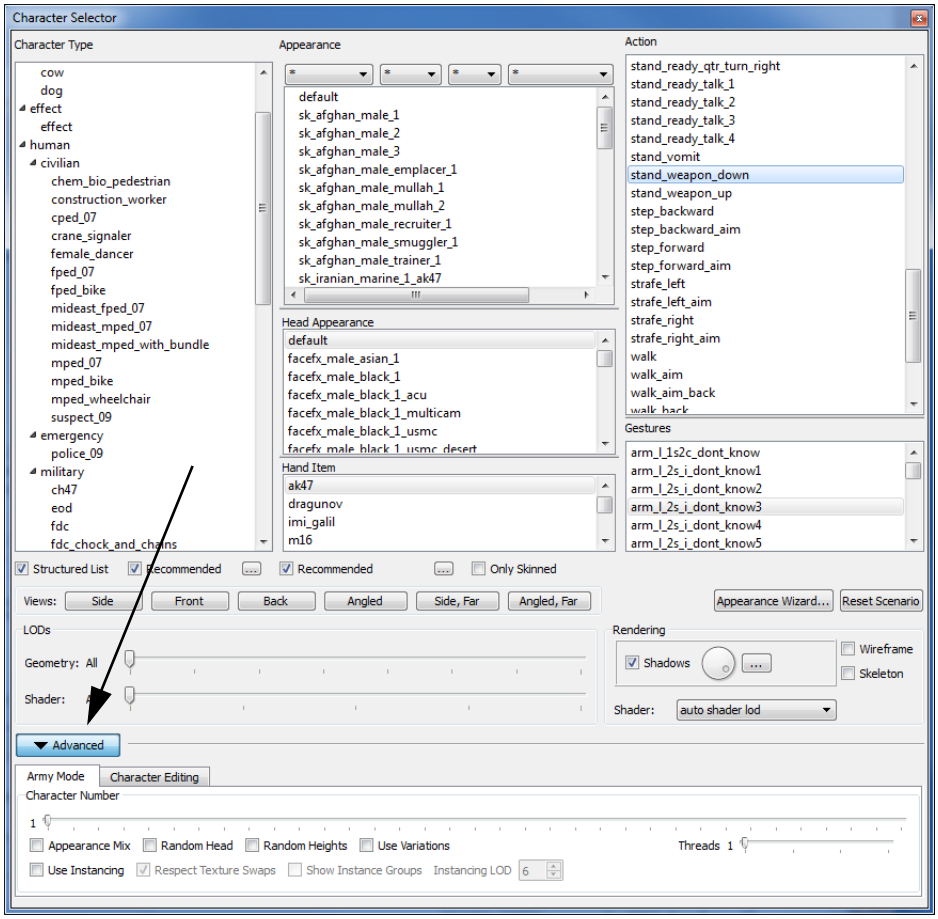


Figure 5-6. Army mode

2. Select the Army Mode tab.
3. Drag the Character Number slider to increase the number of characters displayed in the viewer window (Figure 5-7).

Figure 5-7. Multiple character instances (army mode)

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

Instancing Options in Army Mode

The Army Mode tab has several options for testing instancing. When you use instancing, characters are lower quality, but display more quickly and use less memory. You can see the result of using instancing by turning on the profiler. (For information about the profiler, please see "Displaying the Performance Profiler," on page 5-16.)This option is useful only if you are viewing multiple characters in army mode. If you enable instancing, the following check boxes become available:

- Respect Texture Swaps. Models have a base mesh that has different texture maps applied to it to create the different appearances for a character. For example, a character that wears a suit, might have several different color suits. Replacing the base texture for different appearances is called a texture swap. If this option is selected, the Character Viewer displays the different textures. However, texture swaps affect performance. You can choose to not use texture swaps.
- Show Instance Groups. When you enable instance groups, the characters are colored to show the distance at which LOD changes take effect (Figure 5-8).

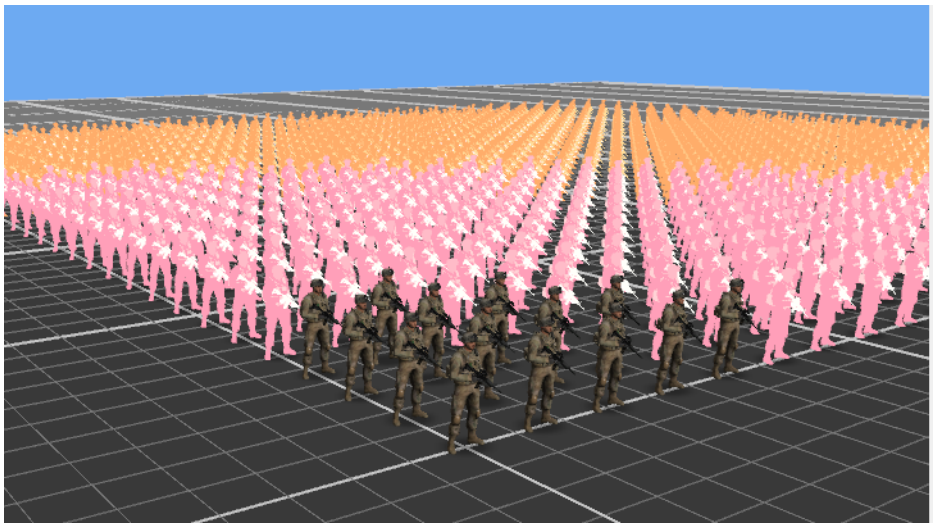


Figure 5-8. Instance groups with Instancing LOD set to 4

- Instancing LOD. Specifies the LOD at which instancing turns on.

To enable instancing:

1. Click the Advanced button at the bottom of the page. Two tabs are added to the bottom of the Character Selector (Figure 5-6).
2. Select the Army Mode tab.
3. Select Use Instancing.

5.2.8. Advanced Character Editing

The Character Editing tab lets you display bounding volumes (Figure 5-9), inverse kinematics (Figure 5-10), aim vectors (Figure 5-11), and LODs. You can also show how a character looks with different rending approaches.

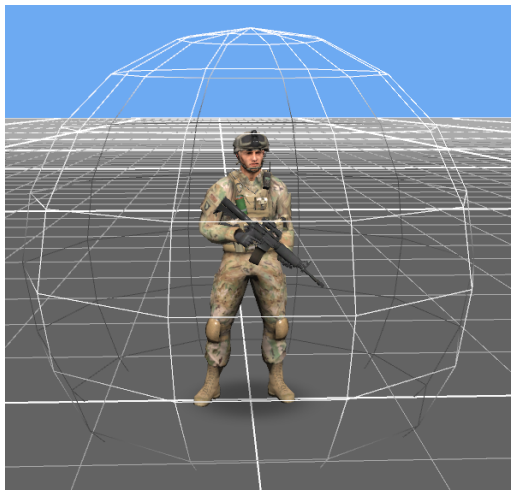


Figure 5-9. Bounding volume

Figure 5-10. Inverse kinematics

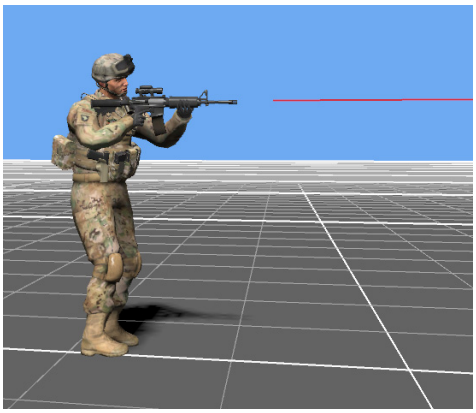


Figure 5-11. Aim vector

To view a character's bounding volume, inverse kinematics, or aim vector:

1. Click the Advanced button at the bottom of the Character Selector. Two tabs are added to the bottom of the page.
2. Select the Character Editing tab (Figure 5-12).

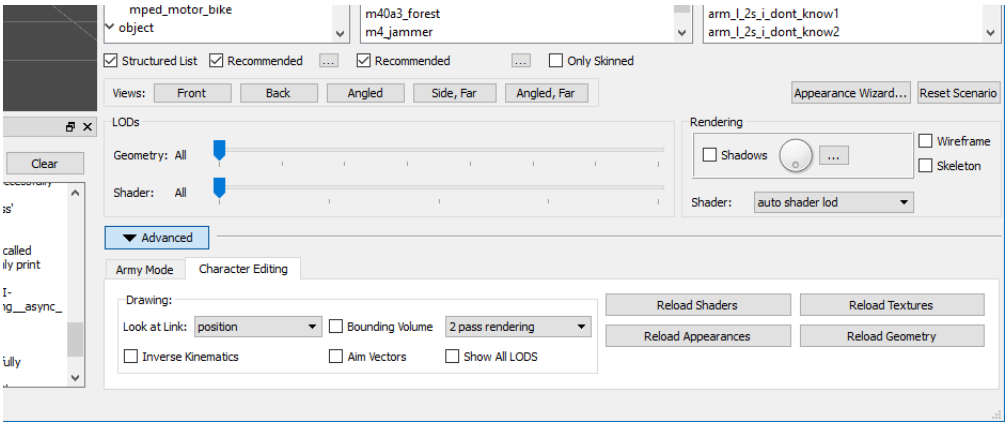


Figure 5-12. Character Editing tab

3. To see the bounding volume, select the Bounding Volume check box.
4. To see inverse kinematics, select the Inverse Kinematics check box. Not all characters display inverse kinematics. This depends on the hand item selected.
5. To see aim vectors, select the Aim Vectors check box. Only characters that have a weapon that is configured for aim vectors and an action that involves aiming display vectors.

5.2.9. Reloading Characters that have been Edited

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. You can reload shaders, textures, appearances, and geometry.

To reload a character's shaders, textures, appearances, or geometry:

1. Click the Advanced button at the bottom of the Character Selector page. Two tabs are added to the bottom of the Character Selector.
2. Select the Character Editing tab (Figure 5-12).
3. Click the button for the type of data that you want to reload.

5.3. Displaying the Performance Profiler

The Character Viewer can display performance statistics for the characters that you are viewing. It displays frame rate statistics, a histogram, and geometry statistics.

To display the performance profiler:

1. Click the character display window to make it active.
2. Press **`Ctrl+p`**. Continue pressing **`Ctrl+p`** to cycle through the performance displays.

5.4. Running a Lua Script

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

To run a Lua script:

1. Choose **Window** → **Script Editor**. The Code Editor dialog box opens.
2. Type the code you want to run.
3. Click Trigger.

5.5. Viewing Character Geometry Details

DI-Guy characters are rendered based on information in many different configuration files. Trying to understand a character's 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.

The Geometry Viewer is also provided in DI-Guy Scenario. It is accessible as the Geometry tab on the Character Objects:Character page and the Scenario Objects:Scene Object page.

If you select a node in the Geometry Viewer in DI-Guy Scenario, the node is highlighted in the 3D View.

➤ To open the Geometry Viewer, choose **Window** → **Geometry Viewer**. The Geometry Overview window opens (Figure 5-13).

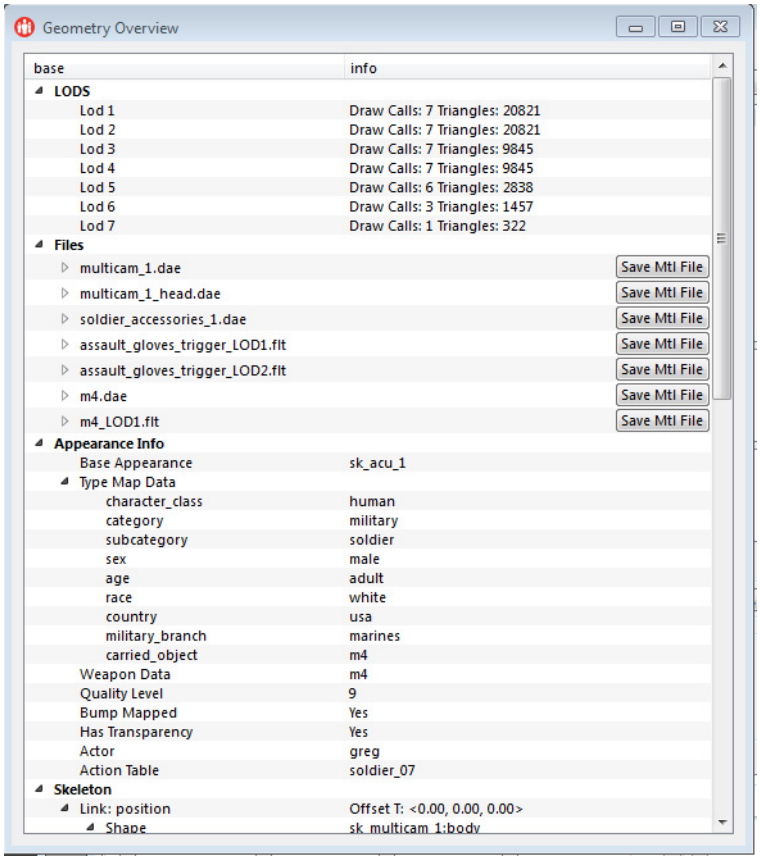


Figure 5-13. Geometry Overview

5.6. The Appearance Editor

The Appearance Editor is a graphical user interface that lets you edit the character appearance files. By using the Appearance Editor rather than editing the files in a text editor, you can avoid syntax errors and invalid information. And since the Appearance Editor shows the configuration for the character selected in the Character Selector, you do not have to figure out which configuration file to edit for that character.

Although the Appearance Editor lets you edit configuration files without touching the actual files, it also displays the contents of the configuration files, so that you can edit them directly if you want to. Sometimes it is quicker to edit the file, particularly if you want to delete a large section of text.

The Appearance Editor is a tool for experienced modelers. It is beyond the scope of this manual to teach all the aspects of modeling needed to use it effectively.

To use the Appearance Editor:

1. In the Character Selector window, select the character whose appearance you want to edit.
2. Select the Appearance Editor window.
3. In the Appearance list, select the appearance that you want to edit.
4. In the list of parameters, edit the parameters as desired. You edit parameters by:
   - Choosing options from a list.
   - Select or clearing check boxes.
   - Typing new values.
   - Clicking buttons to add, insert, and delete new parameters.

5.6.1. Visualizing Character Structure

You can display the following aspects of character structure:

- Bounds. Display the bounding volume.
- Skeleton. Display the skeleton.
- Link Names. Display the names of the skeleton's links.
- Connection Names. Display the names of nodes that have been added to the model for use at runtime. These nodes might not have a joint.
- Vehicle Footprint. Display the areas where the vehicle intersects the terrain.

➤ To visualize character structure, select or clear the options in the Visualization group box.

5.6.2. Editing Configuration File Text

The Appearance Editor displays the text of the configuration file that is being edited in the editor. You can edit the configuration file text directly. If you edit the configuration file text directly, you can refresh the editing GUI. You can control the amount of detail shown in the editing window, as follows:

- Display partial or complete appearance. By default, the window only shows the leaf-level parameters for the appearance. To show the parameters that the character inherits from the base appearance, show the complete appearance.
- Shape sets. By default, shape sets are not displayed in the editing window.

➤ To show the complete appearance, select the Show Complete Appearance check box.

➤ To show shape sets, select the Show Shape Sets check box.

➤ To refresh the editing GUI, click Reload Appearance Text.

5.6.3. Editing Appearance Effects

You can add effects to an appearance, such as dust, brake lights, flames, and rotor wash. The Appearance Editor lets you edit the supplied appearance effects and add more. You can then add them in the Appearance Effects line of the Appearance section of the editor.

To edit appearance effects:

1. Select the Show Appearance Effects check box. The editor displays a list of shared appearance effects.
2. Edit the effects. The GUI works the same way as it does for editing an appearance.
3. To return to editing the appearance, clear the Show Appearance Effects check box.

5.7. The Character Viewer Log Window

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

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

5.7.1. Setting the Notification Level

The Log window displays messages at the notification level that you specify. The levels are:

- Errors only.
- Errors and warnings.
- Informational messages and higher.
- Debug messages and higher.
- Everything.

➤ To set the notification level, choose an option from the list in the Log window.

Index

D

E

F

G

H

J

L

M

O

P

Q

R

S

T

V

W

Link - Simulate - Visualize

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