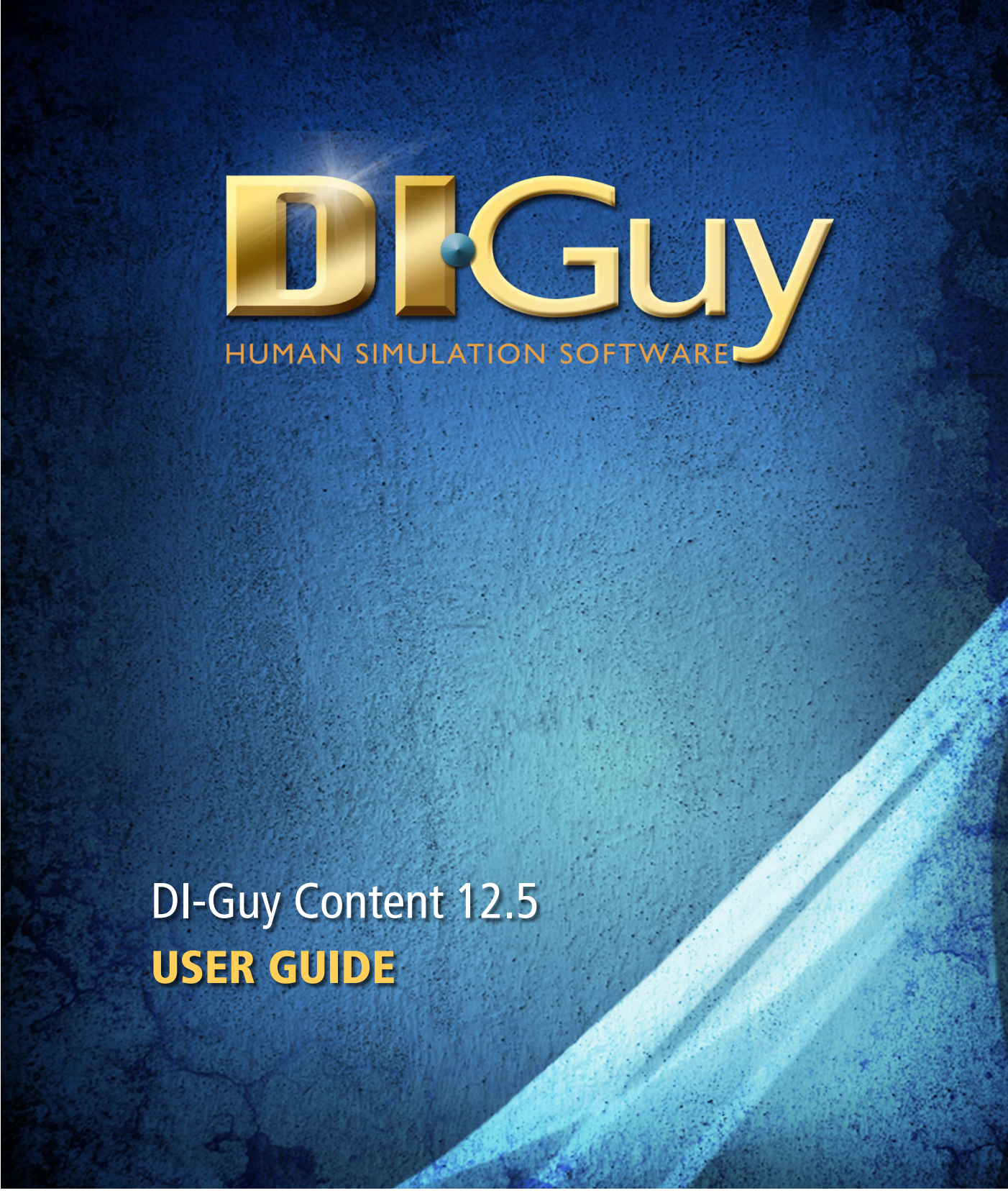


DI-Guy Content 12.5
USER GUIDE

DI-Guy Content Guide
Version 12.5

Boston Dynamics
78 Fourth Avenue
Waltham, MA 02451 USA

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

Part No. DI-Guy 12.5 Content Guide

DI-Guy Software License Agreement

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

DI-Guy User Guide - Table of Contents

Introduction

Welcome to the new *DI-Guy Content Guide!*

DI-Guy is software for realistic human simulations. This document explains how to customize DI-Guy content, and includes the appearance catalogs to show all the content available in DI-Guy.

The DI-Guy Product Line

DI-Guy Content can be used with all of the DI-Guy product line, an integrated set of human simulation products. Members of the DI-Guy Product Line are designed to work together seamlessly:

The DI-Guy SDK – embed the DI-Guy library in your real-time application and populate your world with lifelike human characters.

DI-Guy Scenario – author and visualize human performances in a rich, user-friendly graphical environment. Use DI-Guy Scenario as an end visualization application or save scenarios and load them into your DI-Guy SDK enabled application.

DI-Guy AI – generate crowds of autonomous characters to quickly populate your worlds with hundreds and thousands of terrain-aware, collision avoiding DI-Guys. Used as a module on top of DI-Guy Scenario and DI-Guy SDK.

Expressive Faces Module – enable DI-Guy characters to have faces that display emotion, eyes that look in directions and blink, and lips that sync to sound files.

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

Networking Module – broadcast and receive DI-Guy characters using DIS or HLA. (DI-Guy Scenario only – note that DI-Guy SDK is also routinely networked, see the DI-Guy SDK User Guide on networking to find out how)

Special Effects – use an extremely versatile and customizable particle-based system to create smoke, fire, water, weather, explosions, dust trails, and other special effects. (DI-Guy Scenario only)

For more information about the DI-Guy Product Line, go to www.diguy.com or email us at diguy@diguy.com or sales@diguy.com.

Documentation

DI-Guy comes with lots of documentation on how to use DI-Guy:

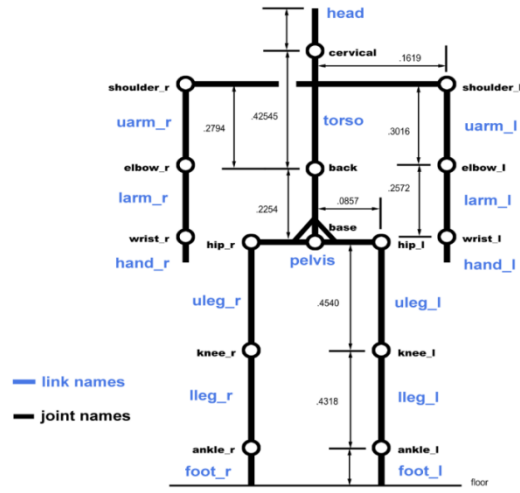
- ***DI-Guy SDK User Guide*** – How to program using the DI-Guy SDK
- ***DI-Guy SDK Reference Manual*** – All the functions and classes of DI-Guy SDK
- ***DI-Guy Scenario User Guide*** – How to use DI-Guy Scenario
- ***DI-Guy Content Guide*** – Images of DI-Guy Content and Customization Guide
- ***DI-Guy AI User Guide*** – How to use DI-Guy AI
- ***DI-Guy Motion Editor User Guide*** – How to import, edit, and customize DI-Guy motions
- ***DI-Guy for Vega Prime*** – How to use DI-Guy with Vega Prime
- ***DI-Guy Digest*** – XML-based file that explains DI-Guy content to non-DI-Guy applications

DI-Guy Conventions

Note the following coordinate system conventions used within DI-Guy:

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

Diagram of representative skeletal structure for DI-Guy characters, including names of joints and links.

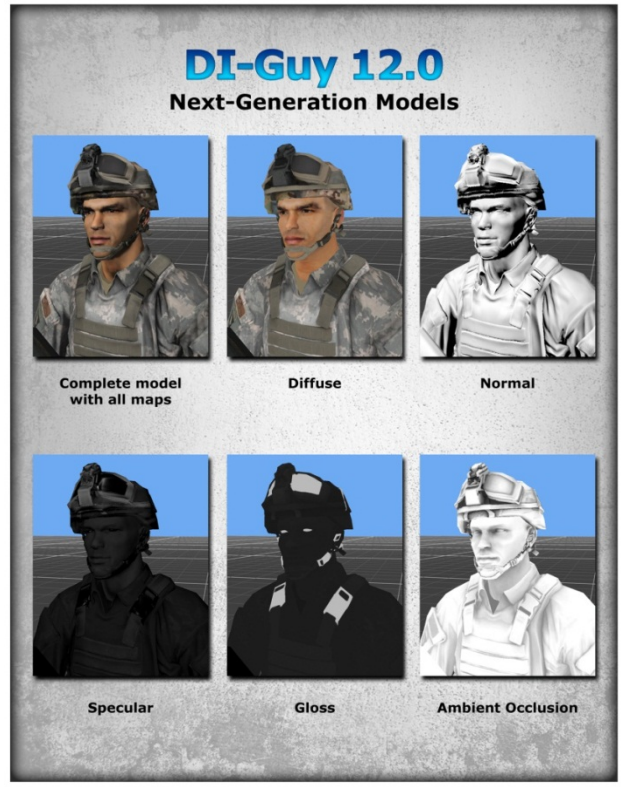


DI-Guy 12.0 Content Release Notes

MODELS

Geometry

DI-Guy is proud to present its next-generation geometry models with the release of DI-Guy 12. The appearances of the popular soldier07, mped07 and fped07 character types and others have been upgraded. These models have the same or reduced numbers of polygons as previous characters, standardization of polygon reduction for Level-of-Detail (LOD), and most importantly draw faster because DI-Guy has significantly reduced the number of draw calls to the graphics card per character.



Multiple Texture Maps

Where previous DI-Guy models only made use of a single diffuse texture maps, DI-Guy 12 characters use three texture maps to create amazingly realistic human character visuals. These maps are:

- diffuse
- normal
- specular

Note that the specular map is actually three grayscale textures combined together representing:

- specular intensity in red
- gloss intensity in green
- ambient occlusion in blue

DI-Guy 12.5 introduces a new variation texture that identifies different parts of a character and allows shaders to modify those at runtime.

Customization

DI-Guy characters continue to be fully customizable.

A popular and easy to perform customization is to swap in a new texture for a character while using the same geometry. 3rdparty image editing programs such as Adobe Photoshop* are suggested. For example, if a user would like to see Justin Bieber's face on an mped_07 (DI-Guy's base male pedestrian character), they should find multiple suitable images of Mr. Bieber, open a copy of the current mped_07's texture such as head_diffuse.png, and then merge the Bieber images into the target texture map, being careful to not *topologically* alter where key pixels in the texture map are located, such as the corners of the eyes.

More advanced users can edit geometry as well using 3rd party tools such as 3D Studio Max. If the user needs the character to be lighter or heavier in terms of weight, they can simply edit the vertices of the character's max file laterally in 3DS Max. The developer needs only be careful to not inadvertently change the number of vertexices. Each vertex is associated with one or more bones. Too drastically altering vertices or changing the numbers of vertices requires "reskinning" the object, essentially resetting the bone associations and weights of each vertex. Once geometry customization has been completed, the geometry is exported as a .dae file and some alterations to the config files of DI-Guy must be performed to include the new appearance.

Expressive Faces

This is another area where DI-Guy 12.0 stands head-and-shoulders (so-to-speak) above previous releases. Unlike default DI-Guy characters, DI-Guy Expressive Faces characters can blink, gaze, lip-sync to audio, and emote. DI-Guy 12.0 introduces OC3 FaceFX as our new partner for expressive face technology. Please see the accompanying Expressive Faces Guide for an overview on generating new heads.

1. Customizing DI-Guy Content

DI-Guy is highly customizable. Motions, geometry, textures, characters, and special effects can all be customized. Be sure to use the custom directory described below 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, please use *Emacs* or *Wordpad* rather than *Notepad*, as *Notepad* does not reliably preserve end-of-line information and may corrupt your configuration files

Be sure to encrypt your final customized geometry!

When customizing .flt and .dae geometry files, always encrypt your geometry to protect DI-Guy intellectual property. Note that this is a requirement as per the DI-Guy Model Use Agreement. To encrypt your customized geometry, simply open up a command prompt and type "diguyEncryptGeometryFile <your filename>.

Where to place customized files

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

Note: It is strongly suggested that you leave all files in **`$DIGUY/`** unchanged and place all your modified and new files in **`$DIGUY/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.

The DI-Guy Data Directory Structure

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

- Geometry **`$DIGUY/custom/geometry/`**
  - Textures **`$DIGUY/custom/geometry/rgb/, png`**
  - Scene Objects **`$DIGUY/custom/geometry/sets/`**
  - OpenFlight Models **`$DIGUY/custom/geometry/flt/`**

  - Collada Files **$DIGUY/custom/geometry/dae/**
- Configuration Files **$DIGUY/custom/config/**
- Motions **$DIGUY/custom/motions/**
- Sound Effects **$DIGUY/custom/sfx/**
- Scenarios **$DIGUY/custom/scenarios/**

Understanding DI-Guy Configuration Files

Content available in DI-Guy is declared in and loaded via the configuration files, allowing DI-Guy to be easily customized via this "soft coding" of content. By default, the configuration file **$DIGUY/config/diguy.cfg** is used by DI-Guy to define content. Note that this file includes other configurations files which also may include more nested configuration files, etc.

If you want to better understand how DI-Guy organizes its data, we encourage you to examine, trace through, and modify files via use of the custom directory "shadow" or "mask" technique described above. All of the config files are in ascii text format, so they are easy to read and understand.

Following is a list of definition of core concepts / terminology used in the DI-Guy configuration file system.

DEFINITION: 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".

DEFINITION: Shape – A mesh or polyhedral associated with a link. Unlike the link which is conceptual in nature, shapes are actual drawables.

DEFINITION: Shapeset – A collection of one or more shapes.

DEFINITION: Joint – Describes the way 2 rigid links are joined together. Popular joints include pin, frozen, 6DOF, ball, 3DOF, and slider.

DEFINITION: Actor – The flesh-and-blood actor whose performance was captured using mocap. 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.

DEFINITION: Character_type – The combination of the set of actions a character can perform with the set of visual appearances the character can use.

DEFINITION: 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). Note that many skinned appearances include equipment that may be segmented.

Here is a brief description of what you will find in diguy.cfg and the files that support diguy.cfg. For ease of understanding, it is broken into three sections, described briefly, and then in more detail below:

1) PREFACE – this section defines very basic things about the content

2) MAIN SECTION – this section defines the bulk of DI-Guy content, often by pointing at lists of Level Two include file content.

3) LEVEL TWO FILES – common files using common syntax to describe DI-Guy content

SECTION 1: PREFACE – this section of the diguy.cfg file defines very basic things. Not typically needed to be modified except for 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 config files themselves
- required_sdk_version – specifies what version of DI-Guy software woth which this data is compatible.
- common/loaders.cfg – Loaders here refers to motion data loaders. Here you can find the exact name of every variable for a s particular DI-Guy character_type. Each "loader" has a name, and that name is referenced in the character_type definitions described later.
- common/shaders.cfg – Obsolete. Replaced by %DIGUY%/geometry/shaders configuration files.
- 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 are described here.
- common/shape_and_joint_data.cfg – the name of a shape is joined to the name of the joint. Thus one can determine what shapes will be drawn when a joint is drawn.
- common/common_actors.cfg – the actor "object" is described. An actor is described by its shapesets and links.
- common/common_characters.cfg – various DI-Guy primitive characters, such as lines, are described here. Character definitions usually have a graphics_class, actor, link list, and default_equipment.
- diguy/required_actors.cfg – Obsolete.

- diguy/required_characters.cfg – defines the character "waypoint"

SECTION 2: MAIN SECTION – this section of the diguy.cfg file defines most of the DI-Guy content. Users wishing to customize DI-Guy content will want to understand this section and the LEVEL TWO content it references.

- diguy/diguy_actors.cfg – this file contains an extensive list of individual actor configuration files. See diguy/actor_<actor_name>.cfg below.

- diguy/diguy_characters.cfg – this file contains an extensive list of character_type definitions. See diguy/char_<character_type>.cfg definition below. The file also includes the config file diguy/diguy_character_type_maps.cfg which associate a single word (e.g – "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 – this file defines the set of character_types that match simple descriptions, for example, more than a dozen character types are deemed relevant when querying what character_types are available as male pedestrians. Like the diguy/diguy_character_type_maps.cfg described above, this list supports wizard-type queries into the DI-Guy content database for best matches to user search criteria.

- diguy/exface_shapesets.cfg – this file defines expressive face appearances and geometry available to particular actors.

- diguy/gestures.cfg – this file defines an extensive lists of gesture motions available.

- diguy/motex_arcs.cfg – this file defines a list of available motion textures, used to add visual noise to character motions.

- diguy/default_particle_descriptions.cfg – this file defines default particles and the parameterizations that define them.

- diguy/appearances.cfg – this file has a list of DI-Guy appearance category include files. See diguy/appearances_<appearance_category>.cfg below 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".

SECTION 3: LEVEL TWO FILES

- 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 the common/links_<link category>.cfg that 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 – the actor shapeset file 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 – the actor link file 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 – this file 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.

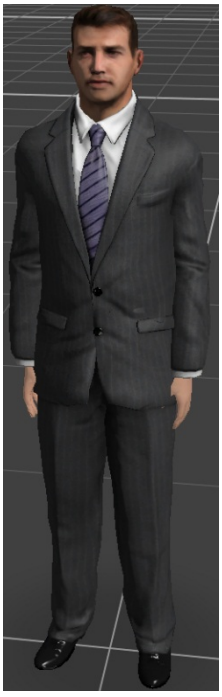
Autoload configuration files

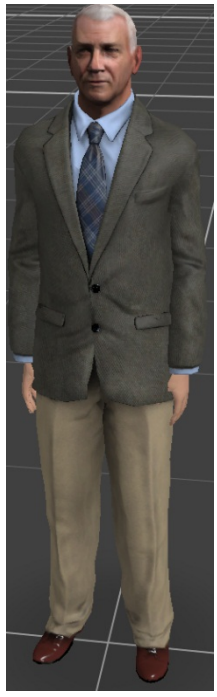
As described above, any files placed in the **`$DIGUY/custom/config`** directory will mask their counterpart in the **`$DIGUY/config`** directory.

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

2. Customizing Appearances using Texture Swaps Tutorial

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 made to be specific to a country; clothing can be made to match a target culture, sensorized appearances can be created to simulate UV or infrared visualization, etc.



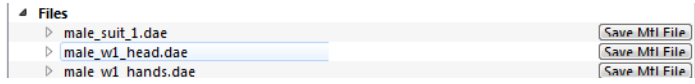


DI-Guy uses texture swapping to add variety to its character appearances while keeping the memory footprint small. Here are two mped_07 appearances, male_suit_1 and male_suit_2. Note how the polygons of the bodies are the same but the texture maps make it appear that the men are wearing different jackets,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

STEP ONE: SELECT A STARTING APPEARANCE TO MODIFY AND UNDERSTAND HOW IT IS BUILT

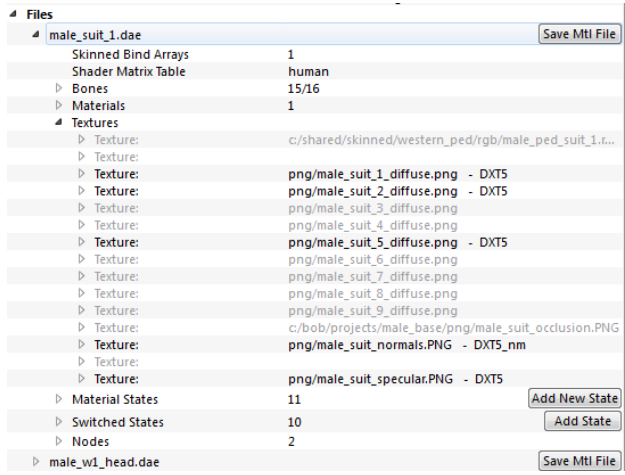
Start the DI-Guy Character Viewer from the Start Menu. Select mped_07 and review the mped_suit_1 and mped_suit_2 appearances.

From the pulldown menu, select Geometry Viewer. You will notice that mped_suit_1 is composed of three geometry files:



Already, we can imagine that head and hands are used for different faces, hair, and skin color.

For our customization, let's keep things simple and create a new appearance for male_suit_1 where his tie has green stripes in addition to the purple. First we need to find the texture map that male_suit_1 uses. 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, suit_2_diffuse and suit_5_diffuse are also highlighted because they had been visualized during the Character Viewer



```
materials

material_states

        state male_suit_1
                material = male_suit_1

                texture
                        filename = png/male_suit_1_diffuse.png
                        texture_type = diffuse

                texture
                        filename = png/male_suit_specular.PNG
                        texture_type = specular

                texture
                        filename = png/male_suit_normals.PNG
                        texture_type = bump

        state male_suit_6
                material = male_suit_1

                texture
                        filename = png/male_suit_6_diffuse.png
                        texture_type = diffuse

                texture
                        filename = png/male_suit_low_specular.png
                        texture_type = specular

                texture
                        filename = png/male_suit_normals.PNG
                        texture_type = bump
```

sessions when the screengrab was performed. It is not hard to imagine that male_suit_1_diffuse is used for the diffuse texture, male_suit_normals for the normal map, and male_suit_specular for the specular map. But to understand that better, select “Save Mtl File”. Use a text editor to open the file “male_suit_1_mtl.cfg”, found in %DIGUY%/custom/geometry/dae, as seen in the image to the right.

STEP TWO: MODIFY THE TEXTURE MAP

This step is as easy or as hard as your artistic skills and ambitions. This author’s artistic skills are weak, but his ambitions low! First, open the file %DIGUY%/geometry/png/male_suit_1_diffuse.png using your favorite paint program.

Remember to be careful about not modifying the topology. The x,y coordinates of the image map to the u,v of the target mesh. Changing the topology would require a re-mapping. You can see the tie in the upper right. Below are zoomed images of the tie before and after adding the green stripes. Save the texture as male_suit_1_g_diffuse.png in the %DIGUY%/custom /geometry/png directory.

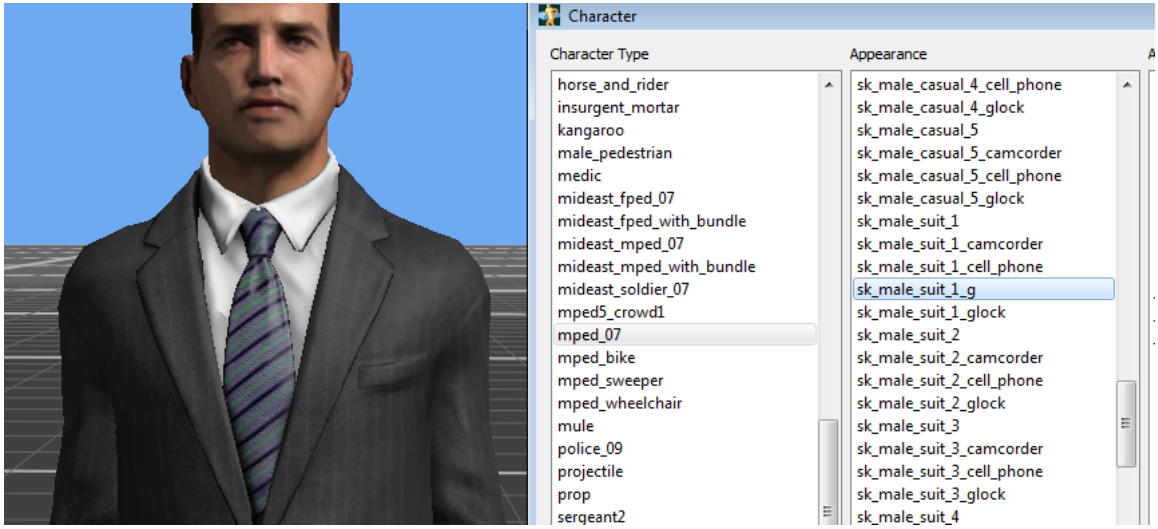
STEP THREE: CREATE YOUR NEW TEXTURE SWAP AND DECLARE YOUR NEW APPEARANCE

In the Character Viewer, Select "Add New State". Set the State Name to "male_suit_1_g" and browse for your saved png file for the new Diffuse entry. Select OK when complete.

Notice that the male_suit_1.dae is highlighted and has an asterisk. Select "Save mtl file" again, which will update the states available.

You'll need to manually create a new appearance config entry for your character. Open the file %DIGUY%/config/diguy/appearances_skinned.cfg. Create a new file called "autoload_my_custom _appearances.cfg" in the %DIGUY%/custom/config directory. Copy the multi-line entry for appearance sk_male_suit_2 into your autoload file. Change the name of that entry to sk_male_suit_1_g. Change the head back to sk_male_w1_head (or leave it as is if you like). Change the graphics_state_switch_map1 to use male_suit_1_g. Delete the graphics_state_switch_map2 line.

Restart the Character Viewer – the new appearance should now be available with a beautiful new tie:



3. Customizing Appearances by modifying geometry Tutorial

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

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

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 typically are built using OpenFlight geometry. However, "rigid" Collada files are also supported. This example assumes OpenFlight geometry.

Step 1: 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.
   (Polytrans by Okino Software www.okino.com is a good tool for doing this.)

2. Write out each LOD (up to 7) as a separate flight 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. Use meters if you can help it.)

7. Set the OpenFlight LOD switch settings to: **`switch in = 1000, switch out = 0`**.
8. Save the file in **`$DIGUY/custom/geometry/flt/`** .
9. Save the texture files (**`.rgb`**, **`.rgba`**, **`.rgb.attr`**, **`.rgba.attr`**, **`.int`**) in:

   `$DIGUY/custom/geometry/rgb/.`

Step 2: Customize the config entry

Now that the geometry is ready, it is time to customize the config entries so that DI-Guy knows about the new prop appearance.

10. The system requires a configuration file that points to the geometry that defines the new prop at each level of detail. Specify this file name as:

    `$DIGUY/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 **`$DIGUY/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 **`config`** 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
```

    NOTE*:* DI-Guy expects an OpenFlight file designation for each LOD. If you have fewer than seven LODs for your model, just use the same file for multiple LODS. Note that LOD1 is the most detailed.

11. Add or edit the file:

    `$DIGUY/custom/config/autoload_custom_appearances.cfg`
12. 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 this file as a model of what to include:

$DIGUY/config/diguy/appearances_base.cfg

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

Your customization should now work. You can test it in DI-Guy Character Viewer: select the character prop. Your new appearance should appear as a choice. Select it and you should see your new prop appearance.

Creating a custom vehicle

DI-Guy comes with a variety of vehicles appearances that you can use in the scenarios you create. 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 above, and in addition:

1. Edit your model so that the positive Z axis points up and positive X points toward the front of the vehicle. 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 1) having your shape_and_joint entry in the shapeset definition below use "y_forward" instead of "position" and 2) adding entry "item = invisible_vehicle" as an additional item in your appearance defininition

2. Put your custom shapeset in:

$DIGUY/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 the following file as a model of what to include:
  $DIGUY/config/diguy/actor_vehicle_shapesets.cfg

For example, to add a vehicle called "my_bmw" to the system, place the following block of text in the config 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
```

3. Create a custom appearance entry in **$DIGUY/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: select the character vehicle. Your new appearance should appear as a choice. Select it and you should see your new vehicle appearance.

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

Creating a custom human: Segmented characters using OpenFlight

You can create new customized human appearances for DI-Guy. Unlike the vehicle and prop, you will have to identify which characters will use the new appearance, which actors to associate with the appearance, and properly name all the groups in your OpenFlight model so that they properly get hooked up to the character's link/joint hierarchy.

To add a new human appearance to DI-Guy, perform the same basic steps as the prop example, and in addition:

1. Choose a Character Type that you want to add the appearance to. Pick a character whose list of actions best matches your needs.

2. Be sure that your model has the positive Z axis points up and the character is facing the positive X direction.

3. Your OpenFlight file will need to contain multiple geometry groups with names that match the first value in each shape_and_joint key/value line in your shape_set_definition (see below). For example, you will need a pelvis group, a torso group, a head group, etc. Each group should have

its origin be its parent joint location. For example, the origin of the head group should be located at the middle base of the skull where the cervical joint is located.

4. Choose an actor to associate your geometry with. An actor defines the skeleton your geometry will be hanged on. Potential actors include Beth, Bill, Greg, Gregg, Ken, Lance, Yumiko, etc. In the `$DIGUY/config/diguy` directory you will find files of the form `actor_<actorname>.cfg` and `actor_<actorname>_links.cfg`. The skeletal offsets can be found in the links file. Your best choice for actor will be to use the actor associated with the Character Type to which you want to add your new appearance. For example, if you want to make a new appearance for Character Type soldier_07, open the char_soldier_07.cfg file. You will see that this character uses "greg" as the default actor.

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

5. Put your custom shapeset in

   `$DIGUY/custom/config/autoload_actor_<actorname>_shapesets.cfg`.

   - If this file already exists, append a **`shape_set`** entry for the new human appearance.
   - If the file does not exist, create a new one, using the corresponding actor shapeset file as a template.

For example, to add a human appearance called "handsome_dude" to the system, place the following block of text in the config file:

```
shape_set handsome_dude
  actor = bill
  is_base_shape = 1
  filename = handsome_dude_LOD1.flt
  filename = handsome_dude_LOD2.flt
  filename = handsome_dude_LOD3.flt
  filename = handsome_dude_LOD4.flt
  filename = handsome_dude_LOD5.flt
  filename = handsome_dude_LOD6.flt
  filename = handsome_dude_LOD7.flt
  include common/shapes_and_joints_human.cfg
```

Note that the include common/shapes_and_joints_human.cfg is as follows:

```
shape_and_joint = pelvis base
shape_and_joint = torso  back
shape_and_joint = head   cervical
shape_and_joint = uarm_l shoulder_l
shape_and_joint = larm_l elbow_l
```

```
shape_and_joint = hand_l wrist_l
shape_and_joint = uleg_l hip_l
shape_and_joint = lleg_l knee_l
shape_and_joint = foot_l ankle_l
shape_and_joint = uarm_r shoulder_r
shape_and_joint = larm_r elbow_r
shape_and_joint = hand_r wrist_r
shape_and_joint = uleg_r hip_r
shape_and_joint = lleg_r knee_r
shape_and_joint = foot_r ankle_r
```

6. Create a custom appearance entry in **$DIGUY/custom/config/autoload_custom_appearances.cfg** as follows:

```
appearance handsome_dude
    actor = greg
    item = handsome_dude
    character_types_data_table = available_to_male_pedestrians
    quality_bias = 6
    appearance_map = male/adult/white / -/-/-
```

Your customization should now work. You can test it in DI-Guy Character Viewer: select any of the male pedestrian Character Types; you should see "handsome_dude" in the Appearances list. Select it and you should see your new human appearance.

Creating new weapon/equipment combinations

It is really easy to create new appearances from existing characters and weapons. For example, the weapon "pk_machine_gun", is used by appearance "taliban_2". What if you would like a soldier to have an appearance where he wears a bdu (battle dress uniform) and is holding the machine gun? To make that appearance available, create or append to the file `$(DIGUY)/custom/config/autoload_custom_appearances.cfg` the following appearance:

```
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
 character_type = soldier2
 character_type = soldier_precise
 character_type = soldier_qk
 character_type = cbd
```

```
 character_type = sergeant
 character_type = sergeant2
 character_type = soldier_signals
 quality_bias = 5
 appearance_map = male/adult/white / usa/army/pk74
```

The file `$(DIGUY)/config/diguy/appearances_human_military_soldier.cfg` contains the “taliban_2” appearance. To create the “pdu_bk” appearance, it was as simple as copying the taliban_2 appearance and then replacing the first item, “taliban_2”, with “bdu”.

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

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

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

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. Following is a decription of the process we recommend for creating a skinned custom character in DI-Guy, a new capability in DI-Guy 10, however it should not be considered not the only way to add skinned models to DI-Guy.

Create (or better yet copy) a model that has a DI-Guy Compatible Skeleton

All DI-Guy actors are built Z up and X forward. Because the standard bones and biped in 3D Studio Max do not follow this convention, you need to be careful that your model uses a DI-Guy compatible skeleton. A sample skeleton is provided in `$(DIGUY)/geometry/max/greg_object_bones_skeleton.max`.

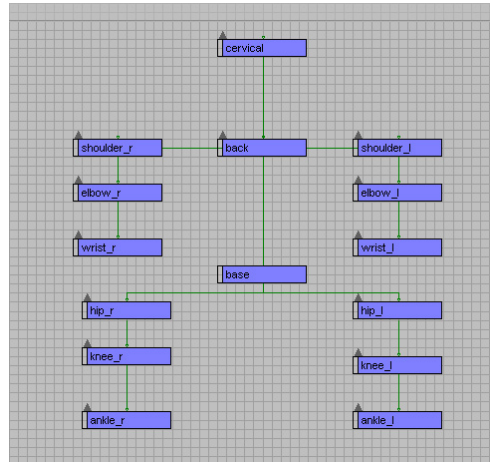
Further information on the skeleton is provided in the next few sections for the reader's benefit to explain what is found in the max file.

Defining links for a DI-Guy Compatible Skeleton

Links for a DI-Guy compatible skeleton should be positioned in world space based on the offsets listed in `$(DIGUY)/geometry/config/diguy/actor_greg_links.cfg`, except that all shapes are positioned directly from the origin (in other words, in world space) rather then starting from their parents' local coordinate systems. The image below is how the shapes should look in 3D Studio Max prior to setting up the hierarchy and locally rotating the arms and legs.

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 Skelton

The hierarchy links are then 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 above, so be sure to use those names and double-check your spelling.

Set Object Properties of a DI-Guy Compatible Skeleton

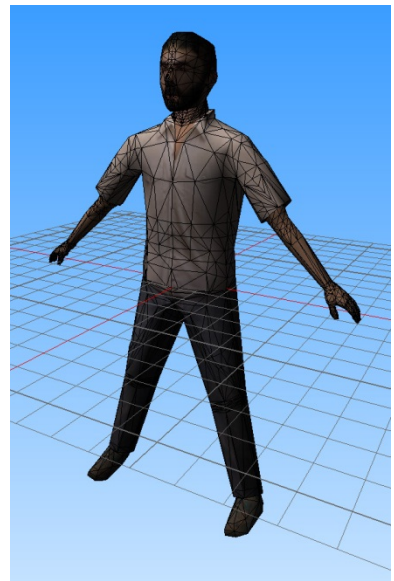
Select all of the objects and then select Animation > Bone Tool > Bone Properties from the toolbar. Set the flag "Bone On".

Importing an existing Human Model into 3D Studio Max

Many DI-Guy users have custom characters they wish to use in DI-Guy. This section describes how to prepare a model for importing into 3D Studio Max for use by DI-Guy.

Prepping the model for import

In your modeling tool:



1) Rotate the models arms and legs away from the body and other parts to a configuration similar to the image at the right.. This is done so vertex can more easily be selected in the skinning process and is not possible in the default DI-Guy convention of arms and legs straight down.

2) Combine meshes into a single object.

3) Remove unnecessary geometries not required for the end skinned model.

4) Make sure there are no t-intersections on any faces as t-intersections will result in tears and gaps when animated.

5) Scale the model to meters.

Importing the Model into 3D Studio Max

After importing into 3D Studio Max:

1) Rename the imported object to “body_lod1”. DI-Guy skinned models supports up to seven levels-of-detail (LOD) within the same file by defining additional objects and using the naming convention “body_lod1” through “body_lod7”.

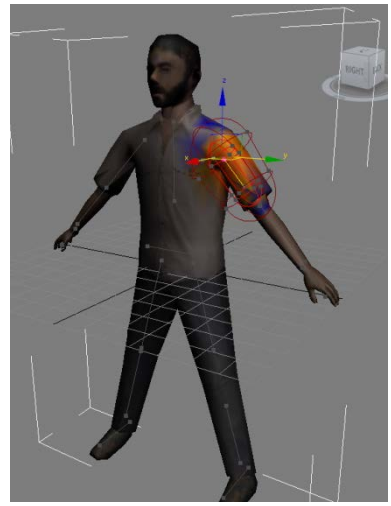
3) Delete any remaining groups.

4) Select all vertices and weld them together using the default factor of 0.0001.

5) Select all faces and verify they all use material ID 1.

6) Finally, add smoothing groups to the faces to calculate normals.

Align the DI-Guy Compatible 3D Max skeleton with the imported geometry.



Before skinning the object skeleton, the arms and legs meshes of the imported geometry need to be carefully rotated and otherwise manipulated to match the underlying DI-Guy skeleton. Make sure this is done precisely to ensure that the center-of-rotation of the skeleton's joints match the geometry. These rotation needs to be done exclusively on the mesh so that the base skeleton rotation is not altered. As mentioned, the final 3D model and skeleton should have the arms and legs rotated away from the body and other parts to make it easier to skin. HINT: Select "Local" under Reference Coordinate System then start to rotate the geometry.

Skin the Model in 3D Studio Max

It is time to begin the skinning process. Skinning refers to the process of associating polygonal vertices with one or more bones. Developers familiar with other 3D programs such as Autodesk Maya will recognize this as the first part of the rigging process with a bone skeleton. As DI-Guy characters are driven by almost entirely by motion capture, a full animation rig is not necessary.

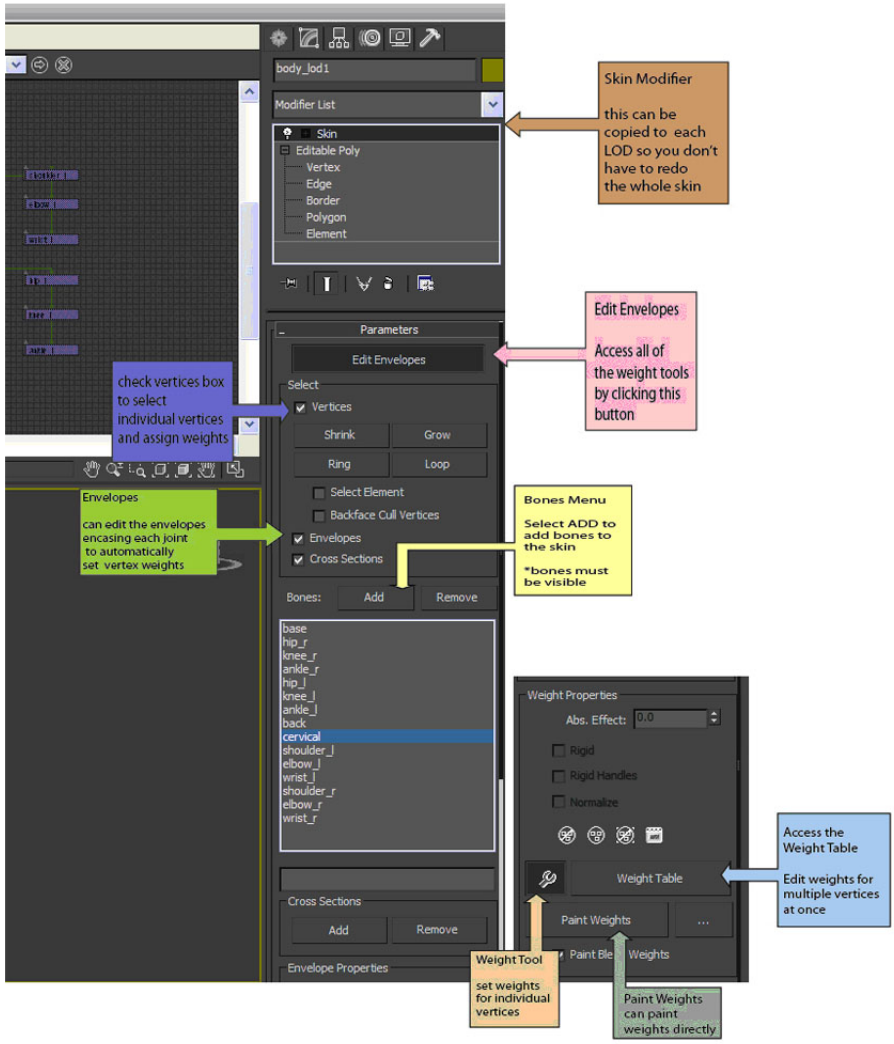
Select the imported human object and under the Modifier List add "Skin". Then add all of the bones from the object skeleton to the skin modifier.

Hide the Object Skeleton

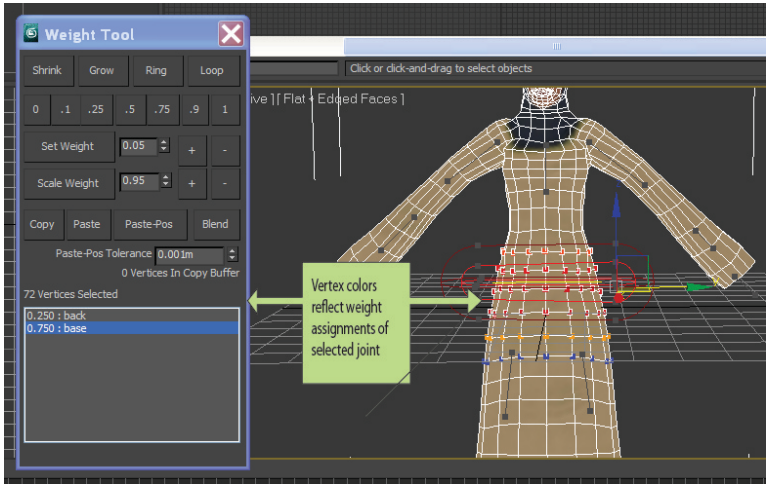
Now hide the objects that make up the skeleton so they will not get in the way of editing. The proper way to do this is via the Schematic View. Select all of skeleton objects, then right click and select "Object Properties…" under "Interactivity", and then check the box next to "Hide".

Edit the Envelopes

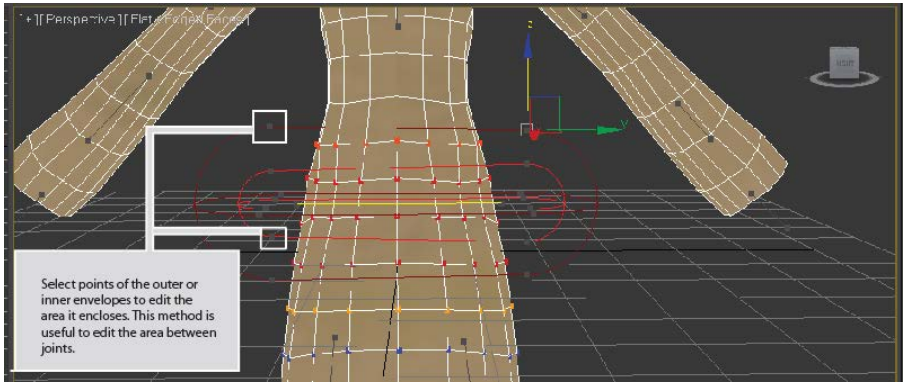
Under the Skin Modifier Parameters menu, you should notice that there is an "Edit Envelopes" button. Press this in order to edit the assigned paint weights. There are a variety of ways in which to assign paint weights to your model's vertices.



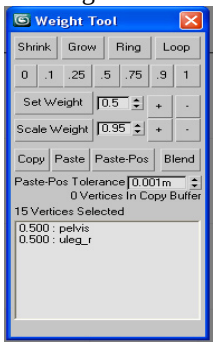
Use of the Weight Tool: The weight tool and weight table are by far the most accurate tools as they allow the user to set paint weights for selected vertices with a mathematical value. This tool will also be familiar to Maya users, as it is similar to the weight table. These tools provide a precise method for assigning weights to vertices.

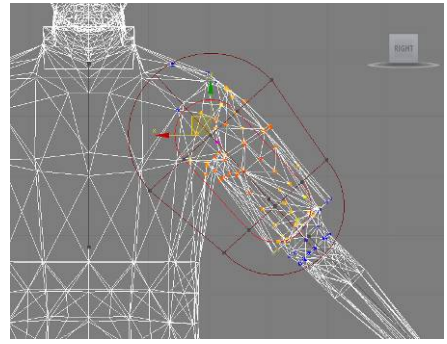


Use of the Envelope Tool: The envelopes are automatically generated around each piece of the skeleton when you add the bones to the skin. 3ds Max generally does a good job of placing the envelopes, but occasionally the user will need to edit them. The envelopes are useful for areas where vertices are shared by multiple joints, such as elbows or knees.



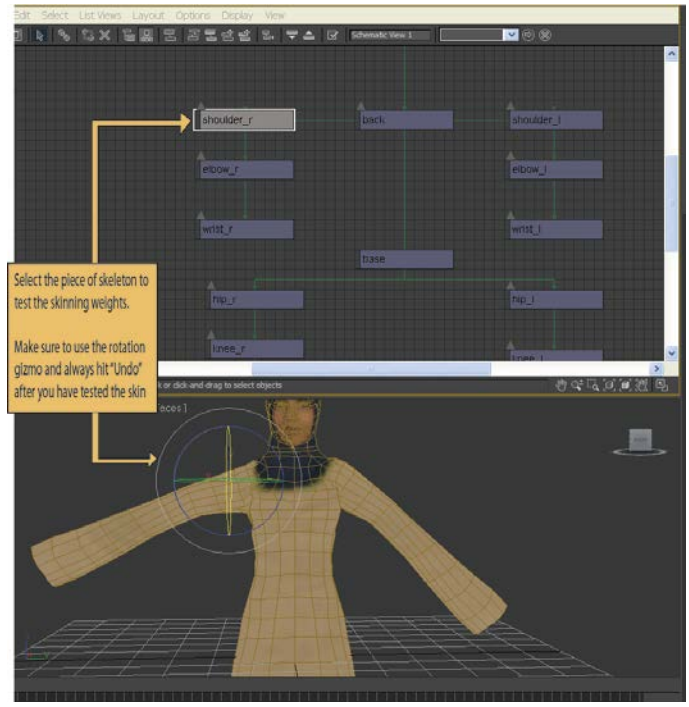
Select and then start editing the envelopes. Move and scale the influence and drop off all envelopes to split the weight of the vertices across center of the joints. Do this so that when the model is animated the geometry along the joints will look correct. A way to fine-tune the envelope's influence is to manually select the vertices and use the "Weight Tool" window to set the weight or split it between multiple joints. Please note this entire process can be time consuming and may take multiple iterations to get everything looking and working correctly.



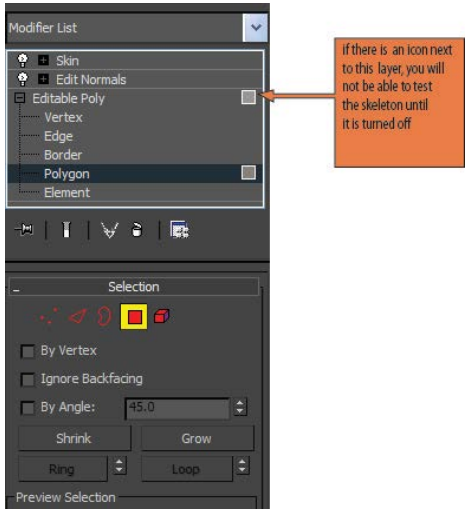


Testing the Skin:

In order to test the skin, open the schematic view and select a piece of the skeleton that you wish to test. Utilize the rotation gizmo (hot key: E) to test and make sure you hit "Undo" so that your bone isn't permanently moved into that position.



Important Note: If you have edited anything below the skin modifier (edit poly, edit mesh), this will show up as an icon next to that layer. You will not be able to edit the skeleton until this has been turned off, by clicking on the component that is highlighted.



LOD Creation in 3ds Max

Copy "body_lod1" by selecting it and hitting "ctrl+v" or by going to Edit → Clone. 3Ds Max will prompt with "Clone Options". Choose "copy" and rename the new object "body_lod2"

The skinned body requires LOD1, LOD2, LOD3 and LOD4. For LOD2, use the MultiRes modifier to decimate the model by 50% of its polygons.

Hide "body_lod1" and select "body_lod2" (the geometries should still be identical). Decimate the polygons to the correct count for LOD2. The chart below gives an approximate estimate of the poly count per LOD:

LOD #	Poly Count:
LOD1	5000
LOD2	2500

LOD3	1000
LOD4	500

Once you are satisfied, Copy the skin modifier from body_lod1 and paste into the modifier pallet each of the LOD's you have created.

HINT: When decimating a model using the "MultiRes" modifier, the normals can get messed up. This is easily rectified by adding an "Edit Poly" modifier on top and selecting all of the polygons and adding the smoothing groups to "6". If this doesn't fix the problem, adding a "Edit Normals" modifier, selecting all the normals and hitting "unify" should fix the issue.

Create LOD3 by copying LOD2 and attach the head and hands by File>Import>Merge the head and hands (LOD2 for both) from the head.max file. For LOD3, decimate the model by 60%. Use the Unwrap UV modifier to edit the head and hand UVs into a position such that they can be shared on the body_diffuse.png/ body_normals.png / body_specular.png. LOD3 will retain the Skin modifier by having been copied from LOD2.

Finally create LOD4 by File>Import>Merge "body_lod4" from an existing model. Use the Unwrap UV modifier to edit the body_lod4 UVs into a position such that they can be shared on the body_diffuse.png/ body_normals.png / body_specular.png. LOD4 will retain the Skin modifier by having been imported from a working DI-Guy model.

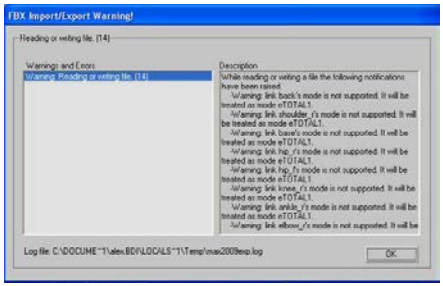
Name material in a DI-Guy consistent manner.

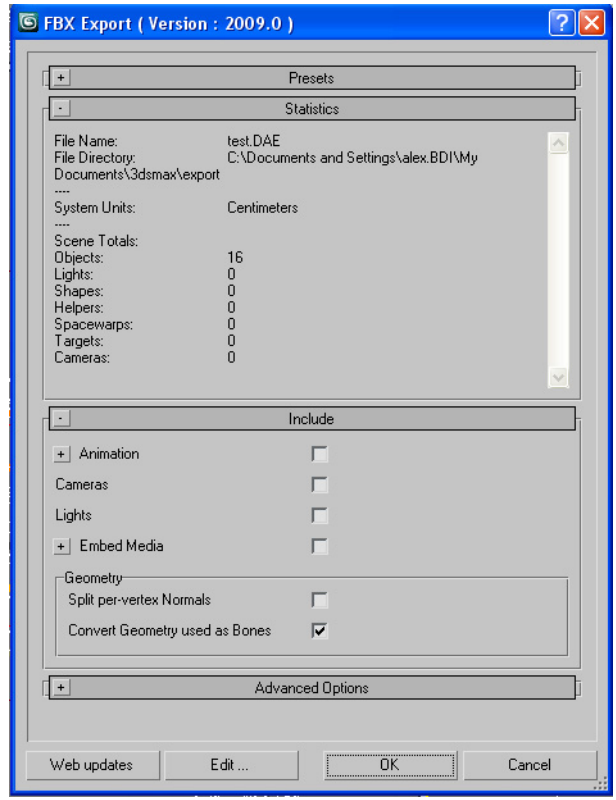
Remember the material will be looked up later in DI-Guy configuration files. DI-Guy supports geometry with one material which may mean that the head.max file needs to be cleaned of any multi-materials.

Export the New Human Model as a Collada File

After editing all of the joint envelopes and verifying that the joint rotations look good, it is now time to export this model in the Collada format. Make sure all of the objects that make up the base skeleton are hidden prior to exporting. In 3D Studio Max, select *File > Export > Save as type: Autodesk Collada (*.DAE)*. Make sure in the FBX Export window that the "Convert Geometry used as Bones" flag is checked, then press "OK". (This was tested using FBX Plug-in 2009.0, Build Number 20080212)

A “FBX Import/Export Warning!” window with a list of warnings regarding the hidden object skeleton will then appear; these warnings can safely be ignored.





A Note on Files

With DI-Guy 12 more files are generated, the new .dae files allow for more use of texture swapping to create and combine new appearances, while the multiple textures enable the more advanced and realistic appearances achieved with DI-Guy 12. A typical set of output files looks like:

head.dae
body.dae
hands.dae
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, use the MultiRes modifier to 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

Copy the new Collada model into the `$(DIGUY)/custom/geometry/dae` directory and make sure to copy the rgbs and pngs this model uses into `$(DIGUY)/custom/geometry/rgb` and `$(DIGUY)/custom/geometry/png` respectively.

Create a New DI-Guy Shape Set and Appearance

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

1. The file `autoload_shape_set.cfg` loads the shape set that stores the Collada .dae file and assigns it to an actor. In the example code below, a new shape set "sk_arab_male_1" (note that 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
```

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

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

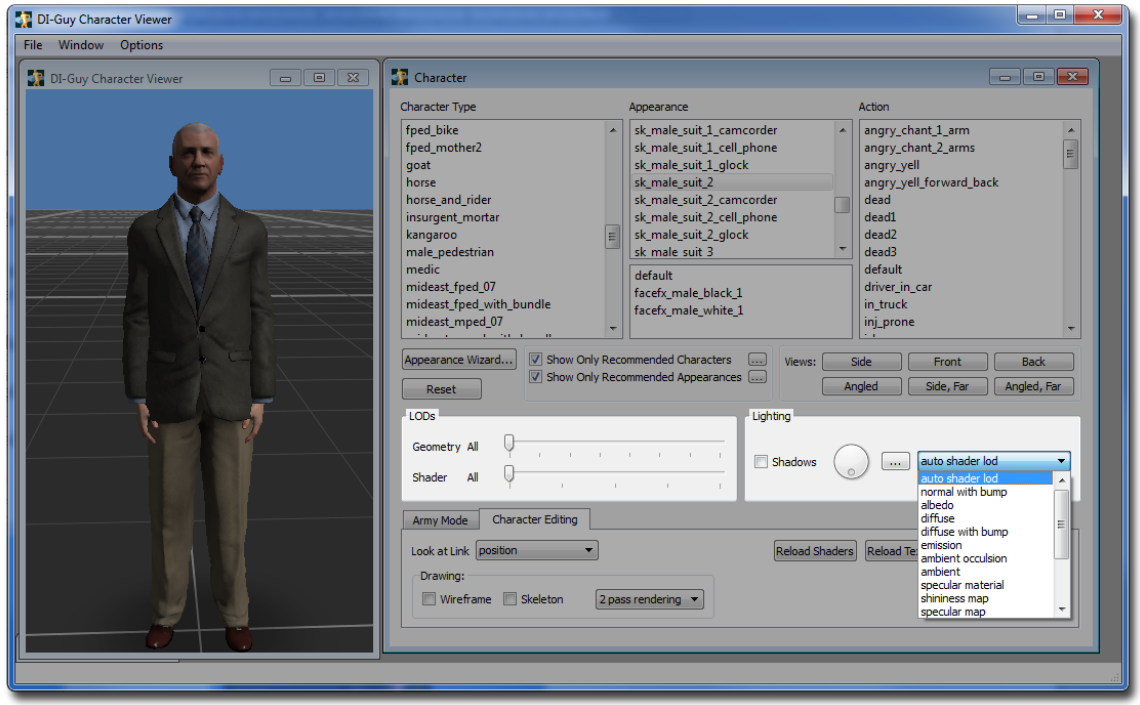
Test New Content and Create Material Files in DI-Guy Character Viewer

To test your new appearance, launch the DI-Guy Character Viewer, and then select the character type and appearance corresponding to your new model. You should be able to see your new model, and you can then see how it looks when performing Actions.

At this point, create the mtl.cfg file which will point all of the various .dae files to their respective .png files by opening the Geometry Viewer. Under the Material States, click Edit and browse for the appropriate files. Click “Save Mtl File” to save the mtl.cfg file which can now be found in the dae folder.

4. Character Viewer

DI-Guy 12.0 Character Viewer has substantial improvements including the ability to author and test materials. The following outlines the new features of this convenient tool.



LODs
Geometry LOD slider • Shader slider

Lighting
Rotating light source • Shadows tick box • Shader pull-down menu

LODs

Shader Slider

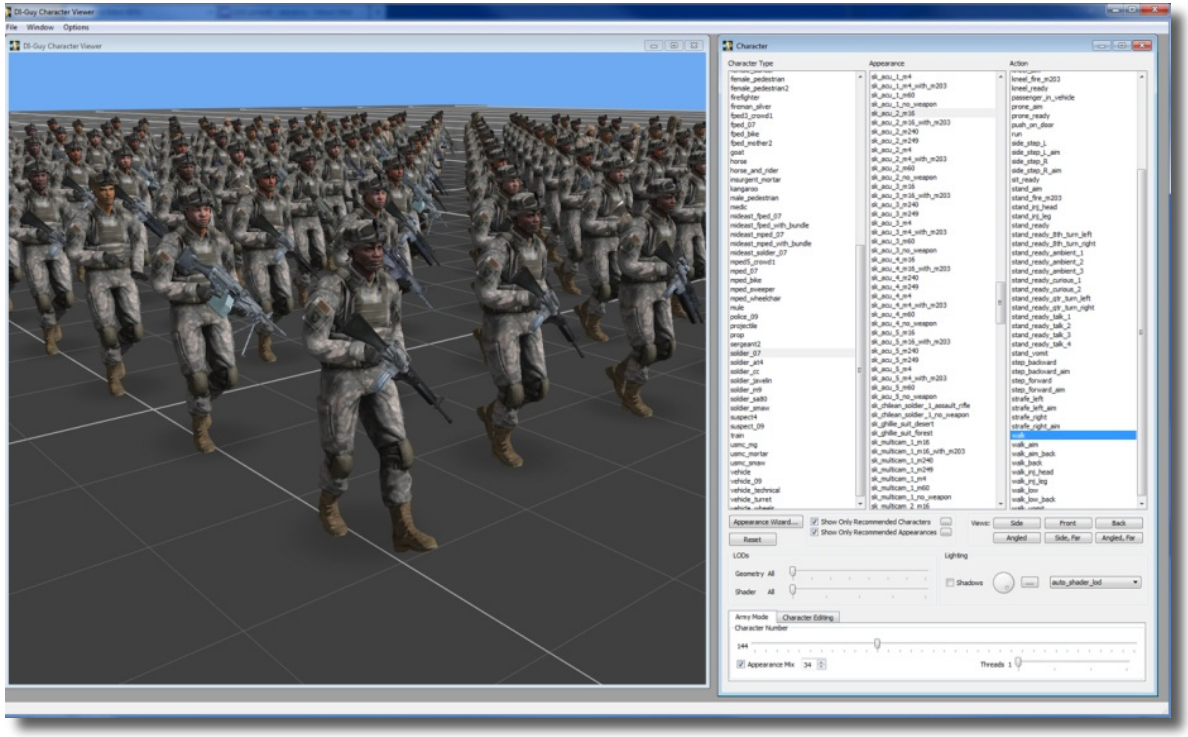
This slider scrolls through the available shaders of a character, previewing the quality and performance difference between real time normal maps, per-pixel and per-vertex lighting solutions.

Character Editing

New character tools allow end users to quick see the results of external content changes. Reload Shaders, Textures, Objects, and Appearances Buttons ~ In previous versions the user has to close and restart Character Viewer, but now with one click, Character Viewer automatically updates the character with the new information, These buttons make for easy viewing of any customizations a user has made to the shaders, geometry, and/or the material files.

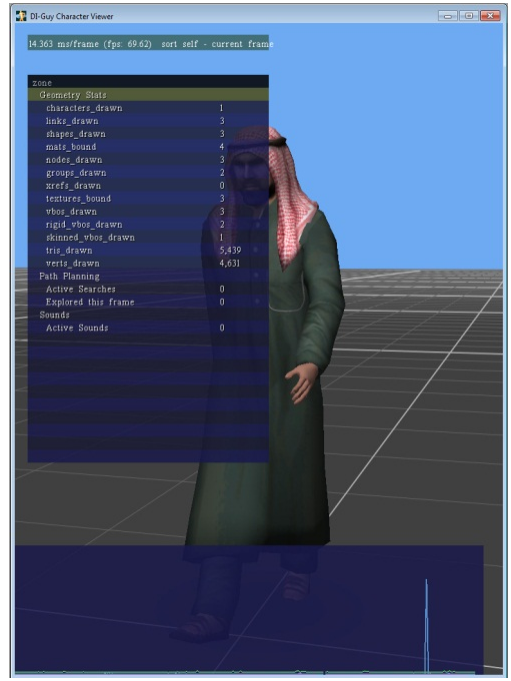
Army Mode

This feature allows the user to increase the character number using the Character Number slider. They can also vary the appearances of their "army" by checking the Appearance Mix tick box. And preview multi-threaded performance. Both the character editing and army mode UI's are available when the Options>Advanced menu item is set.



Geometry Stats Profiler

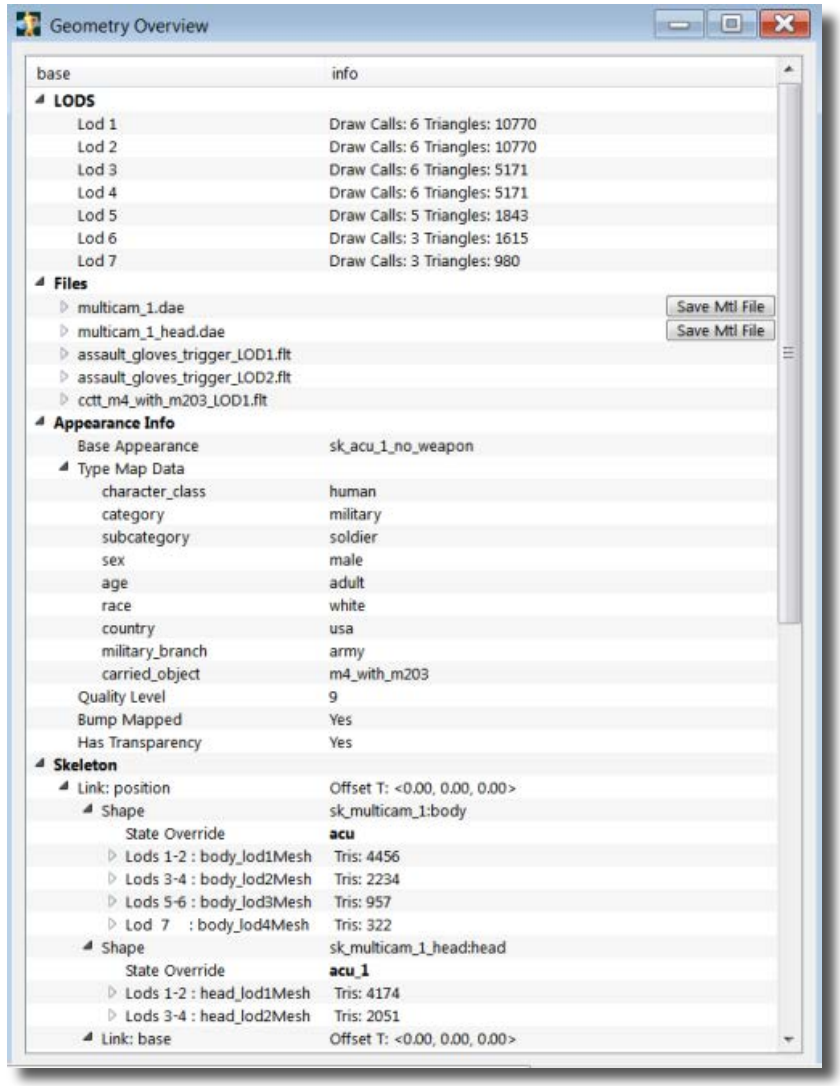
The Geometry Stats window can be accessed by clicking Ctrl-P which allows user to view geometry information, as well as real-time performance level reader. This is a convenient way of seeing the performance quality while using Army Mode.



Select Ctrl-P to toggle between different modes: **Hierarchical • Hierarchical with Graph • Geometry**

Geometry Viewer

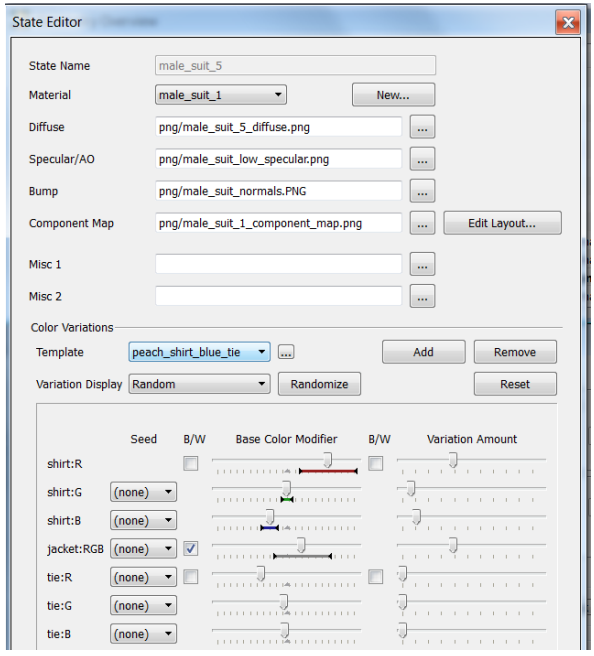
A brand new feature, the Geometry Viewer provides detailed information on LODs, Files, Appearance Info, and Skeleton, presenting the end user with a straightforward way of seeing the unified result of many configuration files. The Geometry Viewer GUI provides new tools to edit material states and swaps, giving the end user tools for easily modifying and adding new character appearances. Under the Each File item, is the Material States item. There the user may Add and Edit new states which allows user to choose the various maps that contribute to the final model, i.e.: diffuse, normal, and specular maps. Edited and saved .mtl files reside in the dae folder.


Geometry Overview
base
info
LODS
Lod 1
Draw Calls: 6 Triangles: 10770
Lod 2
Draw Calls: 6 Triangles: 10770
Lod 3
Draw Calls: 6 Triangles: 5171
Lod 4
Draw Calls: 6 Triangles: 5171
Lod 5
Draw Calls: 5 Triangles: 1843
Lod 6
Draw Calls: 3 Triangles: 1615
Lod 7
Draw Calls: 3 Triangles: 980
Files
multicam_1.dae
Save Mtl File
multicam_1_head.dae
Save Mtl File
assault_gloves_trigger_LOD1.flt
assault_gloves_trigger_LOD2.flt
cctt_m4_with_m203_LOD1.flt
Appearance Info
Base Appearance
sk_acu_1_no_weapon
Type Map Data
character_class
human
category
military
subcategory
soldier
sex
male
age
adult
race
white
country
usa
military_branch
army
carried_object
m4_with_m203
Quality Level
9
Bump Mapped
Yes
Has Transparency
Yes
Skeleton
Link: position
Offset T: <0.00, 0.00, 0.00>
Shape
sk_multicam_1:body
State Override
acu
Lods 1-2 : body_lod1Mesh
Tris: 4456
Lods 3-4 : body_lod2Mesh
Tris: 2234
Lods 5-6 : body_lod3Mesh
Tris: 957
Lod 7 : body_lod4Mesh
Tris: 322
Shape
sk_multicam_1_head:head
State Override
acu_1
Lods 1-2 : head_lod1Mesh
Tris: 4174
Lods 3-4 : head_lod2Mesh
Tris: 2051
Link: base
Offset T: <0.00, 0.00, 0.00>

Variation System Editing

DI-Guy 12.5 introduces a GPU based variation system where a special texture map called a component map defines the different areas on a characters body. A tiny per character color shift texture map is then used to create unique character appearances.

The material file contains a list of variation templates for each material state. Each template contains information dictating how much each item of the character should color shift.



5. Creating a custom character_type

Sometimes being able to customize the appearance of an existing character is not enough. This is most often the case when the user wishes to animate a joint hierarchy that doesn't match any of the existing DI-Guy characters. CREATING A CUSTOM CHARACTER TYPE IS FOR ADVANCED USERS ONLY.

A custom character_type example: Vehicle with turret

Following we will create a new character_type for use in DI-Guy, called "vehicle_turret". It will use the existing "vehicle" character_type as a template, but include an extra 3 degree-of-freedom link to allow a turret to also be animated.

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

There will be 2 basic steps for creating the character_type:

- Create the config files that define the new character_type, and test it using DI-Guy Character Viewer. Keep refining until all load errors disappear and see your character in the viewer.

- Verify that your new joints work by modifying the OpenGL pose example in programming examples.

Remember that a secondary goal when customizing DI-Guy config files is to be sure to not take steps that may mask future changes to DI-Guy data. Thus always copy a standard DI-Guy config file into your custom directory, add more content, and then save it with the same name.

Step One: Creating the config files

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

- `$(DIGUY)/custom/config/autoload_vehicle_turret.cfg`
- `$(DIGUY)/custom/config/autoload_vehicle_turret_appearances.cfg`
- `$(DIGUY)/custom/config/autoload_vehicle_turret_loader.cfg`
- `$(DIGUY)/custom/config/common/variables_vehicle_turret.cfg`
- `$(DIGUY)/custom/config/diguy/char_vehicle_turret.cfg`
- `$(DIGUY)/custom/config/diguy/actor_vehicle_turret.cfg`
- `$(DIGUY)/custom/config/diguy/actor_vehicle_turret_shapesets.cfg`
- `$(DIGUY)/custom/config/diguy/actor_vehicle_turret_links.cfg`

Let's examine each file separately and understand its contents:

File #1 - autoload_vehicle_turret.cfg

This is the top level file for the vehicle turret. It simply 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
```

File #2 - 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 `$(DIGUY)/config/common/appearances_base.cfg.`

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

File #3 - 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 `$(DIGUY)/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
```

File #4 - common/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 `$(DIGUY)/config/common/variables_vehicle_wheels.cfg.`

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

File #5 - char_vehicle_turret.cfg

This file defines the vehicle_turret character. The file `$(DIGUY)/config/diguy/char_vehicle.cfg` was used as the template for this file. Note that we will use the standard DI-Guy vehicle's action table (we will drive the turret programmatically in step 2).

```
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
```

File #6 - 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 `$(DIGUY)/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
```

File #7 - actor_vehicle_turret_shapesets.cfg

We define a single shape_set 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've named the shape_set "silly_test"! The shapeset references the firefighter OpenFlight files. Note that the group "base" will be associated with the joint "position", while the group "head" will be associated with the joint "turret". See the preceding "Customizing Appearances" section for more discussion of 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
```

File #8 - actor_vehicle_turret_links.cfg

This file defines the link/joint hierarchy of vehicle_turret. The file is the same as `$(DIGUY)/config/diguy/actor_vehicle_links.cfg` except that it appends an extra link called "vehicle_turret_turret". Note that 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.

Step Two: Modify the Pose programming example to exercise your new joint

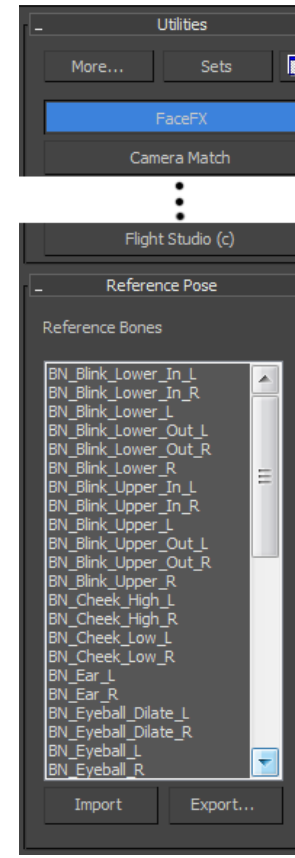
Edit the $(DIGUY)/programming_example/diguy_ogl/pose/pose.cpp file. Make the following changes:

- Change the three static back index variables to be turret index variables.
- Change the bdi_log_stdout_enable setting to “BDI_LOG_INFO”.
- Change the character type in the create_character call from “soldier” to “vehicle_turret”.
- Change the action in the force_action call from “stand_ready” to “still”.
- Change the three get_var_index calls to retrieve your turret variables instead of back variables.
- Initialize the three turret variables to zero.
- Change the three weight variable indices from back indices to turret indices.

- In the glut_idle_callback() function, set all three turret variables equal to sin(t). Remove any reference to the back variables.

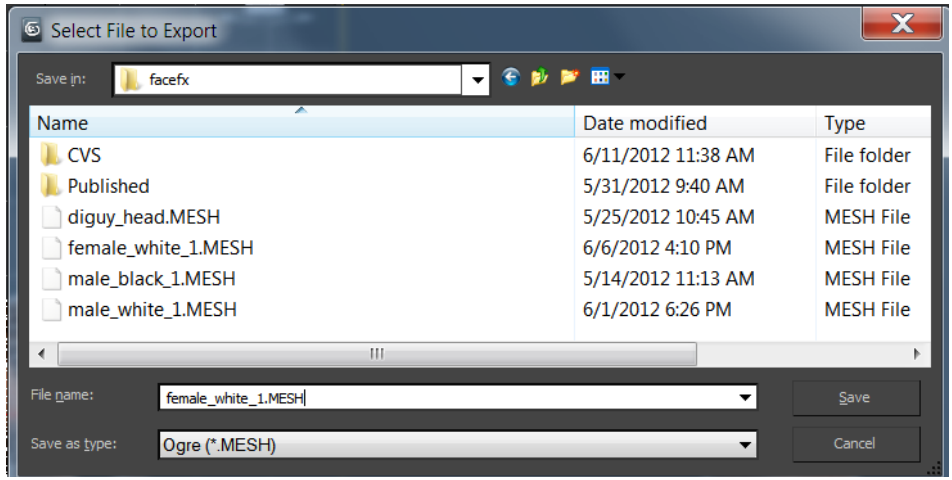
Recompile and run the example. You should see a fireman with an extra head that rotates in an unusual fashion. If you have problems, check that during initialization it successfully lists the three turret variables, and that your indices are not -1.

6. Creating a FaceFX Head Appearance



Creating and Exporting Data from 3DS Max

- Start with a sample facefx model found in $DIGUY/geometry/facefx there should be a number of .max files to use as a base.
- Modify Geometry, textures etc. 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.
- Verify that the character can go through all the emotions and poses in the animation data.
- Pick the FaceFX utility plug-in (see image) Click C**reate Actor** and give the character a reasonable name that will match the associated head appearance.
- Click **Export** on the Reference Bones panel and select all bones with the BN_ prefix.
- Click **Batch Export…** on the Nodes Panel pick the following script that has a list of poses and the keyframes they happen at $DIGUY/geometry /facefx/diguy_batch_export.txt
- Click **Save Actor** and place the new actor file in $DIGUY/custom/geometry /facefx
- Select the head of the character and do 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 $DIGUY/custom/geometry /facefx/



- Note: 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 into the $DIGUY/custom/geometry /dae directory.

Setting Up and Publishing a New FaceFX Actor

- Load FaceFX Studio

- Open the actor file that was saved in $DIGUY/custom/geometry/facefx

- You should now see a the face that was exported as an ogre mesh file.

- If you switch to the face graph view there will be a unstructured node graph that doesn't have constraints or connections. Select Actor>Sync to template and import $DIGUY/geometry/facefx/diguy_head_face_graph.fxt this should now give you a character that can animate and lipsync to audio.

- Follow the FaceFX Studio manual on how to author animation and create animsets for use in DI-Guy

- Run the command File>Publish to Game and publish the new actor in $DIGUY/custom/geometry/facefx/published

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 2 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 compatable 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 will apply to the characters hand item and keep the color of the hands in sync with the color of the head. This is applied after any state switches in the body appearance.

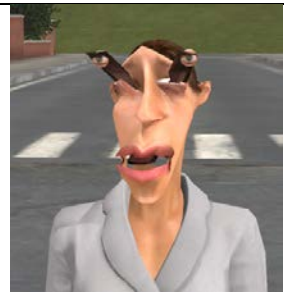
Note: the head appearance functionality isn't limited to FaceFX, non-facefx heads can be made swappable as well, the shape_attachment_data should just not specify "facefx"

Gotchas:

- 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 $DIGUY\config\common\shader_matrix_index_tables.cfg. If you intend to use a different naming convention for your face bones you MUST add a new shader_matrix_index_table entry to a custom version of shader_matrix_index_tables.cfg.

- Number of Joints - we currently use a max of 50 link matrixes in shaders, if you intend to build a expressive face with more joints then 50 you will need to 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's been unclear where in a 3 application/2 file format -export-pipeline the problems arise. An illustration of the issue:


Result of an incorrect scale factor


Correct scale factor

To fix this add a bit of meta data to $DIGUY\custom\config\diguy\facefx.cfg

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

- You may also need a stiff drink to counteract the fears of a bug-eyed alien invasion.Scaling of heads in Max is also very tricky, it may require scaling the BN_Head node, then disconnecting all children and scaling the bones back to a scale factor of <1.0,1.0,1.> and then renomalizing 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's usually nessassary to reset the skin by checking and unchecking **Always Deform** in the advanced section of the Skin Modifier.

7. Customizing scene objects, motions, textures, and sounds

Customizing scene objects

To add custom scene objects (terrains), copy the OpenFlight Format database to the $**`DIGUY/custom/geometry/sets`** directory.

Customizing motion

You can customize motion using the *DI-Guy Motion Editor*. See the *DI-Guy Motion Editor User Guide* for details.

Customizing texture

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

Textures use the Silicon Graphics RGB or RGBA or PNG format.

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

Customizing sound

To customize textures, edit an existing sound and save it with the same name in the **`custom/sfx`** directory.

You can also simply add new sounds to this directory.

A Reminder on Derivative Work

Any DI-Guy content (whether it be a model, texture, motion, etc) 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 *Boston Dynamics* to discuss licensing for other uses of derivative models.