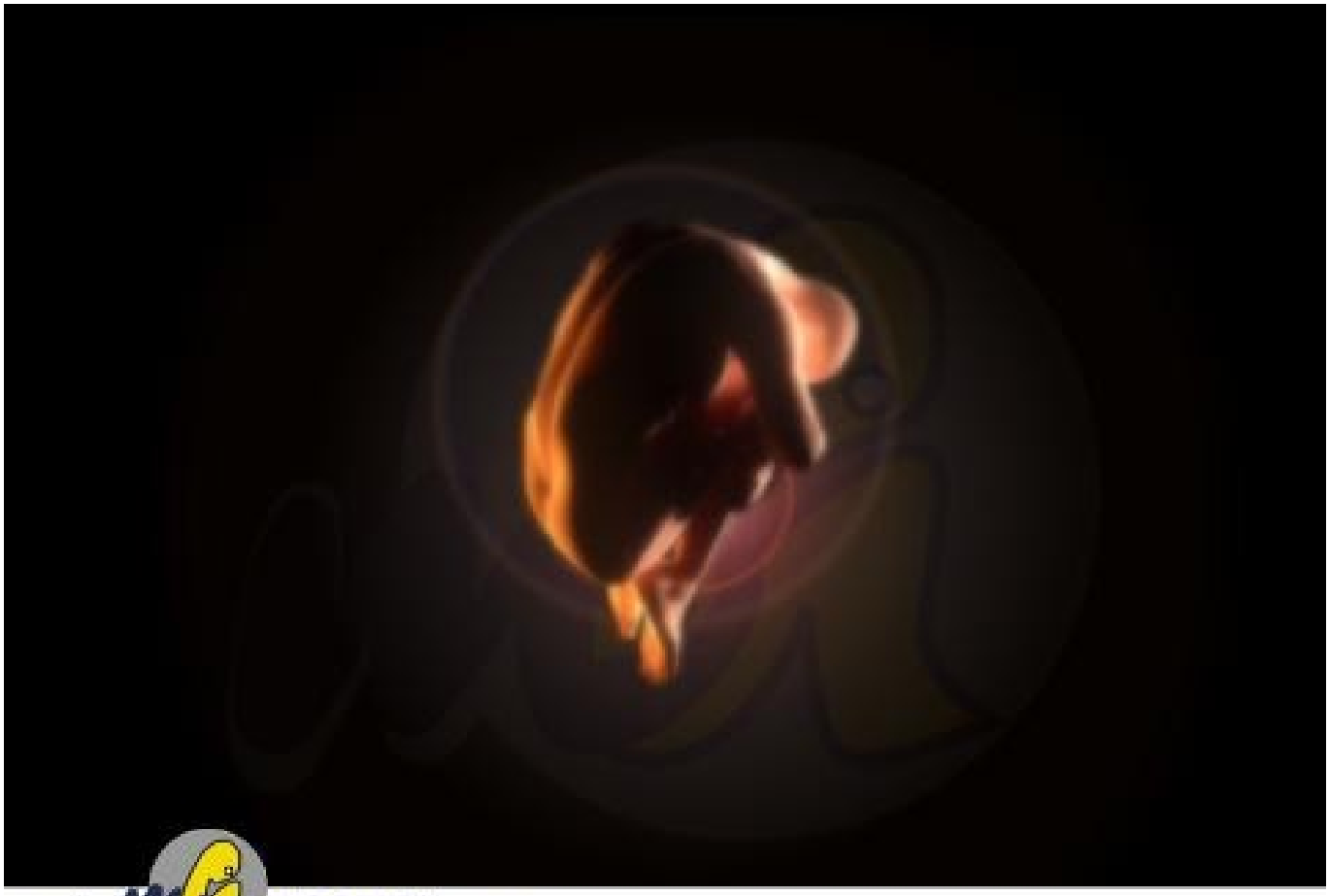


MakeHuman© - 1.0 - Pre-Alpha

User's Guide

Released 1st March 2009

Table of Contents

1. Introduction

This Pre-Alpha release is intended to provide users with a technical preview of the features that have been redesigned following feedback resulting from the 0.9.1 release candidate of Nov 2007. More complete releases are expected to follow soon. The purpose of this Technical Preview is to encourage feedback on the features and their useability via the MakeHuman community forums. If you take a few moments to contribute to the forums your feedback will help the team to improve future versions of the MakeHuman© application . This is an Open Source project and depends entirely upon community members contributing their time and ideas for free.

MakeHuman© is a free interactive modelling tool for creating custom 3D human characters. These characters can be modelled very quickly and can then be exported and used with many other modelling and rendering programs to incorporate realistic human figures into computer generated images and animations. The export feature supports the 'obj' format (amongst others), which is widely compatible with other renderers and modelling software. Features that make this software unique include the tetra-parametric GUI© and the Natural Pose System©, for advanced muscular simulation.

The home page for MakeHuman© is at http://www.makehuman.org/ which provides links through to download software and updates, to the MakeHuman community forums and other community resources. The MakeHuman project is an open source project hosted on sourceforge at http://sourceforge.net/projects/makehuman/ . Documentation for the project, including the FAQ (Frequently Asked Questions list) is maintained on the MakeHuman Wiki at http://makehuman.wiki.sourceforge.net/ .

This document is the Users Guide for the MakeHuman product and covers standard installation and use of MakeHuman. This document is maintained on the MakeHuman Wiki at http://makehuman.wiki.sourceforge.net/UG_Introduction. When accessing this site, be aware that the 'active' Wiki version of this document corresponds to the 'next' release of MakeHuman that is currently under development and may differ slightly from the Users Guide that corresponds to the version of MakeHuman that you currently have installed.

A 'Quick Start' document is also available at http://makehuman.wiki.sourceforge.net/Quick_Start, which covers many of the same topics as this Users Guide, but in a more succinct format, that may be more suitable for those accustomed to not reading documentation. MakeHuman is designed to be consistent and intuitive to use. Most people with prior experience of graphical editors and modellers have found the Quick Start document sufficient to enable them to use the product (simply dipping into this reference as required to answer specific technical questions).

In later releases standard installers will be available for Windows, Mac OS-X and Linux. For this pre-release a compressed archive is currently available for:

- Windows
- Linux (Ubuntu 8.10)

To install the software on any other platform you are likely to need to build a copy from the source files. This is not particularly complicated because the application has been written in a highly portable manner, but you will need a platform-dependant compiler and some understanding of how to compile and build applications. This process is not covered in this document, but is covered in the Developers Guide which is available on the MakeHuman Wiki at http://makehuman.wiki.sourceforge.net/DG_Introduction.

1.1 The New GUI

This version of the software is a Pre-Alpha technical preview. The GUI incorporates modelling controls based upon Ethnicity, Gender, Age, Muscle Tone and Body Mass. The full release will provide features to enable you to model individual anatomical features using controls that are more standardised than they were in the 2007 release candidate. This pre-release does not allow you to pose the figure. The full release will support the posing of figures. This version does not include props such as clothing or hair and does not yet contain animation features.

The current Pre-Alpha release supports the export of modelled and posed figures for input into a very wide variety of other modelling and rendering applications. The export capabilities will be further expanded with the full release. This GUI incorporates a small number of standardised 'intelligent' tools designed to minimise the learning curve for new users while providing powerful features to enable all users to rapidly model a character that meets their needs. Knowledge has been built into the tools so that, for example, if a female figure is being modelled (as defined by the Gender tool), then any body mass added to the model will accumulate fat in those areas of the body where a woman typically accumulates fat.

1.2 The Mesh

All MakeHuman humanoid figures are based on a single, highly optimized, light and professional mesh. Modelling of the mesh is performed by deforming the mesh rather than altering its topology. The mesh has been through a series of iterations to improve the structure so that deformations can be realistically applied while maintaining a low polygon count to minimise processing overheads. The mesh supports subdivision to enable higher density, smoothed meshes to be exported for high quality rendering. A considerable number of mesh deformation targets have been created by artists to provide you with a large number of realistic starting points from which to model particular ethnic, gender, age and body mass figures of your own design.

1.3 Python Scripting

The MakeHuman application has been structured to expose a great many of the program internals through the Application Programming Interface (API). This open structure has been documented and published to encourage the development of new scripts and plugin functionality that will help the application to develop and rapidly adapt to the needs of the user community.

1.4 External Rendering

MakeHuman incorporates a range of plugins to export a modelled and posed figure in 3D graphics formats supported by a very wide range of external modellers and rendering engines. These export functions have been written as Python plugins that can easily be extended and can serve as examples to enable other interfaces to be readily constructed. Support for the Renderman format is built into the MakeHuman application. Aqsis is the officially supported Renderman format renderer, but export to other formats and other renderers is also provided.

2. Installation of MakeHuman

In the full release of MakeHuman installers will be available for Windows, Mac OS X and Linux. Installers will be not available until the Alpha testing begins. Until then, compressed archives will be available for Windows and Linux which will need to be unzipped.

You will need about 30 MB of disk space for the zipped archive and a further 50MB of disk space for the application code once it has been unzipped. When you have successfully installed the application you can delete the zip file to recover the disk space that it occupies. It is recommended that you run MakeHuman on a machine with at least 512MB of memory and at least a 800MHz processor.

If you don't have Windows or Linux, you may still be able to build, run and use this application. The MakeHuman source code is available to enable you to perform a build on your own. This requires a bit more effort, additional knowledge and compilation tools. Considerable information on building a copy of MakeHuman can be found in the Developers Guide on the MakeHuman Wiki at http://makehuman.wiki.sourceforge.net/DG_Introduction.

2.1 Windows

The Windows version needs the Microsoft Visual C++ 2008 Redistributable Package, because MH is compiled with python 2.6. Without these libs, the app can crash at startup.
Download the Windows archive file and unzip it into a separate directory on your file system. Double click on makehuman.exe to start the application.

2.2 Mac OS X

The Pre-Apha code is not currently available as a compressed archive for Mac OS X due to lack of maintainers.

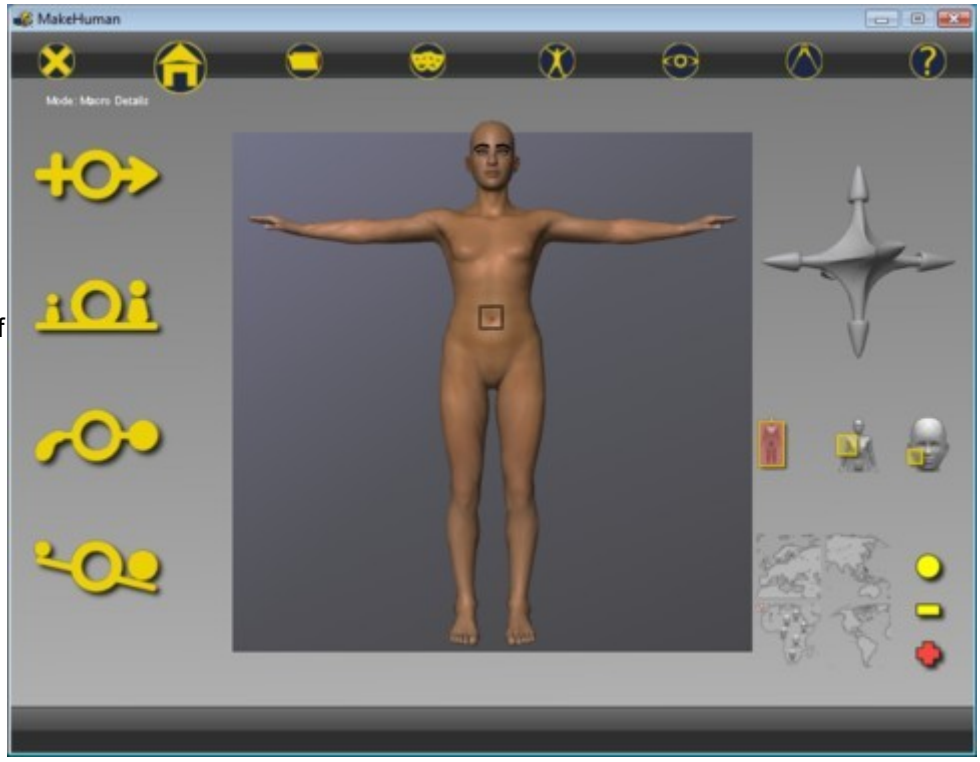
2.3 Linux

A zipped package is available for Ubuntu 8.10.
Download the archive and unzip it into a separate directory on your file system. Double click on the makehuman binary to start the application.

3. Graphical User Interface Controls

3.1 Default Display

When you first launch the MakeHuman application you should see the default human figure facing you with outstretched arms. As illustrated in the screenshot on the right, the figure should appear in the centre of the window and a set of toolbar buttons should be displayed along the top of the window to enable you to change modes. This is the modelling mode screen and enables you to control the anatomy of the figure (see Anatomical Controls). You can return to this screen at any time by pressing the Home button on the toolbar (second from the left).



The big pointy object on the right hand side of the screen is not active in this pre-release, but will be able to be used in future releases to scale and translate body parts (see Transformer Tool below). Various other controls are displayed on the left hand side of the screen and there is a bottom tool bar where buttons specific to the current mode can be displayed (empty in this image).

The small square box symbol shown in the middle of the figure's stomach in the image on the right is the cursor and should follow your mouse movements.

3.2 Controlling the Display

The human figure is displayed in 3D. You will need to get familiar with how to navigate around in the 3D environment, turning the model, zooming the camera in and out and panning around the window because you'll be doing this a lot as you design and pose your human figure. These basic navigation functions can be performed using either hotkeys on your keyboard or by using your mouse (or graphics tablet). A 3 buttoned mouse is recommended.

The numeric keys, along with the '+', the '-' and the '.' keys and the up, down, left and right keys on your keyboard are used as hotkeys. The numeric key

settings make most sense if you have a numeric pad on your keyboard. If you don't then you may still find it useful to use the ordinary numeric keys to perform operations that are not available using the mouse. For example resetting the display to fixed views.

Certain other hotkeys are available such as 'Ctrl Z' and 'Ctrl Y' for undo/redo.

Rotating the Figure:

- Left click the mouse and drag up, down, left, right or press 8, 2, 4 and 6 on the keyboard.

Zooming In and Out:

- If you have a central wheel on your mouse this can usually be configured to zoom in and out, otherwise you'll need to use the '+' and '-' keys on your keyboard. Pressing the '.' character resets the zoom (and removes any panning to point the camera at the centre of the figure).

Panning:

- Right click and drag or use the up, down, left and right keys to drag the contents of the viewing window around, effectively panning the camera. Pressing the '.' character returns the figure to the centre of the window (and resets any zoom).

Preset Positions:

- Pressing '1' returns to the default front facing view, '3' displays a side view and '7' an arial or top view. These keys do not reset the zoom, or panning, so press '.' if you need re-centre the camera.

Undo/Redo:

- Pressing 'Ctrl Z' can be used to undo the previous operation (change to ethnicity, gender, age, etc.)
- Pressing 'Ctrl Y' redoes an operation undone by using 'Ctrl Z'.

3.3 Main Toolbar (Changing Modes)

The main toolbar along the top of the MakeHuman window allows you to change mode. Each mode supports a particular category of function and provides controls and options specific to that function. The main functions and the controls and options that they provide are detailed in their own sections of this document. The following paragraphs on this page provide a very brief summary.

- The Exit button terminates the MakeHuman application.
- The Home button returns you to Modelling Mode (see Anatomical Controls).
- The File Mode button changes to File Mode to enable file interactions (see File Options).
- Expression Mode is not supported in this Pre-Alpha technical preview.
- Pose Mode is not supported in this Pre-Alpha technical preview.
- The Export Mode button changes to Export mode to access external rendering applications (see Export Options).
- Dimensions Mode is not supported in this Pre-Alpha technical preview.
- Application Help Screens are not provided with this Pre-Alpha technical preview.

3.4 Mode-Specific Toolbars

When you change modes a toolbar of buttons to perform mode-specific functions is displayed along the bottom of the MakeHuman window.

Modelling Mode Tetrawidget (see Anatomical Controls)

File Mode Toolbar (see File Options)

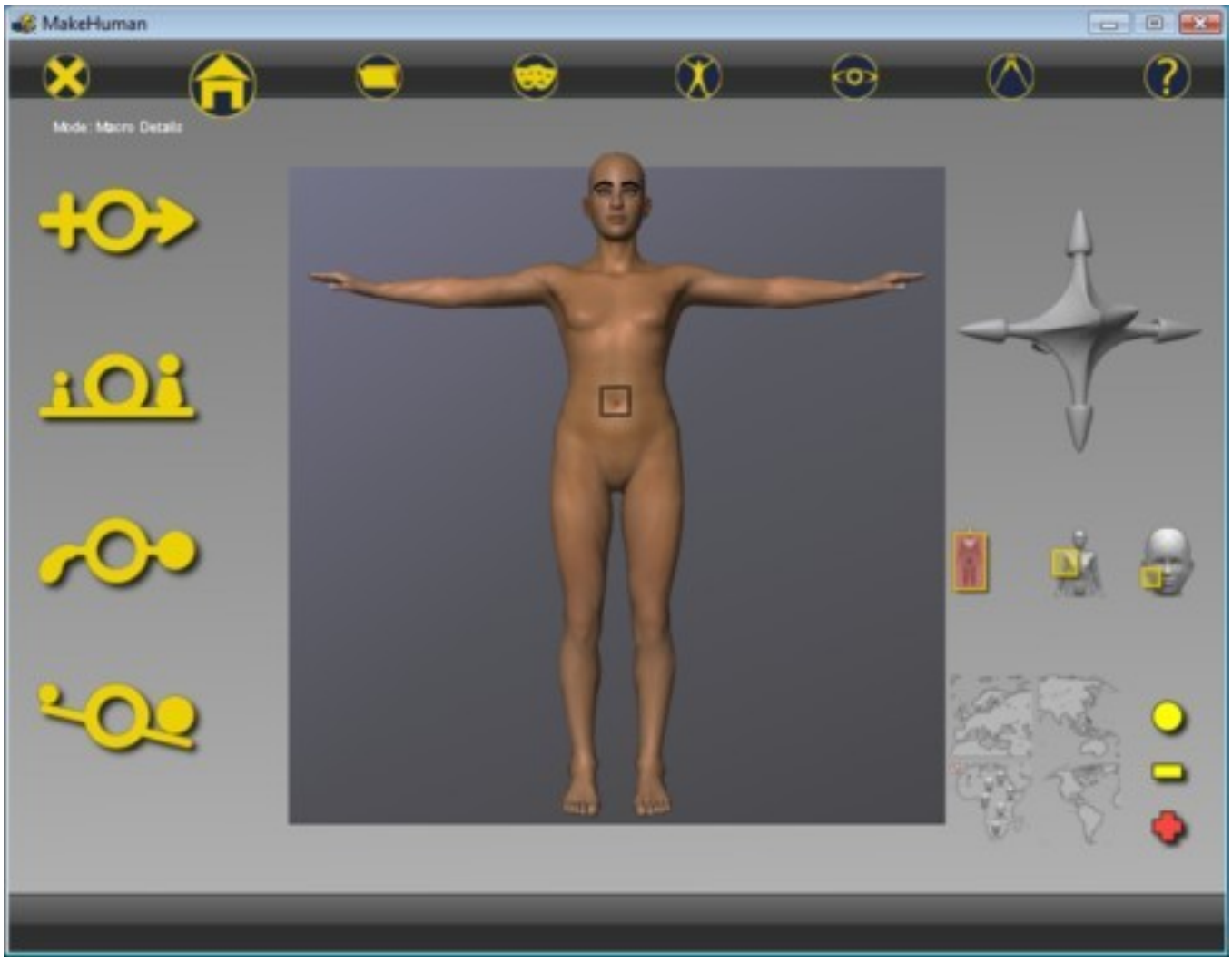
Expression Mode Toolbar (Not Supported in this prerelease)
Pose Mode Toolbar (Not Supported in this prerelease)

Export Mode Toolbar (see Export Options)

Dimensions Mode Toolbar (Not Supported in this prerelease)
Application Help Screens (Not provided with this prerelease)

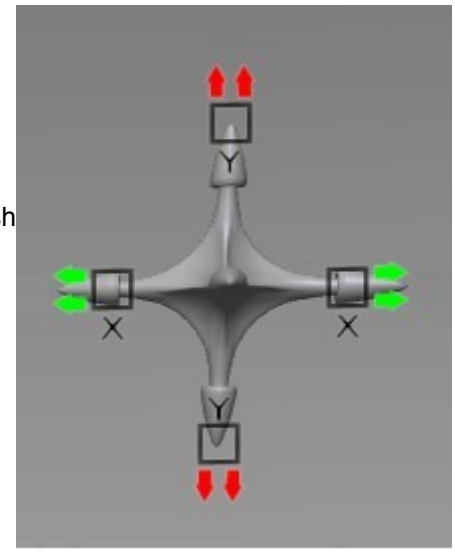
4. Anatomical Controls

MakeHuman will provide course-grain and fine-grain controls for defining the anatomy of your figure. This pre-release Technical Preview only provides course-grained controls.



A significant number of ethnic characteristics have been modelled by hand with corresponding gender and age variants. When you use the MakeHuman application to model a specific figure, the application uses a combination of these hand-modelled variants to create a specific model that meets your specifications, allowing you to easily generate a unique character. You can also apply generic variants for muscle tone and for portliness.

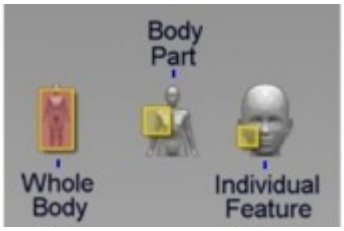
4.1 Transformer Tool



This tool is displayed in the top left corner of the main body of the screen, but is not active in this Pre-Alpha release.

The transformer tool is a all-in-one tool to quickly scale and translate the mesh parts. It's orientation follows the current orientation of the mesh, so it's very easy and intuitive to identify the axis to act on. Clicking on the point of arrow will enable translations mode, while clicking on the inner part, will enable the scale mode, on the specified axis

4.2 Granularity Selector



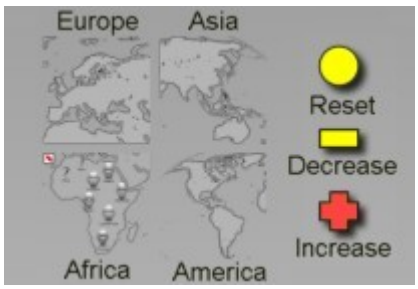
MakeHuman will provide course-grain and fine-grain controls for defining the anatomy of your figure. This Pre-Alpha release does not provide this level of control, so this selector has no effect. The following bullets describes how the granularity selector is currently expected to work.

- You select the extent to which changes are applied using the 3 buttons located on the right hand side of the Modelling page between the Transformer tool and the Ethnicity map.

- By default, changes are applied to the whole body, but you can elect to apply them to discrete body parts, such as the head, the upper arm, the forearm, etc. or to individual anatomical features, such as the nose tip, the left nostril, the right cheek, the lower mouth etc. Note that applying characteristics to the whole body takes a fair bit longer than to individual body parts and features, so you're likely to see the timer a bit more frequently.

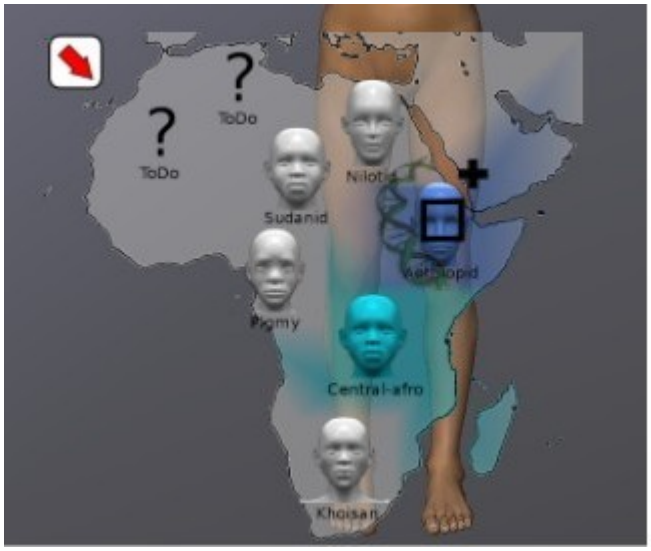
- A list of the selectable body parts is given below. A list of the individual anatomical features that can be selected is also given below. The individual features are also illustrated online at http://makehuman.sourceforge.net/FaceGroups/HeadFaceGroups.html as they correspond to the face groups used by the application.

4.3 Ethnic Mixer



The world map shown on the right hand side of Modelling Mode page enables you to select and mix together different ethnic characteristics. This Pre-Alpha technical preview only contains a small collection of the ethnic groups that will be available in the full release. The groups available in this pre-release are all African.

When you click one of the continents on the small graphic, a larger map of that region is displayed (remember that only the African map is activated in this Pre-Alpha technical preview). The larger map contains icons illustrating the facial characteristics of each of the different ethnic groups available from that region . The icons are arranged so that they correspond to the characteristics traditionally associated with that region.



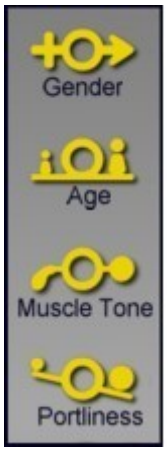
The buttons to the right of the small graphic allow you to select whether you wish to increase or decrease the proportion of characteristics inherited from the ethnic group that you click. The reset button allows you to remove the influence from that ethnic group on the current figure. To adjust the proportion of the features inherited from each ethnic group first click the 'increase', 'decrease' or 'reset' button, then click the facial icons on the larger map as many times as you need to apply the amount of change you wish to apply.

To indicate that you have finished editing this characteristic press the button at the top left corner of the enlarged graphic. Once you have set the ethnicity in this way the areas that you have used to build the 'ancestry' of your character will be shown on the map using darker colors to indicate more influence. The message line near the top of the screen also gives you a numeric breakdown of the ethnic proportions of your figure.

4.4 Tetrwidget Toolbar

The controls on the left hand side of the Modelling page allow you to adjust the gender, age and the physique of the figure. Pressing the left or right side of each button activates a '-' or '+' icon. Clicking the mouse on the figure then reduces or adds to that setting. Note that you have to click on the figure rather than the space around the figure.

To deselect the tools from the toolbar so that you can revert to using the mouse to rotate/pan/zoom you'll need to open and closer the ethnicity map in this Pre-Alpha release.



Gender - The Gender button allows you to make the figure more or less masculine, adjusting the general form of the entire body, generally making body parts more angular for the masculine figure and smoother for the female figure.

Age - The Age button controls aspects of the anatomy that characterise age with older characters tending to attract looser skin and muscle forms with more wrinkles.

Muscle Tone - The Muscle Tone button controls how full and muscular the figure is, with the higher settings transforming the character into a more muscular individual.

Portliness - The Portliness button controls how lean or overweight the figure is.

4.5 Body Part List

This Pre-Alpha release does not support the selection and manipulation of individual body parts.

In the final release it is anticipated that the following body parts will be able to be independently selected when using 'Body Parts' selection in Modelling Mode:

- Head

- Neck
- Torso - Abdomen, Pelvis, Left Shoulder, Right Shoulder
- Arms - Left Upper Arm, Right Upper Arm, Left Forearm, Right Forearm, Left Hand, Right Hand
- Legs - Left Thigh, Right Thigh, Left Calf, Right Calf, Left Foot, Right Foot

4.6 Individual Features List

This Pre-Alpha release does not support the selection and manipulation of individual anatomical features. In the final release it is anticipated that the following anatomical features will be able to be independently selected when using 'Individual Features' selection in Modelling Mode:

- Head - a large number of individual muscle groups on the face can be selected. There are 4 separate zones around the scalp that can be independently selected. The ears consist of 3 independently selectable sections.
- Eyes - Each eye can be selected.
- Mouth - Each of the teeth in the upper and lower jaw can be selected and the tongue can be selected.
- Neck - The upper and lower neck, and the Adam's apple can be selected.
- Torso - Various muscle groups and regions of the torso can be independently selected.
- Arms - Individual muscle groups on each arm can be selected.
- Hands -The palms and each segment of each finger and thumb can be individually selected on either hand, as can individual finger nails.
- Legs - Individual muscle groups on each leg can be selected.
- Feet -The heels, cores and each segment of each toe can be individually selected on either foot, as can individual toe nails.

5. Poses (Not Supported in this 1.0 Pre-Alpha technical preview)

A key part of the functionality of the MakeHuman application will be to support the posing of characters (comparable with the sort of posing of characters available in the MakeHuman 0.9.1 release candidate). This functionality is not available in this Pre-Alpha Technical Preview release.

The default position has the human figure standing facing straight towards the camera with arms outstretched. Currently this is the only pose supported. In the final release it is anticipated that there will be a library of predefined poses which can be loaded to cause the character that you have specified to adopt that pose. The Pose Mode screen will then allow you to select individual joints and adjust the settings approriate to that joint. Once you have defined a pose you can save it and subsequently apply the same pose to other figures.

5.1 The Pose View

To enter the 'Pose' view, you will be able to select the 'Pose' icon from the toolbar. This view will contain a 'map' of the body enabling you to select a joint to work on. Once a joint is selected a TetraWidget specific to that body part will appear, enabling the settings for that joint to be manipulated.

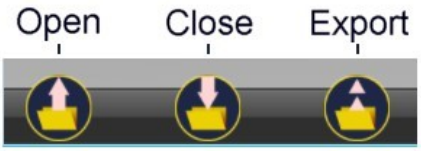
It is anticipated that it will be possible to pose the following body parts:

- **Head**
- **Neck**
- **Left and Right Collar**
- **Left and Right Upper Arm**
- **Left and Right Lower Arm**
- **Left and Right Hand**
- **Pivot**
- **Left and Right Upper Leg**
- **Left and Right Lower Leg**
- **Left and Right Foot**
- **Fingers and Thumbs**
- **Toes**
- **Eyes**
- **Mouth**

6. File Options

In this Pre-Alpha Technical Release the file options on the 'File' mode screen do nothing.

This Pre-Alpha does include an experimental 'obj' export capability that can be run from any mode by pressing the "e" key on your keyboard. This exports a wavefront object (and the corresponding material layer). It also exports a BVH file (a BioVision Hierarchy file) that can be imported on Blender to build a skeleton. All of these files are saved using hard-coded file names in the same folder as the MakeHuman executable is stored.



In the full MakeHuman application the File Mode screen and the options presented on it will be used to control the opening and saving of files.

7. Export Options

Three export options are provided by a toolbar at the bottom left of the' Export' mode screen.

7.1 Aqsis

Aqsis is an Open Source rendering application that implements the Renderman® standard. You can see examples of its capabilities and download the sorftware from http://wiki.aqsis.org/. MakeHuman provides an option to directly export an Aqsis compatible Renderman format file containing a posed and morphed humanoid model.

The files generated by using this export option are written into the 'renderman_output' directory on your file system.

7.2 Pixie

Pixie is an Open Source RenderMan renderer for generating photorealistic images released on a GNU Lesser General Public License (LGPL). You can see examples of its capabilities and download the sorftware from http://www.renderpixie.com/. MakeHuman provides an option to directly export a Pixie compatible Renderman format file containing a posed and morphed humanoid model.

The files generated by using this export option are written into the 'renderman_output' directory on your file system.

7.3 POV-Ray

The Persistence of Vision Raytracer (POV-Ray) is a popular and freely available rendering application capable of generating stunning images and animations. You can see some of the images that it is able to generate and download the software from http://www.povray.org.

MakeHuman provides an option to directly export a POV-Ray 'include' file containing a posed and morphed humanoid model. This option also exports a pigment map and generates a sample 'scene' file that contains a series of examples illustrating a range of ways of using the model. These examples and further options for rendering the exported object are explained in more detail in Appendix 1 of this document. Additional information can also be found on the Wiki page at http://makehuman.wiki.sourceforge.net/UG_POVRay_Export, but be aware that this page refers to more recent versions of the export functionality, so not everything discussed there is relevant to the release covered by this document.

The files generated by using this export option are written into the 'pov_output' directory on your file system.

8. Rendering Options

When complete you will be able to set up MakeHuman to automatically render using an external renderman compliant engine. This functionality is not available in this Pre-Alpha release. Alternatively you can Save or Export your figure (see File Options and the Export Options sections of this Guide) and import into a wide range of other rendering packages.

If configured to automatically render in an external renderman compliant engine you will be able to simply press the “rendering” (or “preview”) button and MakeHuman will save a standard rib file and invoke your chosen renderer (e.g. PRMan, Air, 3Delight, Pixie or Aqsis). The MakeHuman team only officially supports the Aqsis Open Source renderer which generates high quality renders but which is very slow, with a close-up render of a figure often taking more than 15 minutes. It is however anticipated that the performance will improve with Aqsis V2.0.

8.1 Installing Aqsis

Editorial Note: Need instructions for installing Aqsis and setting it up so that MakeHuman can use it to automatically render files.
For information about Aqsis please refer to the Aqsis Wiki at http://wiki.aqsis.org/doku.php?id=doc:index.

8.2 Installing Other Renderers for Automatic Rendering

Editorial Note: Need instructions for installing other renderers and setting them up so that MakeHuman can use them to automatically render files.

8.3 Other Renderers

To use other renderers you will need to export the relevant anatomy, pose and texture information from MakeHuman in one of the supported Save or Export formats. Many renderers directly support the '.obj' format. Otherwise you'll need to use a converter to convert the object definitions into a format supported by your renderer of choice. For example, the free PoseRay utility can be used to convert '.obj' files.

8.3.1 POV-Ray

MakeHuman includes export functionality that can generate a POV-Ray 'include' file which can be used to render a MakeHuman humanoid figure. POV-Ray is a free and popular raytracing engine available for Windows, Mac OSX and Linux that can be downloaded from their home page at http://www.povray.org/. A Wiki page describing the POV-Ray export functionality of MakeHuman and the rendering options available can be found at http://makehuman.wiki.sourceforge.net/UG_POVRay_Export.

9. Application Programming Interface

The core OpenGL 3D engine for MakeHuman is written in C, but the large majority of the functional behaviour is implemented using Python, which is an Open Source Object Oriented programming language.
Python is an extremely 'readable' language and lends itself to the development of enhancements and plugins to extend the capabilities of MakeHuman.

The MakeHuman installation applications for Windows, Mac OSX and Linux incorporate everything you need to run the MakeHuman application, create 3D human models and run Python plugins. To develop Python plugins or other code that interfaces with MakeHuman you will need to be familiar with the Python language and Python development techniques. You may also choose to install a full Python development environment, possibly including an Interactive Development Environment. The Python home page can be found at http://www.python.org/ where you can download the full free Python development environment and access Python community resources and documentation.

The MakeHuman application itself contains a Python interpreter. If you write a Python script called 'foo.py' you can invoke it with the command "makehuman foo.py".

The MakeHuman Python Application Programming Interface (API) is fully documented at http://makehuman.sourceforge.net/API/. The following table presents a brief description of the functionality that the main MakeHuman Python scripts expose as an API. The application data structures are also contained within the class definitions for the Python code. Information about the evolution and design of the MakeHuman mesh is contained on the Wiki at http://makehuman.wiki.sourceforge.net/The_Humanoid_Mesh. Each face on the MakeHuman mesh is assigned to a face group which is used for the selection of different anatomical features within the MakeHuman GUI. Images illustrating each of these face groups can be seen in the face group documentation at http://makehuman.sourceforge.net/FaceGroups/index.html.

Module/Application	Description
main.py	The Main Application handling the interaction with the user and calling functionality appropriate to the actions initiated by the user.
module3d.py	A Python Module in the 'mh_core' directory containing all of the base classes needed to manage the 3D data structures at runtime. This includes the data structures themselves as well as methods to handle the manipulation in memory of those data structures. For example: the Vert class defines the data structures to hold information about mesh vertices objects, the Face class defines data structures to hold information about mesh face objects. These base classes implement a nested heirarchical structure for the objects that go to make up the scene that is shown to the user. For example, a FaceGroup object contains groups of mesh face objects as defined by the Face class. An Object3D object contains all of the FaceGroup objects that go to make up a particular discrete object, such as the eyeball or the head. The Scene3D object contains all of the Object3D objects that go to make up the entire scene.

files3d.py	A Python Module in the 'mh_core' directory that handles 3D file formats supported by MakeHuman. This module will support integration with functions to handle most common 3D file formats. This includes functions to transpose imported 3D data into a standard internal format for each 3D file format supported by the import functionality and functions to generate 3D data structures that correspond to 3D file formats supported by the export functionality. It also includes generic transformation utilities such as the dataTo3Dobject() function which takes an object defined in the standard internal format and makes it visible to the user.
algos3d.py	A Python Module in the 'mh_core' directory containing all of the algorithms used to perform high-level 3D transformations on the mesh that is used to represent the human figure in the MakeHuman application. These include morphing for anatomical variations and pose deformations, collision deformations, etc. This module also contains a number of functions used during the development cycle to test and to visualize the mesh.
aljabr.py	A Python Module in the 'mh_core' directory containing a set of the most common 3D algebraic operations used in MakeHuman. These are mostly the vector and matrix operations core to any 3D application.
textures3d.py	A Python Module in the 'mh_core' directory containing a series of functions to perform standard processes on bitmaps. These functions provide simple higher level functionality to other functions, such as the 3D algorithms in algos3d.py.
widgets3d.py	A Python Module in the 'mh_core' directory containing a series of classes that implement GUI utility functions.
guitoolbar.py	A Python Module in the 'mh_plugins' directory that implements the application toolbars that provide access to the various MakeHuman operating modes and to the functionality available in each mode.
guicommon.py	A Python Module in the 'mh_plugins' directory that implements the 'guicommon' class structures and methods to support functions common to the different operational modes made available through the MakeHuman GUI toolbar.
guimodelling.py	A Python Module in the 'mh_plugins' directory that implements the modelling functionality available in Modelling Mode (the default mode and the Home option on the toolbar).
guirender.py	A Python Module in the 'mh_plugins' directory that implements the rendering functionality available in Render Mode. This functionality connects through to the various external rendering applications supported by MakeHuman.
guifiles.py	A Python Module in the 'mh_plugins' directory that implements the file handling functionality available in Files Mode. This functionality supports the loading of texture and morph target files.
guifileselector.py	A Python Module in the 'mh_plugins' directory that implements the file selector GUI object used in Files Mode.
halfedges.py	A Python Module in the 'mh_plugins' directory that implements the data structures and functions to support vertex/face analysis using HalfEdges. This technique introduces two HalfEdges for each edge, each with opposing orientations aligned along the length of the edge. The module provides high-level functions to developers to support validation of data structures and interrogation of the mesh such as support for tracking around the edges of a single face or a hole in a mesh object and the identification of all polygonal faces sharing a particular vertex. These functions are particularly useful with the type of surface mesh used by MakeHuman where no more than two polygons should share an edge.

metadataengine.py	A Python Module in the 'mh_plugins' directory that implements a metadata search engine to rapidly locate records within library files.
subdivision.py	A Python Module in the 'mh_plugins' directory that handles the subdivision of surfaces. Subdivision can be performed before exporting or rendering the model to smooth the mesh and improve the appearance of the finished image.
blendersaveobj.py	A Python Utility in the 'utils' directory that saves a human figure in a blender compatible object file.
blendersavevertscolor.py	A Python Utility in the 'utils' directory that saves MakeHuman vertex colour information in a blender compatible format.
mh2renderman.py	A Python Module in the 'mh_plugins' directory that implements a function to export a human model in Renderman format and to invoke the Aqsis or Pixie rendering engine to render the Renderman format files.
mh2obj.py	A Python Module in the 'mh_plugins' directory that implements a function to export a human model in Wavefront obj format.
mh2povray.py	A Python Module in the 'mh_plugins' directory that implements a function to export a human model in POV-Ray format. POV-Ray is a Raytracing application (a renderer) that is free to download and use.
mh2bvh.py	A Python Module in the 'mh_plugins' directory that implements a function to export a Biovision motion capture file in BVH format.

The MakeHuman Developers Guide at http://makehuman.wiki.sourceforge.net/DG_Introduction contains a description of how the application works and an API section that contains links to details of both the Python and the C application components.

10. Plugins

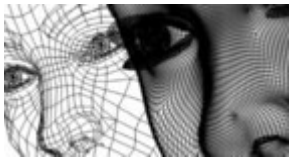
Each of the different application modes, along with the export and rendering functions to integrate with different renderers are coded as plugins and can serve as an example to plugin developers to illustrate how extra plugins can be implemented. Further information about how to develop plugins to work with MakeHuman can be found on the Plugin Development section of the MakeHuman Developers Guide at http://makehuman.wiki.sourceforge.net/DG_Plugin_Development.

It is anticipated that future development will include conversion utilities to integrate with an extended range of modeller and rendering applications, feature enhancements, such as animation features, clothing, hair and prop integration and tools/utilities.

There are plugins under development for extended characters based upon alternative meshes, for example a 'toon' mesh for cartoon-like characters with exagerated features. It is also possible to envisage plugins to support the modelling of other types of creature.

The Wiki version of this page at http://makehuman.wiki.sourceforge.net/UG_Plugins will be used as a reference page to accumulate information about plugins as they are developed.

11. The Mesh



The principal aim of the MakeHuman project is to develop an Open Source application capable of realistically modelling a very wide variety of human anatomical forms in the full range of natural human poses from a single, universal mesh.

Central to this is the design of a 3D humanoid mesh that can readily be parametrically manipulated and deformed to represent alternative anatomical characteristics while retaining and respecting a common structural skeleton that permits poses and the corresponding deformations to also be parametrically manipulated. This objective has been pursued to afford the artist the maximum degree of experimental freedom when using the software. It frees the artist from the artificial constraints that are inherent to a model that has pre-established gender or age.

By pursuing this aim the MakeHuman Team have developed a model that can combine different anatomical parameters to transition smoothly from the infant to the elderly, from man to woman and from fat to slim. The vast wealth of potential combinations provides the artist with an extraordinarily broad range of possibilities for artistic expression but presents many interesting problems to the development team. In particular it adds to the classical problems of 3D modelling (number of polygons, square or triangular faces, etc.) the problems of constructing a super mesh that can be transformed into any form of human while being sufficiently optimised to be able to be manipulated on desktop machines, yet still producing a professional quality of output. By contrast, most modellers only have to produce a single model that can be created and adapted for a single project.

The current MakeHuman mesh has evolved through successive iterations of the MakeHuman project, incorporating lessons learned, community feedback and the results of considerable amounts of study and experimentation. No generic mesh is perfect and this mesh has inevitably been subject to some compromise and will undoubtedly continue to be refined in future releases. Nevertheless, the current mesh represents a remarkable achievement and is a great source of pride for the MakeHuman team. The current iteration, known as the 'Zmesh' comprises a state of the art universal humanoid model. A paper describing the characteristics and capabilities of the mesh along with a brief history and discussions about potential future enhancements can be found on the MakeHuman Wiki at http://makehuman.wiki.sourceforge.net/The_Humanoid_Mesh.

12. Licensing

12.1 Overview

Licensing is a complex matter, particularly for software and data intended to be free and widely reusable, such as is the case with MakeHuman©. Most Copyright law is oriented heavily towards protecting commercial interests, so using this law to give you rights while protecting you from rights acquired by others is a delicate balancing act. The intention of the MakeHuman team is to license the reuse of its Copyright materials, free or charge, both commercially and non-commercially. Standard time-tested licenses have been used to attempt to bring clarity to this complex legal issue.

The MakeHuman Mesh licensing paragraph below has been particularly delicately worded so as to try to afford you extensive and wide ranging authority over creative works that you produce while attempting to avoid that the use of MakeHuman by others places unnecessary resrtictions upon you. MakeHuman uses a standardised mesh topology which is deformed using standard deformation algorithms (morph targets). Files exported by different artists therefore often bear a great many similarities. It is the intention of the MakeHuman team that you should be able to copyright your own, sufficiently unique or distinct artistic works, without overly constraining other MakeHuman users who will be using very similar meshes.

12.2 The MakeHuman Application

MakeHuman© is an Open Source application that you can use and redistribute in original or modified form under the terms of the GNU General Public License as published by the Free Software Foundation.
Copyright© 2001-2009 is retained by the MakeHuman Team (makehuman.org) who grant you permissions to use released code under the conditions and terms of the GNU General Public License 3.0 or (at your option) any later version of the GNU GPL.
This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

12.3 MakeHuman Source Code Files

Source code for both released versions of the MakeHuman application and for versions currently under development is also governed by the terms of the GNU General Public License as published by the Free Software Foundation. These source files are made available in the hope that they will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

12.4 Other MakeHuman Application Code

From time to time the MakeHuman Team may release other application code in various forms. Unless otherwise explicitly stated this application code is also covered by the GNU GPL. For example, the MakeTarget scripts for use with the Blender Modelling application are distributed under the GNU GPL and the MakeHuman morph target files

that they generate are covered by this license. Meshes generated or converted using any such tools are covered by the Mesh licensing statements below. Such application code is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

12.5 The MakeHuman Mesh

The MakeHuman© application uses a 3D mesh to represent human figures. This mesh was developed in a number of successive iterations (see The Humanoid Mesh) over a number of years to support a diverse range of physiological variants and pose-based deformations. The application uses a mesh with a standard topology (how the vertices, edges and faces relate to one another), which is manipulated by applying standard morph targets (mesh deformations).

The specific data files used to define and deform the mesh, distributed with the MakeHuman application are covered by the terms of the GNU General Public License 3.0 and you are granted permission to use and redistribute those files in original or modified form under those terms. These data files can also be manipulated using the Blender MakeTarget script, but they, and derivative works remain under the terms of the GNU GPL.

When you export, save, or in any way convert a MakeHuman data file, you may store any deformations that you have applied/created, along with the topology of the base mesh and any positional information that you have not materially changed. You are the Copyright holder of changes that you have made. The MakeHuman Team retains Copyright over the topology and the base mesh, but grants you permission to copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the generated materials over which they hold Copyright, including the topology and positional information constituting the base mesh provided that this does not prevent others from doing the same.

You do not acquire exclusive Copyright over the topology or positional information constituting the base mesh, so others can, in perpetuity also create works derived from the topology and positional information constituting the base mesh and the morph target data used to deform the base MakeHuman mesh. Note that you may be able to establish copyright over works that you produce using the MakeHuman application if your work is sufficiently unique or distinct. The definition of how unique and distinct a work needs to be to be copyrightable is jurisdiction dependant and can vary over time as legal systems evolve and adapt to new legal challenges.

You own any 2D images that you create using exported MakeHuman mesh data and any copyright that applicable law affords you over those materials. Note that the definitioin of how unique and distinct a work needs to be to be copyrightable is jurisdiction dependant and can vary over time as legal systems evolve and adapt to new legal challenges.

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

12.6 Documentation

In general the documentation (including the text on this Wiki) produced by the MakeHuman© Team is Copyright© 2001-2009 by the MakeHuman Team (makehuman.org) who grant you permission to use and redistribute in original or modified form. Note that certain names and terms are individually copyrighted or are trademarks, so separate restrictions on the use of those names or terms may exist. For example, this authorisation does not entitle you to produce documents claiming to be the MakeHuman Team or claiming to produce an application called MakeHuman.

The Documentation Archive on this Wiki may also contain documents or links to documents that were or were not produced by the MakeHuman Team and/or may contain their own separate Copyright© notices, which may differ from this generic documentation licensing statement. Any licensing statements within individual documents take precedence over this licensing statement.

Appendix 1 - POV-Ray Export

Note (BUG Warning). The POV-Ray Export in the 1.0 Pre-Alpha released from 1st March generates an error with a missing file called 'texture.tga'. To fix this there are two references, one in each of the generated 'pov' and 'inc' files to 'tga "texture.tga"'. These need changing to 'tiff "texture.tif"' and the file "texture.tif" needs to be copied from "data/textures/" to "pov_output/".

The Persistence of Vision Raytracer is a popular and freely available rendering application capable of generating stunning images and animations. You can see some of the images that it is able to generate and you can download the software from http://www.povray.org . The application is well supported on the newsgroups at http://news.povray.org . The application renders images from text files that describe the scene using a markup language called the Scene Description Language (SDL). It does not incorporate a modeller, but SDL components exported from modellers can be incorporated into POV-Ray scenes.

MakeHuman provides an option to directly export a POV-Ray 'include' file containing a posed and morphed humanoid model. This option also exports a pigment map and generates a sample 'scene' file that contains a series of examples illustrating how to use the model. You can incorporate MakeHuman figures into your own POV-Ray scene files using a few short pieces of SDL that you type or cut and paste into those text files.

The Generated Include File

The POV-Ray 'include' file generated by the export option is designed to enable you to easily add a humanoid object to your POV_Ray scenes. The file contains arrays of coordinates and normals plus a set of macros to generate objects from those arrays, along with other useful settings and utilities.

Mesh2 macro & texture

The main object is generated using the *MakeHuman_Mesh* macro which returns a complete POV-Ray mesh2 object that you use in your scene. A texture called *MakeHuman_Texture* is also defined which uses a uv-mapped image map to apply a reasonable texture to the mesh2 object. These may be the only parts of the generated file that you need to use and they enable you to add your generated humanoid figure to your scene using the following two lines of SDL:

```
#include "makehuman.inc"
object {MakeHuman_Mesh(0) texture {MakeHuman_Texture}}
```

You can of course define your own textures to enhance the image and examples within the sample scene file illustrate more sophisticated techniques, such as sub-surface scattering (SSS).

The macro supports a single parameter (*MakeHuman_Displacement*). You'll usually use a mesh2 object generated using a parameter of 0, but you can also generate extra mesh2 objects that are very slightly smaller or very slightly larger by specify small displacements (usually about 0.003 units). Subsurfaces generated in this way can be used to support a form of sub-surface scattering (see below).

Data Arrays

The include file contains various macros and variables that you may find useful. The *MakeHuman_MakeMesh* macro uses the following arrays that are set within this file.

- MakeHuman_VertexArray - the 3D vectors used for the mesh2 vertex_vectors attribute.
- MakeHuman_NormalArray - the 3D vectors used for the mesh2 normal_vectors attribute.
- MakeHuman_UVArray - the 2D vectors used for the mesh2 uv_vectors attribute.
- MakeHuman_FaceArray - the set of indices triplets used for the mesh2 face_indices attribute. These are listed in sets of 3 with one value for each of the 3 vertices of each triangular face and look a bit like a vector except that the values are all integers.
- MakeHuman_UVIndexArray - the set of indices triplets used for the mesh2 uv_indices attribute. These are listed in sets of 3 with one value for each of the 3 vertices of each triangular face and look a bit like a vector except that the values are all integers.

Sphere & Cylinder macros

The MakeHuman_Spheres macro generates a union of spheres where the spheres are positioned at the vertices of the 3D humanoid mesh. The MakeHuman_Cylinders macro generates a union of cylinders to represent the edges of the faces of the 3D humanoid mesh. These both align with the mesh2 object and can be used in isolation or in more or less any combination to highlight the vertices or edges. They can also be used as an object pigment to simply 'draw' the triangular mesh structure on the surface of the mesh2 object.

```
#include "makehuman.inc"
object {MakeHuman_Sphere(0) {rgb <1,0,0>}}
object {MakeHuman_Cylinder(0) pigment {rgb <1,1,0>}}
```

These macros also support a single *MakeHuman_Displacement* parameter (e.g. 0).

Camera and light_source

A camera definition called MakeHuman_Camera and a light source definition called MakeHuman_LightSource are provided so that you can readily reproduce camera and lighting settings similar to the settings in MakeHuman at the time the 'include' file is generated. These are used as the default settings in the sample scene file that is generated so that the image you get by just rendering the sample scene file can readily be compared with the object as it appears in MakeHuman. The texture will differ and the raytraced image will include shadows that will be missing in the MakeHuman image.

Variables

A set of variables is defined at the top of the include file that you may find useful when using the generated object in your scene. These variables include camera settings as defined in MakeHuman at the moment the file was generated:

- **MakeHuman_CameraX** - The x coordinate of the camera location before it is rotated into position.
- **MakeHuman_CameraY** - The y coordinate of the camera location before it is rotated into position.
- **MakeHuman_CameraZ** - The z coordinate of the camera location before it is rotated into position.
- **MakeHuman_CameraXRot** - The camera rotation around to x-axis (applied after it is translated into

position).
- **MakeHuman_CameraYRot** - The camera rotation around to y-axis (applied after it is translated into position and rotated around the x-axis).
- **MakeHuman_CameraFOV** - The vertical Field of View as an angle in degrees.
- **MakeHuman_ImageHeight** - The height in pixels of the OpenGL viewport in MakeHuman (used to define the aspect ratio).
- **MakeHuman_ImageWidth** - The width in pixels of the OpenGL viewport in MakeHuman (used to define the aspect ratio).

Variables are also provided which describe the position and size of the humanoid object.

- **MakeHuman_MaxExtent** - A 3D vector containing the maximum x, y and z extents of the humanoid object.
- **MakeHuman_MinExtent** - A 3D vector containing the minimum x, y and z extents of the humanoid object.
- **MakeHuman_Center** - A 3D vector containing the x, y and z coordinates of the centre of the humanoid object.
- **MakeHuman_Width** - A decimal value containing the width of the humanoid object in POV-Ray units (the x-dimension).
- **MakeHuman_Height** - A decimal value containing the height of the humanoid object in POV-Ray units (the y-dimension).
- **MakeHuman_Depth** - A decimal value containing the depth of the humanoid object in POV-Ray units (the z-dimension).

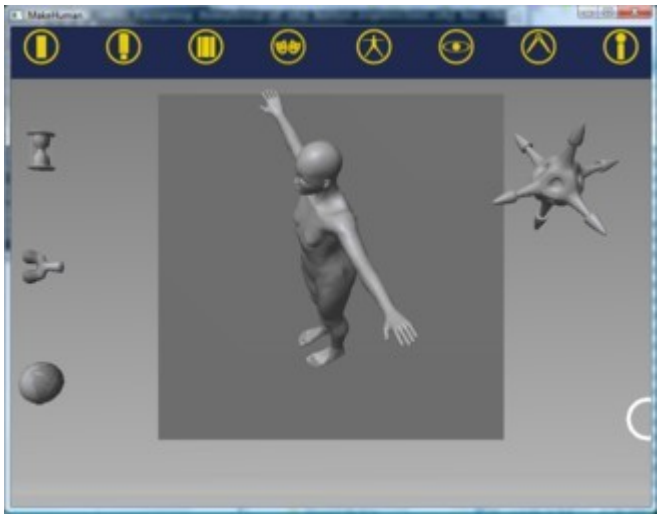
The generated object is centred at the origin. It can be scaled, translated and rotated as required. To move the object up so that its base is level with the XZ-plane you can translate by -y* MakeHuman_MinExtent.y. For example:

```
#include "makehuman.inc"
object {MakeHuman_Mesh(0)
   texture {MakeHuman_Texture}
   translate -y* MakeHuman_MinExtent.y
}
```

The Generated Scene File

The scene file generated by the export option is purely there to serve as an example. In fact it contains more than one example, illustrating a number of different ways that the 'include' file can be used. At the top of the file there is a variable declaration for the variable 'Example'. You can edit this to render whichever example you want:

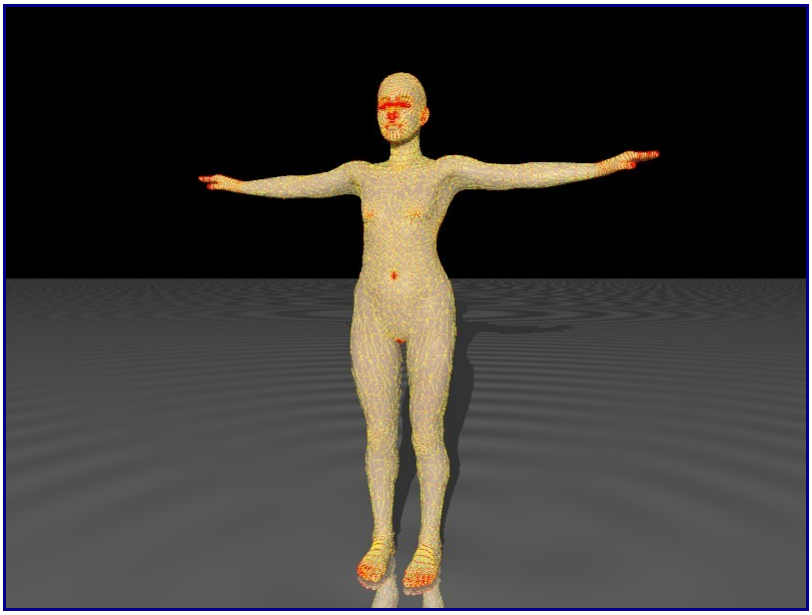
Example = 0
Specifying "#declare Example=0;" at the top of the sample scene file renders a POV-Ray mesh2 object using the camera settings as they were in the MakeHuman application when this file was generated.



The image on the left shows the POV-Ray render. The image on the right shows the MakeHuman screen.

Example = 1

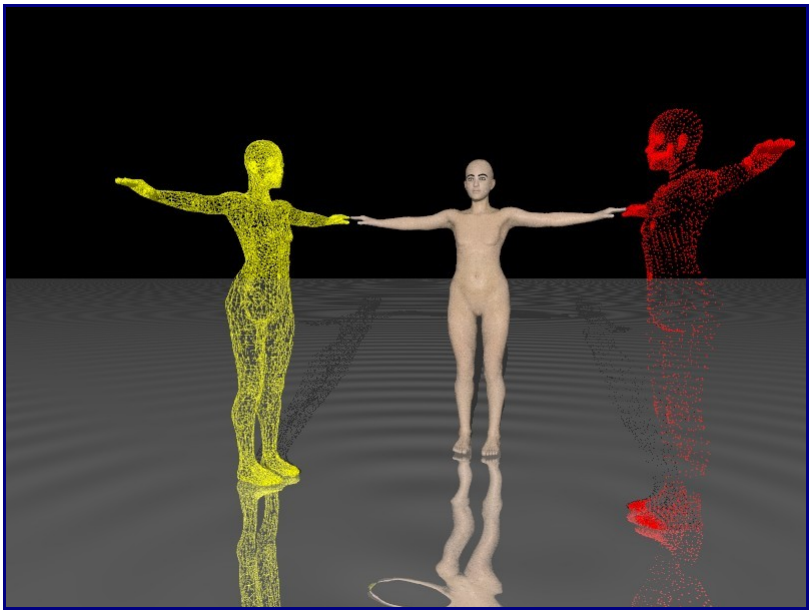
Specifying "#declare Example=1;" at the top of the sample scene file renders a POV-Ray mesh2 object and overlays an object built using spheres at the vertices and an object built using cylinders along the edges of the faces of the underlying mesh.



The sphere and cylinder objects can be used on their own or to add a mesh texture to the mesh2 object. The SDL used in the generated include file also serves to illustrate how easily the vertex array can be used for generating objects that follow the surface of the mesh.

Example = 2

Specifying "#declare Example=2;" at the top of the sample scene file renders a POV-Ray mesh2 object, a sphere-based object and a cylinder-based object, translating them so that they are each separately visible.



The sphere and cylinder objects can be used on their own or to add a mesh texture to the mesh2 object. The SDL used in the generated include file also serves to illustrate how easily the vertex array can be used for generating objects that follow the surface of the mesh.

Other Examples

Other examples are available in the sample scene file, for example, specifying "#declare Example=3;" at the top of the sample scene file renders a scene that illustrates the use of the MakeHuman_Mesh macro with a non-zero surface displacement parameter. In this example, three mesh2 objects are generated of a very slightly different size that are overlayed to provide sub-surfaces. The outer surfaces are made translucent and experimental textures are applied to scatter light between those surfaces.

This solution for simulating Sub-Surface Scattering (SSS) is not perfect and can be difficult to light in a manner that generates the desired effect, but it is provided to encourage folk to experiment and to allow people to play with some more advanced textures.

This technique is described further on the Wiki at http://makehuman.wiki.sourceforge.net/UG_POVRay_Export.

Texture Maps

An image map is provided which corresponds to uv-coordinates that are mapped to the mesh2 humanoid object. This image map is used to define *MakeHuman_Texture*, but that texture definition also incorporates a subtle normal perturbation and a finish statement. You can define your own texture directly if you wish and have the option to incorporate the image map in that texture definition. The following example incorporates the image map without adding a normal or finish block:

```
#include "makehuman.inc"
object {MakeHuman_MeshObject
  texture {
    pigment {uv_mapping
      image_map {tiff "texture.tif"}
    }
  }
}
```

In this example, the image map is contained within the targa format file called *texture.tga*. It is uv-mapped to the mesh2 using uv-coordinates built into the object when MakeHuman_Mesh macro is called.

Image maps can also be used to control the surface normals and the finish applied to the object using any of the file formats supported by POV-Ray (look for 'image_map' in the POV-Ray help for more details).

Animation

At present the MakeHuman application does not provide animation features, so the only available option is to create individual frames and export each frame separately. This can of course be tedious for long animations, but may still be viable for creating animation loops of a few seconds per cycle.

POV-Ray supports animation in that it can generate sequences of still images based upon user defined changes from one frame to the next. It does not put the images together into a video format. You will need a separate application for that and the application you choose will depend upon the video or animation format that you wish to generate. This is discussed further on the Wiki at http://makehuman.wiki.sourceforge.net/UG_POVRay_Export.