



## SOUL FORGE ART DESIGN SPECIFICATION

VERSION 1  
REVISION (10)  
PROPRIETARY

Creation		Last Update	
Document	Soul Forge ADS	Saved	10/29/2003
Authors	Jeff Hanna, Greg Grimsby, Brian Traficante, Chris Woodum		
Date	09-May-2003 3:33 PM		

Reviewer Name	Position	Date Reviewed
Frank Coker	Lead Designer	7/1/2003
Steve Marvin	Designer	7/2/2003

**REVISIONS**

Version	Date	Author	Comments
Initial Revision	09-May-2003 3:33 PM	Jeff Hanna	Document created
Revision 1	01-June-2003 2:30 PM	Jeff Hanna	Added Section 1
Revision 2	10-June-2003 1:30 PM	Jeff Hanna	Reworked styles for better readability
Revision 3	17-June-2003 1:58 PM	Jeff Hanna, Greg Grimsby	Art Style section (3) added.
Revision 4	10/29/2003	Jeff Hanna, Greg Grimsby, Chris Woodum	Sections 4 & 6 (characters and armor) added.
Revision 5	10/29/2003	Jeff Hanna, Greg Grimsby	Section 2 (Technical Art Creation Issues) inserted.
Revision 6	6/24/2003	Jeff Hanna, Greg Grimsby, Brian Traficante	Section 5 (Environments) inserted.
Revision 7	6/25/2003	Jeff Hanna	Section 1 (Programs) revamped a bit.
Revision 8	7/1/2003	Jeff Hanna	Reworked tool section. Added Tech Specs, Work Flow, and Document Storage.
Revision 9	7/2/2003	Jeff Hanna	Fixed grammatical and spelling errors found during review.
Revision 10	7/9/2003	Jeff Hanna	Changed path information for the nVidia Photoshop tools.



# CONTENTS

- 1 Art Tools ..... 1
  - 1.1 Texturing Tool ..... 1
    - 1.1.1 Additional Photoshop Plugins ..... 1
  - 1.2 Modeling Tool ..... 1
    - 1.2.1 Additional Max Plugins And Scripts ..... 1
    - 1.2.2 Optional, But Useful Plugins And Scripts ..... 3
  - 1.3 Character Animation Tool ..... 4
  - 1.4 Particle Editor ..... 4
  - 1.5 Object Viewer/Packager ..... 4
- 2 Technical Specifications ..... 5
  - 2.1 Scale ..... 5
  - 2.2 Texture Formats ..... 5
- 3 Art Work Flow ..... 6
- 4 Document Storage, Version Control, and Backups ..... 7
- 5 Technical Art Creation Issues ..... 8
  - 5.1 Surfaces, Surfaces, Surfaces ..... 8
  - 5.2 Bones ..... 8
  - 5.3 UV Coordinates and Texturing ..... 8
  - 5.4 Creating Levels of Detail ..... 9
  - 5.5 Morphing ..... 9
- 6 Art Style ..... 10
  - 6.1 High Level Concept ..... 10
  - 6.2 High Level Guidelines ..... 10
    - 6.2.1 Creatures Must Be Believable ..... 10
    - 6.2.2 Designs Must Have Impact ..... 10
    - 6.2.3 Designs Must Cover a Broad and Varied Spectrum ..... 10
    - 6.2.4 Designs Must Meet A Base Level Of Expectation For The Genre ..... 10
    - 6.2.5 Designs Must Offer The Players Something New ..... 10
  - 6.3 Snowblind And Pre-Development Sketch Art Critique ..... 10
    - 6.3.1 The Art Is Grounded In A Tolkienesque Aesthetic ..... 10
    - 6.3.2 Armor Designs Are Beautiful And Rich, But Not Exotic ..... 10
    - 6.3.3 Creature Designs Are Hit And Miss ..... 11
    - 6.3.4 Not All Location Art Is Interesting ..... 11
  - 6.4 Questions About Art Assets That Artists Should Always Consider ..... 11
  - 6.5 Final Comments ..... 11
- 7 Player Characters ..... 12
  - 7.1 Character Customization ..... 12
    - 7.1.1 Body Customization ..... 12
    - 7.1.2 Head customization ..... 12
    - 7.1.3 Head Model Strategies ..... 12
  - 7.2 Character Animation ..... 13
    - 7.2.1 Animation FAQ ..... 14
    - 7.2.2 Initial Player Character Animation Lists ..... 15
  - 7.3 Fitting Armor To Different Character Races ..... 18
    - 7.3.1 Overview ..... 18
    - 7.3.2 General Armor Fitting Strategy ..... 18
    - 7.3.3 Race Creation Guidelines ..... 18
    - 7.3.4 Armor Fitting Process in Detail ..... 18
    - 7.3.5 Armor and Character Model Customization ..... 19
- 8 Environments ..... 20
  - 8.1 World Construction Overview ..... 20
    - 8.1.1 Broad Questions and Assertions ..... 20
    - 8.1.2 World Technology Issues ..... 20
    - 8.1.3 Other Questions ..... 20
    - 8.1.4 Level Pre-Production Design ..... 20



- 8.1.5 Tileset Creation..... 20
- 8.1.6 Polygonal Mask ..... 21
- 8.1.7 Special Tiles ..... 21
- 8.1.8 Tileset Texture Creation ..... 22
- 8.1.9 Dungeon Creation..... 22
- 8.2 Interior Base Tile Set ..... 26
  - 8.2.1 Tile Properties ..... 26
  - 8.2.2 Edge specification ..... 27
  - 8.2.3 Abutment ..... 27
  - 8.2.4 Tile Visibility Capping..... 27
  - 8.2.5 Transparencies ..... 27
  - 8.2.6 Naming Convention..... 27
  - 8.2.7 Textures..... 27
  - 8.2.8 Tile Descriptions ..... 27
  - 8.2.9 Hallway Tile-Set ..... 28
- 8.3 Exterior Base Tile Set ..... 29
  - 8.3.1 Tile Properties ..... 29
  - 8.3.2 Edge specification ..... 29
  - 8.3.3 Naming Convention..... 29
  - 8.3.4 Textures..... 29
- 8.4 Mega-Tiles and Tile Templates ..... 29
  - 8.4.1 Advantages..... 30
  - 8.4.2 Disadvantages ..... 30
  - 8.4.3 Usage of mega-tiles / tile-templates..... 30
  - 8.4.4 Texturing..... 30
  - 8.4.5 Interaction of Mega-tiles and Templates With Other Mega-tiles and Templates ..... 30
  - 8.4.6 Terrain Bitmap Masks ..... 30
  - 8.4.7 Navigation Bitmap Parameters ..... 30
  - 8.4.8 Material ID Bitmap Parameters..... 31
  - 8.4.9 Bitmap Creation..... 31
- 9 Armor Creation..... 32
  - 9.1 Armor in RPGs ..... 32
  - 9.2 Core Issues ..... 32
  - 9.3 Exploration Outcome Overview..... 32
  - 9.4 Armor Segments and Core Pieces..... 32
  - 9.5 Layering ..... 33
  - 9.6 Tri-Mesh Culling ..... 33
  - 9.7 Armor Suit Styles..... 34
  - 9.8 Armor Design Requirements ..... 36
    - 9.8.1 Requirements..... 36
    - 9.8.2 Remaining issues: ..... 37
  - 9.9 Armor and the Art Pipeline..... 37
- 10 Weapons ..... 38
- 11 Items and Objects ..... 39
- 12 Lighting Requirements..... 40
- 13 Particle Effects ..... 41
- 14 Bestiary ..... 42



# 1 Art Tools

## 1.1 Texturing Tool

Photoshop 7.x will be used to develop Soul Forge. Minor revisions of the 7.x series (aka point releases) may be added after they have been reviewed. We will not be upgrading to any other major version of Photoshop (e.g. 8, 9 ...) during the development cycle.

### 1.1.1 Additional Photoshop Plugins

To augment the capabilities of Photoshop and to integrate it into our version control system certain plugins must be added to a default installation of Photoshop before it is ready to be used in our work environment.

#### 1.1.1.1 LSPhotoshopVC

Much like the 3DSMax Perforce scripts (mentioned in section 1.2.1.5), the Photoshop Perforce plugin provides an interface between Photoshop and our version control system. Its usage is mandatory. All game related 2D content must be under version control at all times. The menu structure and controls for the plugin are identical to the 3DS Max integration and Office integration tools. To install the plugin copy the [\\server\cland\utilities\Photoshop\\_Plugins\Adobe Photoshop Only folder](#) to your Photoshop \plug-ins folder. After you restart Photoshop you'll see the new menu entries in the File menu.

#### 1.1.1.2 DDS Importer/Exporter (w/DLLs)

As mentioned above, the DirectX \*.dds texture format is the format we will be using for all game textures. The only exception currently is for normal maps, which require a special format in order to function. This Photoshop plugin will save an image out in the proper format. The latest version is also required to save out properly formed normal maps from the NormalMapFilter (see below). It is a file-export plugin that will add new file-type entries to Photoshop's save requestor.

Installation of this plugin is a little more complex than anything else in this document. First, copy the [\\server\cland\utilities\Photoshop\\_plugins\File Formats](#) folder to your Photoshop \plug-ins folder. After you have done that copy the [\\server\cland\utilities\nvidia\Windows folder to your c:\](#) drive When you restart Photoshop the new file types will be in the Save and Save As requestors.

#### 1.1.1.3 NormalMapFilter

The NormalMapFilter plugin converts an image to a valid normal map. It is fairly intelligent and can create decent normal maps for simple imagery from either height or color data. It has many options for normal map generation. Documentation is non-existent, though. You'll have to play around with it to find out which options give the best results. Any generated normal maps will have to be saved out with the DDS Importer/Exporter (in the proper format) in order to be used by the 3D engine. Just copy the [\\server\cland\utilities\Photoshop\\_plugins\File Formats](#) folder to your Photoshop \plug-ins folder to install it. When Photoshop is restarted the tool will be listed in the Filter\nvTools menu.

## 1.2 Modeling Tool

3ds max 5.1 (SP1) will be the 3D modeler used for Soul Forge art asset creation. Minor revisions (aka point releases or service packs) may be added after they have been carefully reviewed. We will not be upgrading to any newer major revision of 3ds max (e.g. 6, 7, .etc...) during the development cycle.

### 1.2.1 Additional Max Plugins And Scripts

A number of internal and 3<sup>rd</sup> party plugins and scripts must be added to a base installation of 3ds max 5 before it can be used in our work environment. The plugins and scripts provide export tools for our 3D engine, integration with our version control system, controls for setting up physics proxies, and other workflow enhancements



### 1.2.1.1 Gamebryo Exporter

The installation file for the 3ds max plugin that exports data to our engine format can be found at: \\server\cland\utilities\3ds\_max\R5\_plugins\Gamebryo\GbSourceMaxPluginSetup.exe

You will also need to install the Gamebryo Scene Viewer and Gamebryo Animation Tool. Those applications can be found at:

\\server\cland\utilities\3ds\_max\R5\_plugins\Gamebryo\GbSourceSceneViewerSetup.exe

\\server\cland\utilities\3ds\_max\R5\_plugins\Gamebryo\GbSourceAnimationToolSetup.exe

#### 1.2.1.1.1 Lodestone Extensions To The Gamebryo Tools

To work around some DirectX 9 and UV coordinate exporting issues in Gamebryo we have recoded parts of the tool chain to work better in our development environment. The following steps must be followed in order to get your newly installed Gamebryo tools to be compatible with our engine art path:

- Delete the file \\3dsmax5\plugins\GamebryoMaxPlugin.dlu
- Copy the file \\server\cland\utilities\3ds\_max\R5\_Plugins\Gamebryo\LS\_GamebryoMaxPlugin.dlu to your \\3dsmax5\plugins\ folder.
- Copy the file \\server\cland\utilities\3ds\_max\R5\_Plugins\Gamebryo\NiD3DXEffectShaderLibDX810.nl8 to your C:\program files\Numerical Design Ltd\Gamebryo 1.0 Source\Sdk\Win32\Shaders folder, overwriting the existing file.
- Copy the file \\server\cland\utilities\3ds\_max\R5\_Plugins\Gamebryo\AnimationTool\_DX9.exe to your c:\program files\Numerical Design Ltd\AnimationTool folder
- Copy the file \\server\cland\utilities\3ds\_max\R5\Plugins\Gamebryo\SceneViewer\_DX9.exe to your c:\program files\Numerical Design Ltd\SceneViewer folder.

The Gamebryo tool can be found under the Utility tab within Max.

### 1.2.1.2 Havok

Havok 2 uses the 1.6 version of the Havok exporter Max plugin. The plugin works in both 3DS Max R4 and R5. Before you install this you need to make sure you've uninstalled "Reactor" if it was installed along with Max. The Reactor plugin and the full Havok plugin will not co-exist.

You can find the installer at:

\\server\cland\utilities\3ds\_max\r5\_plugins\havok\havokexporter\_16\_max45.exe.

The Havok tool can be found under the Utility tab within Max.

### 1.2.1.3 Bug Fixed UVUnwrap Modifier

The default 3ds max 5 UVUnwrap modifier has some bugs in it. A bug-fixed version was released by a third party. This file should be used in place of the default one.

Copy the file \\server\cland\utilities\3ds\_max\R5\_plugins\3rd\_party\uvwunwrap.dlm to your \\3dsmax5\stdplugins folder. Overwrite the existing file.

### 1.2.1.4 LSPlugins

The internally developed LSPlugins set is a group of general purpose and specialized plugins that aid in a number of Max related tasks. While they were conceived and written for the Driving Force project many of them will be useful in this game development cycle as well. The plugins can be found at: <\\server\cland\utilities\3ds\_max\R5\_plugins\Lodestone>.

#### 1.2.1.4.1 LSLoft



LSLoft is an enhanced version of Max's default Loft tool. It was designed for use in highway construction for Driving Force. To install just copy the LSLoft.dlo file from the server directory to your \3dsmax5\plugins folder.

#### 1.2.1.4.2 LSOptimizer

LSOptimizer is a polygon reduction tool. It has been found to be more robust and useful than Max's built in MultiRes modifier. The LSOptimizer modifier should be used as your main poly reduction tool whenever possible. To install just copy the LSOptimizer.dlm file from the server directory to your \3dsmax5\plugins folder.

#### 1.2.1.4.3 LSStitch

LSStitch takes two objects of possibly differing vertex counts and creates the polygons and vertices necessary to join them up on their open edges. To install just copy the LSStitch.dlm file from the server directory to your \3dsmax5\plugins folder.

#### 1.2.1.4.4 LSTilePainter

The LSTilePainter utility is a tool for painting seamless, tiled terrain textures on a regularized grid of a given size. In order to be used properly the LSTileMaterial and LSTileSet material plugins must also be utilized. If you need to use LSTilePainter please see Greg and/or Colin for more information on it. Copy LSTilePainter.dlu, LSTileMaterial.dlt and LSTileSet.dlt from <\\server\cland\utilities\3ds\_max\r5\_plugins\Lodestone\> to your \3dsmax5\plugins folder to install it.

#### 1.2.1.5 Perforce Scripts

The version control script package can be found at:

<\\server\cland\utilities\3ds\_max\scripts\Lodestone\VersionControl.mzp>. To install it, run the \*.mzp file from Max's "MaxScript/Run Script" menu command.

Usage of this script is mandatory, as it insures that all Soul Forge work is always under version control. Please refer to the "Art Department's Use of Perforce" (v2) document for instructions on how to use it.

### 1.2.2 *Optional, But Useful Plugins And Scripts*

The plugins and scripts in this section are not required for our work environment. They are useful tools, though, and can help to save time and increase productivity of the artists that choose to use them.

#### 1.2.2.1 NormalMapperUI

NormalMapperUI is a MaxScript front end to ATI's NormalMapper utility. It puts a useable interface on a powerful tool for normal map creation. NormalMapper/NormalMapperUI will be useful for creating normal maps for detailed high-polygon count objects (characters, environments, etc...). The NormalMapFilter Photoshop plugin (see below) will be a better tool for creating normal maps for basic objects (boxes, barrels, etc...)

The script installation package can be found at:

<\\server\cland\utilities\3ds\_max\scripts\lodestone\NormalMapperUI.mzp>. To install it, run the \*.mzp file from Max's "MaxScript/Run Script" menu command. After installation you will have to go into the "Customize/Customize User Interface" menu command and add the command to a hot-key, toolbar, or menu. The tool is named "NormalMapperUI" and is found in the "Lodestone Tools" group.

The NormalMapper installer can be found at <\\server\cland\utilities\ATI\NormalMapper\_2\_2.zip>. Please refer to the included documentation for instructions on how to use NormalMapper.

#### 1.2.2.2 Unwrap Tool

Max 5's UnwrapUVW modifier is vastly improved over previous versions. Unfortunately, some older functionality is missing in the new version. Chuggnuts, a well known MaxScripter, has



written a replacement macro script that adds a lot of the old tools and incorporates a whole slew of new functionality into the modifier. It even has a Texporter-like functionality built in.

This script is not required, but it is strongly recommended for use. To install it follow these simple directions:

- Go to your \3dsmax5\ui\macroscripts folder.
- Rename Macro\_UnwrapUI.mcr to Macro\_UnwrapUI.bak.
- Copy \\server\cland\utilities\3ds\_max\scripts\3rd\_party\unwraptool1-32.zip to your local HD.
- Open the \*.zip file in Winzip.
- Extract Macro\_UnwrapUI.mcr to your \3dsmax5\ui\macroscripts folder.
- Extract uvtools\_24i.psd and uvtools\_24i.bmp to your \3dsmax5\ui\icons folder.
- Restart Max.

#### 1.2.2.3 Uzife Lasso

While designed for Max R4, the Uzife Lasso plugin works correctly in Max 5. This tool is not required, but its usefulness will probably make it an indispensable addition to your workflow. It is simply a painting selection tool. Instead of Max's fence/crossing or click-select tools you can use your mouse pointer as a paintbrush to select items. It has controls for brush size; paint vs lasso selection, and other options. To install just copy the <\\server\cland\utilities\3ds\_max\r5\_plugins\3rd\_party\uzife\_R4.dlu> file to your \3dsmax5\plugins folder. After you restart Max the tool will be available in the Utility panel.

#### 1.2.2.4 Snowblind Plugins

Mtex.dlu, Mtex.ilk, WAXMAX.DLE, WAKMAX.ilk, and Tilecut.dlu are all plugins that are referenced by the Snowblind art assets. It's best to copy these files from the \\server\cland\utilities\3ds\_max\r5\_plugins\snowblind folder to your \3dsmax5\plugins folder.

### 1.3 Character Animation Tool

Discreet's Character Studio 4.1 will be the character animation tool for Driving Force. Older or newer versions should not be used. We may continue to upgrade to newer revisions in the 4.x series. That decision will depend upon unit testing of any new point release that may come out during the development cycle, though.

### 1.4 Particle Editor

As for Driving Force, the in house developed LexFX editor will be used to develop particle effects for Soul Forge. The current LexFX editor will need to be updated to include editing abilities for the new classes of effects we want to create for Soul Forge, though.

### 1.5 Object Viewer/Packager

Final packaging and delivery of game content will occur in our in-house developed viewer/packager (aka, "uber-editor"). Currently the specifications for this tool have not been finalized.



## 2 Technical Specifications

The following specifications need to be adhered to for all art asset creation. Adherence to these rules will help alleviate common problems such as improperly scaled 3D objects, incorrect texture formats, and mismanaged deliveries.

### **2.1 Scale**

All 3D work in Soul Forge should be done in meters. All artists should change the unit preferences in 3ds max to reflect this unit of measure. Working in another unit, and then converting will cause scaling problems, especially on skinned characters. All work should be done in meters from the start.

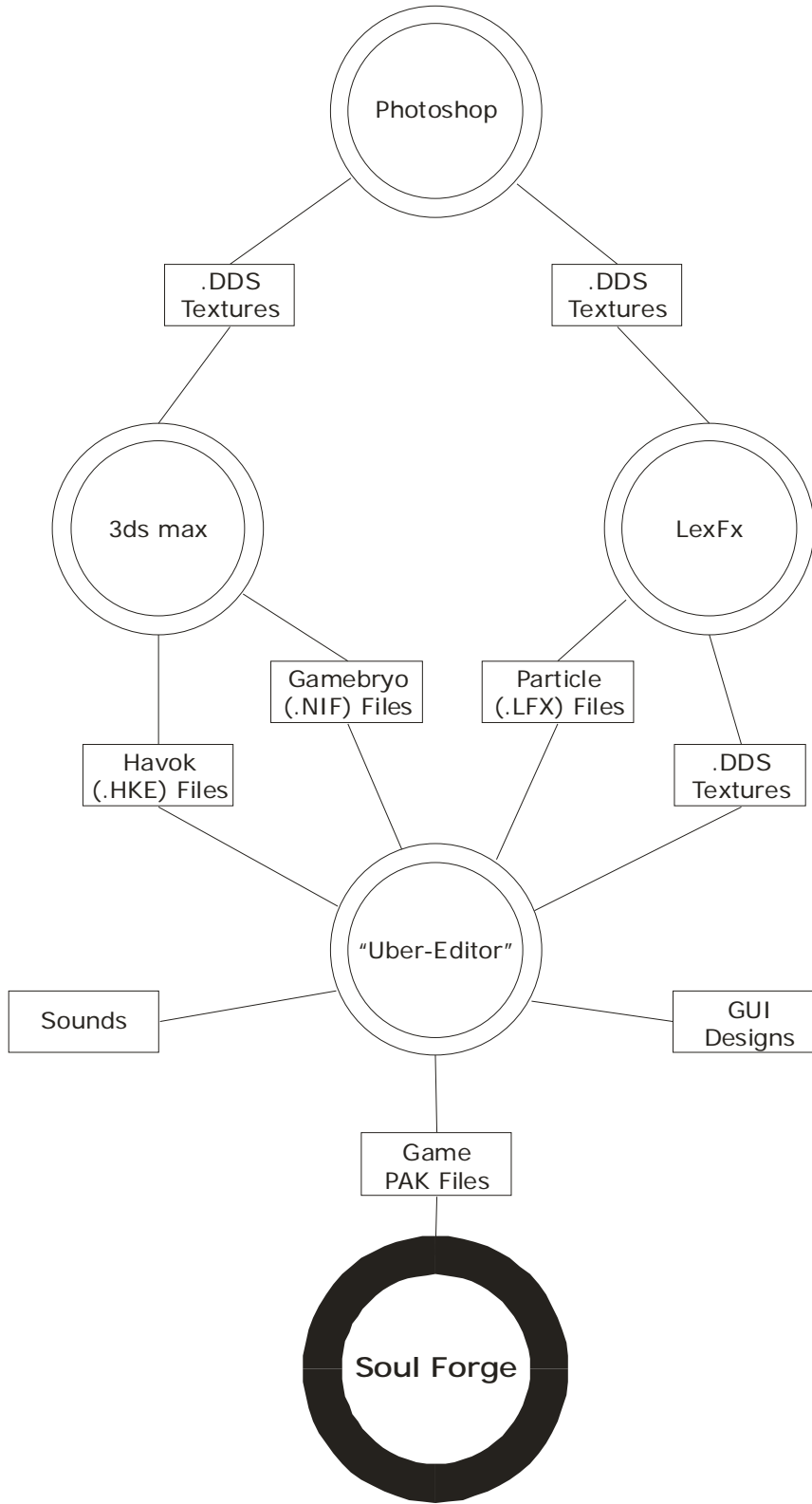
### **2.2 Texture Formats**

Compressed DirectDraw Surface textures (.DDS) will be used in Soul Forge. All artists should use .DDS format textures for all game deliverables. It is recommended that textures be created in Photoshop and saved as Photoshop documents, so that the layer information is maintained. That same texture should then be saved as a .DDS format. All 3D work should have .dds format textures in the materials, as 3D Studio will read and display them. Using .DDS format textures for as much of the art work flow as possible helps minimize any problems that will appear if other image formats get into game deliverables.



### 3 Art Work Flow

This chart shows how all of the various tools and file formats interact to produce the final game data.



## 4 Document Storage, Version Control, and Backups



## 5 Technical Art Creation Issues

### 5.1 Surfaces, Surfaces, Surfaces

3D accelerators have become incredibly fast at computing a batch of similar triangles. To keep the video card working at peak performance, though, certain rules should be followed whenever possible:

- No object should have fewer than 150 polygons. A lot of small objects are harder for the board to process than the same number of objects with more polygons. It's actually wasteful to have the CPU send a bunch of low polygon models to the board and then have the video board essentially choke on them.
- The number of surfaces we want to render per frame is in the 500-1000 range. A surface becomes one batch push to the video board. It takes CPU time to prepare this batch for the video board. Going over 1000 surfaces per frame means that the CPU will be calculating batch pushes that the video board probably won't even get to rendering.
- Each material you apply to an object in max directly equates to a new surface being created when the game is run.
- Creating levels of detail for an object down to the point where the farthest LOD has fewer than 150 polygons will not improve the performance of the game. This is tied in to the first item on this list. An object that that few polygons is actually more expensive for the board to render than one with more polygons.

### 5.2 Bones

We will be using the Skin modifier in max, not the Physique modifier. Skin is much more forgiving in terms of mesh changes. It also allows you to add bones later. If you graft new objects onto skinned objects you can steal the skinned object's vertex weightings.

We can do level of detail setups for bones. This way we can remove bones as the object recedes from the camera. This will help performance for objects that are too small on screen to resolve all of their details.

Twenty-six (26) bones is a good target for the highest bone LOD. This is based on the fact that hardware vertex shaders that can do bone animation address up to twenty-six bones per partition. A partition is a collection of vertices assigned to a bone. Having more bones than this is supported, but going over twenty six bones forces a second partition to be created. Extra partitions per object are not a performance killer, but fewer is better.

Gamebryo has a partition analyzer script that gets added to 3ds max. If you add it to your export tool chain you can see a report of where your weightings are falling.

### 5.3 UV Coordinates and Texturing

Gamebryo supports up to ten (10) UVW channels per object. The more UVW channels per object, though, the greater the overhead for processing those triangles. If the number of UVW channels per object is greater than the number of pixel operations a video board does in a single frame, a second rendering pass will be needed. As rendering passes per frame go up, performance goes down.

Multi-sub object texturing is bad, for the same reasons as above. They also slow down the viewport refresh in 3ds max.

In general, objects should be textured with just one (1) material. That material can have different textures within it, though (base, dark, normal, specular, etc...). Textures can be blended together in a multi-pass scenario. If the number of textures is greater than the board can manipulate in a single frame then extra rendering passes will be needed and performance will be affected.



We've found that the new UVUnwrap tool in 3ds max 5.x to be very powerful. We've also added some Plugins that extended it (please see section 1.2.2.2). The new Stitch and Unfold functions are extremely powerful and quick.

Gamebryo supports UV coordinates greater than one (1) and also negative values. This allows you to tile a texture to infinity if you need to.

Gamebryo does not currently support cropping in the texture's UV space. We will be modifying the Gamebryo code to support both cropping and rotational angles for the W coordinate. We created these modifications for NetImmerse and feel confident that the Gamebryo code can be extended in a similar manner.

Working at double the intended game resolution for textures is advised. This gives you plenty of working space. We can either size the textures down at ship time, or use the game's MIP-mapping controls to just display a lower resolution representation of the texture as the default res. The high resolution source material also gives us the opportunity to ship a high resolution texture pack after the game has been out for a while and video boards increase in complexity.

As mentioned in section 1.1.1.2, we will be using DDS format textures for all images in the game. We generally use DXT1, which is a hardware accelerated format that, in this case, has no alpha channel component. A DDS compressed texture uses the same amount of memory in VRAM as it does on disk. Using a higher order DDS format (DXT3 or DXT5) allows for alpha channels. DXT3 doubles the size of the image (compared to DXT1). DXT5 quadruples the size of the image. Alpha is expensive. Please use it sparingly.

### ***5.4 Creating Levels of Detail***

We use levels of detail as a way to improve performance. It is a strategy to insure that objects on screen are of a level of complexity appropriate to the screen real estate they take up. As mentioned in section 1.2.1.4.2, we will be using our internally developed plugin, "LSOptimizer" to aid in the creation of lower polygon count level of detail objects.

### ***5.5 Morphing***

While morphing is supported in Gamebryo, it carries some large performance costs. Currently the use of morphing in Soul Forge is questionable. As more data regarding morphing comes in we may decide that it is a tool that could be used during the development of the product.



## 6 Art Style

### 6.1 High Level Concept

The Art Style section is to relate the artistic style that we have been pursuing for Soul Forge so far. The guidelines and critiques in this section should act as the basis for all art design and creation.

The art design for Soul Forge should offer controlled exaggeration, accentuating striking features while offering a believable cohesive design devoid of awkward incongruous proportions, surface treatments, or anatomy.

### 6.2 High Level Guidelines

#### 6.2.1 Creatures Must Be Believable

Their texturing, their animations, and their physical weight distribution must feel like they could exist. These characters are NOT cartoons. This does not preclude exaggeration. It simply grounds it in a desire to describe a creature or place that could exist in a fantastical version of our own flesh and blood world.

#### 6.2.2 Designs Must Have Impact

Either through telling a story or invoking an emotion the designs must engage the players. Art is about emotional response. If a design doesn't evoke a strong response then it is weak art.

#### 6.2.3 Designs Must Cover a Broad and Varied Spectrum

Variety is the spice of life. With finite content the difference between the various designs must be strong enough to warrant their existence. Additionally, varied designs will impart different game play interaction. Architectural designs must offer different game play challenges and emotional responses. Creatures need to cover a broad offering of combat styles.

#### 6.2.4 Designs Must Meet A Base Level Of Expectation For The Genre

As a Fantasy RPG, players expect certain fiction to be present. Dragons look like D&D dragons. Dwarves look like Tolkien dwarves. Etc. This, along with the other requirements, makes for a tough design challenge.

#### 6.2.5 Designs Must Offer The Players Something New

This is somewhat incongruous with 6.2.4, but when opportunities arise to offer a new look we should take it.

These guidelines are exciting and should help us to create excellent fantasy art. It will be a challenge to meet all of these guidelines consistently, though.

### 6.3 Snowblind And Pre-Development Sketch Art Asset Critique

#### 6.3.1 The Art Is Grounded In A Tolkienesque Aesthetic

This is a good descriptor because it covers 6.2.4 and 6.2.1 very well. As "Lord Of The Rings" was very successful, we gain some of 6.2.2 and 6.2.3 as well. We fail on 6.2.5 if we follow just a Tolkienesque response. This could be fed, however, into a modified mission statement of:

To make original art suitable for a Tolkien movie.

#### 6.3.2 Armor Designs Are Beautiful And Rich, But Not Exotic

Only one or two designs have a lot of action-figure, cool-factor going for them. The designs as they stand will be a great source for generic armor types, but some treatments of wilder armor are needed. The helmets are very well done. We should strive to capture that character in all of the weapons and armor.



### ***6.3.3 Creature Designs Are Hit And Miss***

The creatures, while well rendered, are hit and miss from an impact standpoint. A handful of designs are great and should be used. Others may prove useful if re-worked. Others are too commonplace or are too devoid of either newness or personality to keep.

### ***6.3.4 Not All Location Art Is Interesting***

The Barbarian buildings are neat. Their construction with bones, stone, and rope are strong in weight and visually exciting with their spiky profiles. The dwarven interiors are majestic. The human Tudor style is blah in the design sketches but if textured and modeled with more whimsy could have hope.

## ***6.4 Questions About Art Assets That Artists Should Always Consider***

- Could this exist in a Tolkien movie?
- Is this cooler than other versions of this I may have seen?
- Would this make a kick-butt action figure if posed dynamically?
- If a landscape or lair, is there sublime majesty in the design? (Websters: Majesty: greatness or splendor of quality or character, sublime: c : tending to inspire awe usually because of elevated quality (as of beauty, nobility, or grandeur))
- Will the design promote those memorable, “You gotta see this or fight this” response?

## ***6.5 Final Comments***

We will pull inspiration from D&D artists, Brom, Vallejo, Frazetta, LoRT and many others.

Don't be scared by the use of Tolkien in so many in my descriptions. The goal is to use that descriptor solely as a sanity check for expectations of the genre and for believability. As a descriptor it shouldn't limit darker, wilder Hellraisery, or sarcastic responses.



## 7 Player Characters

### 7.1 Character Customization

#### 7.1.1 Body Customization

Currently, we have not found an acceptable art workflow for allowing players to change the physique of their characters. Gamebryo doesn't support morph targets on skinned, multi-textured objects, which offers a technical hurdle as well. If this is addressed at some future date, this idea could be revisited. It would require a morph target to be made to represent a thin physique and a muscular physique. All armors for all races would need to have a matching morph target made to fit.

#### 7.1.2 Head customization

There is a strong desire to allow players to customize the facial features of their avatar. Just going through this creative process gets the player into the game and their role. As mentioned before any customization that impacts armor fit needs to be avoided until a time when an art path is resolved to work around the tremendous art strain adjusting armors to fit variable flesh models would cause.

##### 7.1.2.1 Appearance Components

Here is a brief overview of the components offered in the customization of heads. Later below these will be discussed in detail.

Head Model	This is the polygonal model of the head. It uses its own texture.
Beards	This is an integral mesh model variant of the head model. It is textured using a decal layer style from the default shader. Beards tentatively have no alpha, so no wispy strands at the end of the beard.
Hairdos	This is a mesh model.
Skin Color	Skin color is blended in with the same shader code as detail textures. That is 50% gray modulates nothing. Other RGB values modulate the underlying base map.
Tattoos	These are non-alpha'ed bitmaps that mimic the Dark Map channel of the default shader. They can color and darken skin just like real tattoos. There is a desire to explore animated tattoos.
Scars	These are non-alpha'ed bitmaps that mimic detail texture blending of the default shader. They modulate the values present in the base map.
Face Paint	A decal overlay texture with alpha that covers all other layers

##### 7.1.2.1.1 Issues To Always Consider

- beardhair/eyebrow color correlation
- Hairlines

#### 7.1.3 Head Model Strategies

##### 7.1.3.1 Discrete Head Models

(Considered but abandoned)

The first strategy for customization is offering discrete head models. These heads would have varying proportions and would be a sampling of exotic and supermodel faces.



The downside to offering discrete head models relates to the fact that the beard variants already require a separate model for each beard type. Delivering a discrete head variant as well as a discrete beard model gets into a large number of iterative head models

#### 7.1.3.2 Morphed Head Models

The head model of the each race/gender has a number of morph targets built in (perhaps a dozen). These morph targets would have varying proportions and would be a sampling of exotic and supermodel faces. The player could scrub through a number of controller points such as for the nose, eyes, mouth, facial width, etc and could combine elements of the morph targets to their desire.

Currently we have no ability to morph the head since you cannot morph an object with more than one material in Gamebryo.

#### 7.1.3.3 Morphed Head Deformation

Another potential customization method leverages morphing as well. Instead of having a set of discrete head models built into the morph range, the morph ranges attached to each facial area represent ranges of proportions from thin to fat. This offers something akin to what Eve Online offered.

##### 7.1.3.3.1 Beards

For the male characters, a sampling of beards will be offered from close trimmed goatees to full dwarven beards. In order to keep the number of surfaces to a minimum, the beards are delivered as variants of the master head models. Unlike hair which could be turned off when a full helm is worn, beards will need to be on all the time.

In order to support the various beard lines and hair colors, beards leverage the decal channel (or will need a shader equivalent). A decal blends upon the layer below it using its own alpha mask. The alpha can define the wispy, feathered edge of where the beard starts upon the cheeks. To change the beard color, a different beard texture is loaded into this slot. These beard color options will match up the hair color options. In the Bearded Lass image, the beard is part of the head mesh. Notice the nice feathered edge of the beard. No polygonal mutton chops here!



*Bearded Lass*

##### 7.1.3.3.2 Hairdos

Soul Forge will need to offer a number of hairstyles for each gender/race. Since hair needs to be removed under full helms, it will need to be a separate tri-mesh. This also reduces iterative head/hair combo deliveries.

## 7.2 Character Animation

One thing that greatly separates Adventure Style Console Games from current MMORPGs is the way a character animates in combat. Most MMORPGs do not have dramatic combat. It is vital to provide a



dramatic/cinematic performance for the players during combat especially since the game's center point is combat.

### 7.2.1 Animation FAQ

- How many different types of animations do we want to blend together and what limitations do we have in Gamebryo?

The current understanding is that you blend two animations at once. Blending is mainly used for transitions from one discrete animation to the next.

- What should we do with weapons when a character needs to perform a non-combat task that requires both hands, or even one hand? Ex: Lock-picking, trap disarming
- 
- How do we minimize one-off animations? We have animation blending and layering at our disposal. Do we need a running-while-firing bow animation or can we layer two separate animations?

Animation blending has been explored. The workflow is known. Its usefulness is very questionable since very few combat animations are so localized. Most combat animations affect the whole body. We can use layering in MAX if need be, where it can be tweaked and exported as a discrete animation.

- What are the major weapon styles as they pertain to how they would be used in combat and need to be animated? You have bows, thrown weapons, bashing weapons, piercing weapons, etc. Do we have staves aka Dungeon Siege that are wielded like bo sticks? Do we have different unarmed animations like generic punches vs martial art moves?
- How many spell casting animations do we need? Do we want one for casting a projectile spell, another for an AoE, and another for a self-buff?
- How much of the Snowblind animation work can we leverage?
- How long does it take to develop new animations?
- What is the base set of animations for a monster?
- How art-time expensive is it to develop special animations based on skills or otherwise have flourish animations?
- With the list below and any others we come up with, how would we prioritize these animations in importance?
- How would two-handed weapons work in the timing system of the game? Do two-handed weapons attach sequentially or in combination, or potentially both?
- How is the timing of combat handled? How long is a "turn" in Soul Forge? How does this interrelate to our expected game metabolism (pace) and to lag? These all affect how long an animation sequence needs to be.
- How do animations map across the races and sexes? Can we use the same animation without tweaking?
- When will a character need to put away their weapons for animations? When can a character attack? Can you attack while treading water?
- Do characters draw their weapons or do they simply appear in their hands?
- Can animations translate the character away from their original coordinates or do all animations need to return to the same spot?

The initial response from Kyle is that this is not ideal, but if the design calls for it enough, it could be allowed.

- How do physics work into animations? Can a shield stop a sword swing animation? Even if we could have this interaction, will it give pleasing animation results?
- How do we make characters "look at" or swing at their target? If a player attacks a small ground dweller, does the animation engine need to rotate the character at spine01 to orient downward towards the target?



## 7.2.2 Initial Player Character Animation Lists

### 7.2.2.1 One-Hand Sword

- Idle x (Character damaged) x multiple idle animations
- Emotes
- Running x (Character damaged)
- Walking x (Character damaged)
- Sit?
- Jumping
- Climbing(walls, ladders,ropes)
- Swimming
- Treading Water
- Burdened
- Crouching
- Character death x ?
- Character hit x ?
- Dodge
- Parry
- Block
- Stealth
- Draw Melee Weapon
- Replace Melee Weapon
- Forward slash
- Over head slash
- Left Slash
- Right Slash
- Upper Cut Slash
- Thrust
- Special x ?

### 7.2.2.2 Two-handed fighting

- Idle x (Character damaged) x multiple idle animations
- Emotes
- Running x (Character damaged)
- Walking x (Character damaged)
- Sit?
- Jumping
- Climbing(walls, ladders,ropes)
- Swimming
- Treading Water
- Burdened
- Crouching
- Character death x ?
- Character hit x ?
- Dodge
- Parry
- Block
- Stealth
- Draw Melee Weapon
- Replace Melee Weapon-left hand
- Replace Melee Weapon-right hand
- Forward slash -right hand
- Forward slash -left hand
- Over head slash
- Left Slash



- Right Slash
- Upper Cut Slash
- Thrust
- Special x ?

#### 7.2.2.3 Two-Handed Sword

- Idle x (Character damaged) x multiple idle animations
- Emotes
- Running x (Character damaged)
- Walking x (Character damaged)
- Sit?
- Jumping
- Climbing(walls, ladders,ropes)
- Swimming
- Treading Water
- Burdened
- Crouching
- Character death x ?
- Character hit x ?
- Dodge
- Parry
- Block
- Stealth
- Draw Melee Weapon
- Replace Melee Weapon
- Forward Slash
- Over head slash
- Left Slash
- Right Slash
- Upper Cut Slash
- Thrust
- Special x ?

#### 7.2.2.4 Shield

- Block
- Bash

#### 7.2.2.5 Spear

- Idle x (Character damaged) x multiple idle animations
- Emotes
- Running x (Character damaged)
- Walking x (Character damaged)
- Sit?
- Jumping
- Climbing(walls, ladders,ropes)
- Swimming
- Treading Water
- Burdened
- Crouching
- Character death x ?
- Character hit x ?
- Dodge
- Parry
- Block
- Stealth
- Draw Melee Weapon



- Replace Melee Weapon
- Forward Thrust
- Left Thrust
- Right Thrust
- Upper Cut Thrust
- Parry
- Block
- Special x ?

#### 7.2.2.6 Bow

- Idle x (Character damaged) x multiple idle animations
- Emotes
- Running x (Character damaged)
- Walking x (Character damaged)
- Sit?
- Jumping
- Climbing(walls, ladders,ropes)
- Swimming
- Treading Water
- Burdened
- Crouching
- Character death x ?
- Character hit x ?
- Dodge
- Parry
- Block
- Stealth
- Draw Ranged Weapon
- Replace Ranged Weapon
- Draw Arrow
- Nock Arrow
- Release Arrow
- Relax Shot?
- Special?

#### 7.2.2.7 Thrown Weapon (hammers, axes, darts, etc)

- Idle x (Character damaged) x multiple idle animations
- Emotes
- Running x (Character damaged)
- Walking x (Character damaged)
- Sit?
- Jumping
- Climbing(walls, ladders,ropes)
- Swimming
- Treading Water
- Burdened
- Crouching
- Character death x ?
- Character hit x ?
- Dodge
- Parry
- Block
- Stealth
- Draw Thrown Weapon
- Replace Thrown Weapon
- Throw



- Throw 2 at once

### **7.3 Fitting Armor To Different Character Races**

A full discussion of armor creation is covered in the armor section of this document. The issues surrounding fitting armor to different character races needs to be discussed here, though, as it affects both armor and player character model creation.

#### **7.3.1 Overview**

One of the greatest art asset challenges we will have in Soul Forge will be delivering armor that fits the various combinations of race and gender. With three races and two genders we have to fit any armor piece to six flesh models. During the exploration phase of development, a strategy was sought to minimize the amount of re-work time required for fitting armor across these disparate models. The strategies are detailed below.

#### **7.3.2 General Armor Fitting Strategy**

The general strategy for fitting armor across the races will be to apply to the armor a FFD (Free Form Deformation) Matrix and thereafter push and pull the FFD's control points. This is done to fit the armor to the underlying flesh model. This strategy was tested by taking the armor suit on the Snowblind elf and fitting it to the female human. The exact steps will be given later below. From this resulted a time estimate of 1 hour of manipulation to fit armor to a disparate model.

#### **7.3.3 Race Creation Guidelines**

If some rules are enforced when the models for the races are created, many armor pieces will require little manipulation to fit. Here are some suggestions:

- Hands and forearms should be derivative models when possible. For example, the elven hands could be identical to the human hands, except for a simple scale modifier, that is easy to reproduce, and this scale in "burned in" with a reset Xform. This requirement is suggested because hands would be difficult to fit and SKIN deform accurately. It will save time when fitting armor.
- As much as possible, major sections of the physique are the same size across the races, or at least a uniform scale modification only.

#### **7.3.4 Armor Fitting Process in Detail**

- Load the reference flesh model you want to fit armor to. Save out this file as a new, working file name.
- Assuming your armor file is modeled in the reference pose, Merge in the armor mesh objects.
- Place your flesh model in the reference Figure mode pose.
- Begin altering the armor mesh to fit the flesh model. Suggested tools are FFD 3x3 and 4x4 modifiers. Also move vertices with Soft Select set to .1m and Edge Distance set to 5 or so. You may need to add a dummy box to your scene. Rotate this to be parallel with the limbs of your model. You can then use the Pick Coordinate system and select this box to use its pivot system aligned with the limb.
- You may find you want to leverage your work on the left leg and arm to the right side. You can do this by deleting the limbs you didn't fix. Mirror clone the armor. On the clone, delete everything but the limbs you want to keep. Attach this piece to your original armor.
- Be careful if there are UV differences across the breastplate, as this area may not be suitable for mirroring.
- Select the flesh model and, beneath the Skin modifier, attach the armor. This will cause skin to do a nearest neighbor skin weighting which will offer a big head start on proper skinning.
- Tweak the Skin vertex weight values as necessary for the armor's gratuitous spikes, pads, etc. that didn't obtain proper weights.
- Rename the mesh model to an appropriate armor name.



### *7.3.5 Armor and Character Model Customization*

We investigated using morph targets to allow players to customize the physique of their model. The main hurdle in this is specifically the morphing process. The real difficulty comes from reproducing the deformations done to the flesh model on the armor model. Obviously if the flesh-model could potentially be customized between a very thin and a very fat model, the armor must be formed similarly to fit.

After much exploration in MAX, we found no way of leveraging the deformations done to the flesh model onto the armor. This would force the deformations to need to be reproduced on every armor. Currently it looks too costly to allow players to customize the physique of their character because of the impact on the armor.



## 8 Environments

### 8.1 World Construction Overview

The purpose of this section is to describe scenarios for assembling the art needed for Soul Forge environments and offer strategies for placing game function entities, props, and other actors within the world. The scope of this document is not to detail how individual assets will be made. It will deal more with how the separate components will be combined. It also only covers functionality of systems in broad strokes.

This section is offered as a “best guess” description of our current thinking concerning this process. It is geared towards the art workflow for Soul Forge. It often mentions functionality that may be difficult to implement, but hopefully describes such functionality in enough detail and context as to make the evaluation of such systems possible.

#### 8.1.1 Broad Questions and Assertions

- Environments are to be created from tile-pieces. A tool is needed to arrange these pieces into our complete environments.
- There is a desire to group a larger subset of tile-pieces into a “special-tile”.
- A tool is needed to place set-pieces and monsters.
- A tool is needed to define lines used to trigger events, block monsters, and perform other game logic functions.
- Environments will need to be “pathed” and a quick feedback/editing loop is needed to display the paths and fix any errors in the masks.
- Material masks. Are these processed into a non-image map form?
- Edge specification. Where is a tile-piece processed (either algorithmically or manually) and in what file is this stored?
- Are roads delivered as discrete tile piece variants?
- How do outdoor terrain special-tiles fit together? Is enforcing a no-grade rule on outdoor special-tiles acceptable?

#### 8.1.2 World Technology Issues

- Threading. Is there a sweet spot size for a terrain chunk that performs well in a threading situation?
- Normal boundaries. How are these smoothed?
- Shadow calculations. Are lighting effects on STATIC geometry rendered off-camera as chunks are threaded in and then cached?

#### 8.1.3 Other Questions

- What is the workflow like from tile pieces to game?

#### 8.1.4 Level Pre-Production Design

Before a dungeon is created, a schematic design is made showing aesthetic highlight areas, encounter areas, boss locations, and game-play diagrams. This gives the designer a guide to follow when they approach the Uber Editor. This list should detail any one-off set pieces that may be needed as well as any one-off monsters.

If the dungeon design requires a new tileset, then an artist will need to do production sketches of the environment. These will describe some of the structural details, the surface quality, some of the architectural detailing, and potential feature pieces.

#### 8.1.5 Tileset Creation

In order to represent environments in SF, a tileset must be made of interchangeable, interconnecting sections of the dungeon. These 3D tiles are made in a particular assortment to cover the various angles and grades needed (as specified in the Art Design Spec). You have walls, convex corners, concave corners, floors, elevation change pieces (stairs), among the many pieces. These pieces are UVed. A shader is assigned to them. Base map (and probably normal map)



textures are made. The tile pieces have a poly budget they must fit within. Some pieces are fully-interchangeable. Others only fit against a sub-set of tiles. These pieces will have a naming convention in MAX that helps identify what shape it describes. The artist constructs all pieces within the same MAX file.

For threading purposes there is a desire to save off these tile pieces into individual NIFs. Currently there is no provision in MAX or the Gamebryo Exporter for this.

#### 8.1.5.1 Variants

Most tile pieces will have multiple variants created. These variants come in two flavors. The first style of variants is meant to break up the repeating nature of tiles. They have surface variations to make the piece not look like its variant neighbor. The second variant style incorporates some structural detail in the variant missing in other pieces. You might have a wall variant that has a niche and statue. You might have a floor variant that has a pedestal on it.

The key to deciding when a tile is different to warrant a new tile name (and not just a variant suffix) is whether shuffling the piece with other variant pieces it is based off of would produce pleasing results. Would the dungeon look good if it randomly mixed windowed wall sections with normal walls? Would it look too chaotic? If a set can be interchanged randomly without making for weird arrangements, then it is a good variant.

#### 8.1.5.2 Morph Variants

A special kind of variant which needs to be explored is the morph variant. This is mostly germane to outdoor tiles. The morph variant is a tile that has a morph animation that covers a range of interior vertex arrangements. For example, a morph animation is generated for a rolling hill special tile. At frame 0 the hills are at their lowest relief. At frame 100 the hills are at their highest relief. When the morph variant is added to the workspace, a time-line slider at the top toolbar activates and the level designer can set the tile to the frame he or she wants.

Instead of offering relief alterations, the morph variants could scrub between morphs generated by a noise controller-perturbed mesh. Scrubbing through the timeline gives fairly different topologies. The benefit of the morphing strategy is that it gives lots of variability, while having fewer assets in our footprint. Obviously each morphed variant would need to become its own unique instance.

#### 8.1.6 Polygonal Mask

Each indoor tile will have a black polygonal “roof” mask that covers it. This polygonal mask blocks view into rooms players have not yet visited. It also backfaces room structures the camera should not see through.

#### 8.1.7 Special Tiles

Tiles that break the usual interconnection rules will be needed. These pieces are called special tiles. A special tile is a tile that breaks the following rule:

- Aspect ratio is larger than 1 slot on a side. A waterfall that is 3 tile slots tall would be a special tile.

Other rule breakers may later be added to the list, but aspect ratio is currently the only differentiator.

Special tiles, again, still must fit into the edge specification scheme, so there must be some tile piece that is a valid connection to each edge.

Special tiles are a product of MAX. They are mesh geometries that produce topology that could not otherwise be produced with traditional tile pieces.

It is strongly suggested that rooms and lairs not be created as special tiles in MAX, but instead be made in the level editor augmented by variant tiles made especially for the lair. There is nothing



preventing an artist from exporting an entire room from MAX. This room would, again, be a special tile.

### **8.1.8 Tileset Texture Creation**

There is a strong desire to have as few surfaces per tile as possible. This requires that the number of materials per tile be minimized. Shooting for a single material per tile would be a good goal. No tile should need more than 4 materials.

Textures should be able to be rotated. There is a special art flow to generate a texture in this manner. Once done the texture is seamless with itself even if rotated or flipped. Since the tile can be rotated, the textures must support this as well.

Most textures will use DDS DXT1 with three mip-map levels.

For the quality of visuals we are aiming for, normal maps will be needed for at least the indoor terrain tiles.

Currently indoor materials will need shader support for base maps, normal mapping, and lighting from the light manager. Other blendings may be needed for special materials, such as water, and shiny metal, which both may need environmental mapping of some sort. Self-illumination would be needed for lava and projected texture support may be desired for light gobo effects.

A scheme for organizing textures needs to be worked out. One strategy is to have textures organized by environment (such as a directory for tundra textures, stone dungeon, etc...) Another strategy is to have a material based organization (such as, metals, stonework, earth, etc...). Whatever we do we want to support streaming tile loading and also not be in the situation we were in with Driving Force where textures were included in an arena's texture directory as well as the common texture area.

#### **8.1.8.1 Tile Bitmap Masks**

Each tile will have an indexed bitmap of pre-defined colors that is used to define navigational characteristics of a tile at a fairly low res (possibly 60x60 pixels).

Each tile will also have a material bitmap mask that defines what surface is represented throughout the tile (such as stone, grass, metal, water, etc...).

The resolution of these masks can be adjusted to meet any memory footprint or speed considerations. These indexed images may be processed by a to-be-determined utility into a form more digestible by the game.

#### **8.1.8.2 Set Pieces**

Each environment will have a set of placeable decorative objects such as sconces, pedestals, statues, torture racks, vases, crates, etc that are textured and sculpted to fit the aesthetic style of the area.

These set pieces follow the same surface restrictions as tiles.

Set pieces are saved in separate NIFs.

### **8.1.9 Dungeon Creation**

With a layout schematic, tileset geometry, and set pieces available; the level designer will need to assemble these objects into a sizable chunk of the game world. Currently MAX does not offer the accessibility for non-artists to generate environments, nor does it have the ability to visually browse external resources and place them into a scene easily, from a palette, for instance. A tool, which has been referred to as the Uber-Tool, will need to offer this functionality. The Uber Tool may be the proper place to offer other, additional functionality mentioned earlier.



Let's look at a "narrative" that describes some workflow at this point. We'll assume some basic level editing functions such as a 3D window workspace, a 3d tile previewer that is linked to a scroll-list. Optionally, a 3d representation of each piece is given in a scroll window still accompanied by a text list.

#### **8.1.9.1 An Environment Artist's First Dungeon**

The level designer loads up the level editor. He is starting a new dungeon. He has a new tileset that he wants to use. Right now all those tiles reside in a folder as separate NIFs.

The textures for these tiles are either in a texture subfolder, or are in their own folders elsewhere in the development tree under material categories. The Level Editor will need to be able to look to multiple places perhaps defined by a path .ini.

##### *8.1.9.1.1 Edge Specification*

At this point, the tiles have not been processed for their edge specification. Before the artist can begin using some of the speedy placement tools, the edge specification for each tile will need to be defined.

It is undetermined how much we can leverage naming convention on the tiles in MAX for edge specification. Due to the 3d nature of these tiles, a naming convention may be too complex to use in all cases. Perhaps naming convention could be used to identify simple pieces, such as floors, walls, and corners. Pieces that cannot be name in a categorical way may be named "wall\_undefined", or "floor\_undefined". Then in the Level editor, linkage can be set up manually, such that the artist places an undefined 3d tile in the workspace window next to the tile it fits. He can then select both and hit a button that says "Define edge correlation single" or "define edge correlation all similar". The former validated the undefined piece as a connector to the piece in the scene; the latter validates it with all pieces that share the same subset of tiles that links on that tile's edge.

However the edge specification is defined, it needs to be saved out in a file that is bound somehow to the tileset it describes.

*----All of this needs to be worked out.----*

##### *8.1.9.1.2 Tile Drawing Modes*

Once the edge specification has been made, the Level Editor becomes much more powerful than an asset browser. Edge specification opens up "smart" editing modes. Using the first mode "Contiguous Mode", the artist selects a straight wall section. He begins to draw in the 3d workspace. As he traces across the grid, the Level editor chooses corner pieces and straight wall pieces as necessary to keep the wall contiguous. This may work best if the modal functionality could be bound to a key modifier such that drawing with just the left mouse lays down the tile as is, without care to "auto contiguous" mode. Holding down the "shift" key enters "Contiguous mode".

The level editor in "fit filtering" mode enables a filter that ghosts and shuffles to the bottom of the browser pieces that don't fit with the currently selected piece. The artist can click on a cliff, and the browser window only shows the pieces that could join with that cliff piece.

This mode is essential to quick editing. How exactly it functions may not be as described above. The artist may need to define which edge they which to match up. It might be more easy to show which neighboring slot they are looking to fill. This may require an "alt click" function to mark a slot in yellow, not red, thereby giving the "fit filtering" an incoming tile and an outgoing slot.

A variant on the drawing mode is the "Auto shuffle" mode. When holding down the mouse and drawing tiles in the 3D window, this mode will randomly choose from the variants available of the tile pieces being drawn. Repeatedly single-clicking on a tile already in the viewport will shuffle the variants of this tile, if this mode is "on".



#### 8.1.9.1.3 Prefabs (what were formerly called Mega-tiles)

After some time, the level designer should have created an area they are pleased with. They have the basic structure of the area created. Whether this is an entire sprawling level, or just a single, special encounter-room, the level designer can save out this set. If the artist feels this data could be useful as a “prefab”, they can save out this data set as a file that associates this collection of tiles together.

The usage of the term “mega-tile” is being abandoned to mitigate the confusion of the many possible data-sets that could justify being called mega-tiles. You have tiles that are larger than 1x1x1 as they come out of MAX. Artists may deliver a large, rolling hill as a 4x4x2 tile. This tile is larger than the standard tile size. This could desire the name “mega-tile”. An artist in the editor defines a collection of tiles into a set representing a commonly used area, or prefab as mentioned above. This area appears in the tile browser. Is this a mega-tile? An artist links several tiles that make up a room to a parental node for lighting listing, and polygonal mask removal’s sake. Is this a room, or a mega tile? It may prove advantageous for performance to optimize a “room” with Gamebryo’s optimization tools into as few component objects as possible. Once this is done, is the room then a “mega-tile”?

Once saved out, this selected group of tiles appears in the tile browser as a prefab.

It may prove more useful to have prefabs viewed in a separate browser pane since they are not an essential building block. It would be like having a bunch of finished Lego walls thrown in with your big bucket of Lego. In all likelihood you’d pull out the big walls so you could pull from them separately and to get them out of the way of viewing your basic Lego pieces.

#### 8.1.9.1.4 Rooms

There are many reasons why functionality is needed to group tiles in the level workspace together. A “room” is such a collection. The black Fog of War capping needs some way of knowing what collection of caps to remove as a character walks through a level. Defining rooms also gives us a great deal of granularity in the lighting system as the scope of each light can be controlled to include only objects within the room’s node tree. Rooms also have the potential for optimization by combining tri-meshes of like material together and thereby reducing the number of surfaces in view.

The Level Editor will need to have a grouping function it is such that a selection of tiles in the workspace is linked to a parental node. The level editor could change the default name given to this node to “boss\_room”. Once a room is grouped, it gets a nodal “handle” in the 3d window, and perhaps a bracketing bounding box, that can be used to select and move the room as a whole. This section could be saved out as a prefab/mega-tile.

#### 8.1.9.2 Additional Functionality

You can select a tile. The 3D browser window has a “selected Piece” pane. With the tile selected you can click on other tiles in the browser and it will swap out the piece that is selected.

Rooms can be rotated.

Tiles within a room group can be exchanged for another tile piece as detailed above, but pieces cannot be explicitly deleted or added. The artist would need to delete the group node, reselect the tiles wanted, and “regroup” the room.

#### 8.1.9.2.1 Road-Quadrants

A road quadrant is a large collection of outdoor terrain tiles that fulfills a large-scale tiling approach to representing the vast distances between cities. This area represents an encounter area zone and would need to be at least 40x40 tiles (that’s about a quarter mile). These tile-quadrants represent a swath of road and enough padding on the sides to feel like the world continues. These road quadrants come in “straight” runs of road and in “corner” runs that enter from the top edge and exit 90 degrees to the right. There would be numerous variants of each of these. The level designer would rely upon tile variants, morph variants,



special tiles of large, rolling hill areas, tile painting, and foliage set dressing to make the entire road-quadrant feel like it represents a contiguous, but never repeating space.

#### 8.1.9.2.2 *Texturing of Outdoor Terrains*

There is a strong desire from the art team to allow textural painting of terrain pieces in the Level Editor. Being able to “tile Paint” in a similar fashion as the MAX plugin offered would allow the level designer to add variability to the terrains and add some unifying textural features across tile boundaries. Roads could be painted on instead of relying upon preset road locations. Thinned out grass below trees could be textured in. Level design is an iterative, sculpting process and there is a great desire to be able to address textural concerns in the level editing process, not just structural. Imagine if Unreal Ed didn’t let you texture the walls and floors, but forced you to use the texturing baked onto them. Finally another reason for texturing in the editor is that morphing variants may not be able to guarantee topology that matches the textural treatment “embedded” in the tile. The thinned grass slope may turn into a level area that should have thick, healthy grass.

#### 8.1.9.2.3 *Pathing*

As mentioned above, tiles have a navigation mask that clues the path-finding algorithm, as to what parts of the tile can be traversed by what kinds of creatures. The mask defines the size of creatures that can pass, and their locomotion as well. So it details if a small creature can go to a certain spot on a tile, but also if only a flying creature could travel there, for instance.

What is underdetermined is whether pathing needs to be pre-computed. If it does, what is an appropriate area of tiles to logically path together? Given the streaming nature of the terrain tiles as well as the “random” generation of dungeons, do we feel we can pre-compute pathing?

In addition to the masks representing navigational limits, the level designer may want to define volumes that block various types of actors. These volumes may be designed to keep a group of creatures within a certain “home territory” or to spot-fix some level design anomalies, like to block off a rubble wall, cleanly, instead of relying on just the mask and/or physics proxies of the boulders.

#### 8.1.9.2.4 *Placeables*

A level is more than just tile pieces. Levels may have trees, bushes, sconces, statues, cages, projector shadows, and, of course, creatures among other things. These entities need to be placed in the level editor.

The interface for placing these can leverage much of the framework already needed for the tiles. You have a 3D browser showing the pieces. You can select one and then click in the level workspace to place it. Objects auto-plant on the Z axis over the tile they are over.

The mouse wheel could be used to rotate the object that is selected.

Shift-left dragging clones objects.

Because of the large quantity of potential placeables, the 3D browser may need to have either iconic or menu driven collections of placeables. You might have a pulldown where you select from the choices (such as monsters, lighting, dungeon set pieces, trees & foliage, etc...).

The editor will need to display the navigational collision volumes for these set pieces. This way the designer can see how much area an object will take and avoid inter-penetrations.

Placeables are included in Prefabs when they are saved out.

#### 8.1.9.2.5 *Triggers & Behaviors*



Triggers can perform simple actions from triggering the polygonal mask to come off of a room when a door is opened, to setting off animations on set pieces, to spawning creatures, sound effects, etc... Triggers are usually sphere volumes or line-defs. A visual, iconic and/or line-draw representation of these will be needed in the level workspace.

Other game logic functions that belong in the level editor include behavior setting for the monsters associated with an area. The level designer may want a group of monsters guarding a certain room to always run away when encountered to room X. These AI states need to be set in the level editor. The complexity of the AI system and the sophistication of the scripting system will determine how much the level editor can expose.

#### *8.1.9.2.6 Prefabs*

The concept of prefabs feeds into the randomization system. The idea is that assets, such as walls, have variants. Pedestals have variants. Statutes have variants. With the concept of prefabs, these assets can be swapped out with a consistent alternate throughout a level. If a level designer specified some pedestals in a room, these might be swapped out with a stone urn structure. This prefab idea applies to both tile pieces and set pieces.

To improve the functionality of prefabs, the Level-editor may need some additions. If you had a menu function to Select All Instances, where it would select all copies of the select object, you could then hit a menu option "Prefab Grouping", all instances of these tiles or set pieces would be swapped out for a variant by the random generator.

#### *8.1.9.2.7 Putting it All Together*

For interior spaces, the level designer is done with his first pass when he has created his tile-based environment representing the dungeon. Even with just walls and floors, the designer should be able to "walk" the level. The designer would next add lighting and set pieces. Next he would set the triggers and monsters. Once the designer has gone through all the steps mentioned, the level is saved out in its entirety as a .lvl file (or whatever extension we desire). At the very least this file needs to be a master asset, link, and entity list for the level, not unlike an XML file. The editor may need to save out. NIF files for any assets that have been modified by the editing process (such as by tile painting, morph variant setting, room polymesh optimizations, etc...).

It would be a requirement of the level I/O system that the level could be loaded back in and edited in the same state it was while being worked on.

## **8.2 Interior Base Tile Set**

The goal for the interior environment tile-set parameters is to develop a rule-set which can be versatile for creating interior zones both randomly and designer generated, with the feel of games like Diablo, Never Winter Nights and Dungeon Siege. Optimally these tiles would be reused for various interior types such as dungeons, crypts, caves, etc., by simply adding variants to the tiles and altering their textures. Dark crypts, deep dungeons, and hidden lairs will all be capable with our tile-set creation parameters.

The following describes a base tile set for creating an interior dungeon zone. These tiles are the minimum requirement for creating this type of environment.

### **8.2.1 Tile Properties**

Current tile settings are 5 x 5 x 5 meters with the ground surface 1 meter above the zero height fields, leaving 4 meters in height for walls, doorways, arches, etc.

Sub-level changes will be 1 meter in height, either above or below the floor surface. Some of these height changes will result in the creation of additional tiles to transition from these sub-level pieces. For instance, a shallow sewer channel running in the floor may need a special-case wall piece that joins the revetment and caps it off properly. Sub-level pieces have the potential to necessitate a lot of iterative pieces, so their use should be used judiciously.



Currently a tile can only have one walk able surface (in Z axis height) because of the Host's representation of the tile which is only 2D. In order for a character to be able to walk over or under each other they must be on separate "levels".

Props (crates, light sconces, barrels, etc) will generally populate the world independently from tiles. Some instances, however, may require some objects be part of the tile (A tree line to "wall off" an area from players etc...). Trees, barrels, crates, doors, etc, will be placed on the tile in the "editor" depending on their design needs and desires.

### 8.2.2 Edge specification

Tile edges will be uniform in the 0, 1 and 5 meter vertices edge (depending on the tiles transition set). Currently the normals of the edge vertices have to be the same as the tile it will be attached to. Optimally a small threshold similar to the edge inward will provide the best results and minimize the seams from tile to tile.

### 8.2.3 Abutment

Tiles with walls will be abutted to another to reduce the need for extra polygonal and textural properties for each tile, thus making wall thickness' 1 meter when abutted. The advantage for this is a reduction tiles needed for room to hallway transitions and gives further flexibility in interior environment generation. This is in comparison to similar games and gives us an advantage when random generation is concerned.

### 8.2.4 Tile Visibility Capping

Each tile will have a "cap" associated with itself. These caps will be flagged to be programmatically revealing the tiles below when a player gets closer to them. This is similar to most isometric RPG style games. Once an area is revealed the caps will not be used again.

### 8.2.5 Transparencies

At certain camera angles and game scenarios the players view can be obstructed by archways, walls, or even the level above. These items will be created or flagged in a fashion to allow them to "fade away", and return when the players' camera is no longer obstructed by them.

### 8.2.6 Naming Convention

Not explored at this time.

### 8.2.7 Textures

Not explored at this time.

Note - Geometry flipping is not supported in the engine therefore a tile would require a mirrored version of itself for instances that require it.

### 8.2.8 Tile Descriptions

#### 8.2.8.1 Blank / Floor Tile



An empty tile most likely a floor or other walk able surface which will be used to scale rooms, halls, etc.

#### 8.2.8.2 Corner Tile



Generic corner tile for rooms and hallways.

#### 8.2.8.3 Wall Tile



Generic wall tile. Variants may be windows, false doors, crumbled architecture, etc.



**8.2.8.4 Doorway/ Archway**



A typical doorway or archway for entering and exiting a room or hallway. Doors themselves will not be part of the tile.

**8.2.8.5 Wall Angle Joint**



A small angles piece of wall for creating convex corners and turns in walls. This tile can be replaced by a convex corner tile.

**8.2.8.6 Angled Wall**



Wall tile at an angle for possible creating circled rooms and angles niches in rooms and hallways. Depending on the angle may require other tile to interact to return to edge specification of tile set, e.g. round rooms.

**8.2.8.7 Stairs Ascend**



A generic set of stairs for level changes. The ascending tile will need specific wall pieces to maintain wall height through elevation change.

**8.2.8.8 Stairs Descend**



A set of descending stairs. The descending set will have to have the surface included in the tile geometry.

**8.2.8.9 Vertical Z Wall**



A tile used for tiling vertical wall heights. Used only when multiple levels are visible. These would most likely require a specific “edge tile” to begin this transition.

**8.2.9 Hallway Tile-Set**

The Hallway sub-set allows the smallest navigation paths with this current interior tile configuration. These tiles would also allow further customization of the larger interior tile-set.

**8.2.9.1 Hallway Walls**



Parallel walls for narrow hallways and corridors.

**8.2.9.2 Hallway Door**



Door or archway ending hallway.

**8.2.9.3 Hallway Corner**



Hallway corner tile.

**8.2.9.4 T Junction**



A tile for connecting and splitting hallways up.



#### 8.2.9.5 Four-Way



### 8.3 Exterior Base Tile Set

The goal for creating an exterior environment rule-set is to develop a tile-set which can achieve the “majesty” of a game like Dungeon Siege without equaling the number of tiles. Optimally these tiles would be reused for various terrain types such as snow, forest, desert etc., by simply adding variants to the tile-set and altering their textures. Towering waterfalls, Snow peaked mountains, swamp filled badlands, massive chasms, spanning bridges and dark forests, all capable with our tile-set creation parameters.

The following describes a base tile rule-set for creating an exterior terrain zones. These tiles are the minimum requirement for creating an outdoor environment.

#### 8.3.1 Tile Properties

Current tile settings are 10 x 10 x 5 meters with a polygon count of 2000 to 3000 polygons. This should provide enough surface coverage per tile visible to achieve desired performance levels, while still allowing enough geometry to create desired environment styles.

Sub-level changes will be 0 to 5 meter alterations in height, with a 2.5 meter plateau for edge specification. Some of these height changes will result in the creation of additional tiles to complete the transition and specific “design” needs. As well as possible additional sub-level changes for variations.

Currently a tile can only have one walkable surface (in Z axis height) because of the Host’s representation of the tile, which is only 2D. In order for a character to be able to walk over or under each other they must be on separate levels.

Props will populate the world on a tile by tile case, but some instances may require objects be part of the tile (a tree line to “wall off” an area from players etc...). Trees, barrels, crates, doors, etc, will be placed on the tile in the “editor” depending on their design needs and desires.

#### 8.3.2 Edge specification

Tile edges will be uniform in the 0, 1 and 5 meter vertices edge (depending on the tiles transition set). Currently the normals of the edge vertices have to be the same as the tile it will be attached to. Optimally a small threshold similar to the edge inward will provide the best results and minimize the seams from tile to tile.

#### 8.3.3 Naming Convention

Not explored at this time.

#### 8.3.4 Textures

Not explored at this time.

Note - Geometry flipping is not supported in the engine therefore a tile would require a mirrored version of itself for instances that require it.

### 8.4 Special Tiles And Pre-Fabs

Special tiles are created by an artist in 3D Studio max to describe a specific area or location like that is not represented in the base tile-set. The will conform to the tile edge specification but not be editable in the editor. The will act and interact like a normal tile but give the advantages of custom geometry in our world.

Special tiles are no more than a collection of tiles that have been configured in a particular fashion as desired by the designer in the editing program. Pre-Fabs are essentially a prearranged group of



tiles. The pre-fab is saved for multiple uses and faster level creation. After being placed in the editor the tiles can be rearranged and saved as yet another pre-fab.

#### **8.4.1 Advantages**

Pre-fabs are a collection of tiles prearranged in a particular order for fast level creation and multiple uses. Being able to quickly choose from a list of pre-fab rooms or areas will allow the designer to swiftly create a large dungeon by simply connecting pre-fabs together with the sub-set of tiles.

An advantage of special tiles is that while the edges still conform of to the edge specification rules, the interior is free to create different and unique areas from the tile-set its associated with, thus giving even more majesty to our world without increasing our tile-set.

#### **8.4.2 Disadvantages**

Certain types of special tiles will have to be created in 3D Studio Max, thus not allowing a designer to “mold” his or her particular vision of what that tile should represent.

If one object, and depending on its size, a special tile would require the rendering of polygons not necessarily visible to the player.

Possible texturing issues, not yet explored.

More tiles could be needed to make the transfer back to the normal tile-set from special tiles.

#### **8.4.3 Usage of special tiles / tile-pre-fabs**

Tile pre-fabs will be used to represent a prearranged group of tiles in a particular fashion, which they could reuse as often as necessary. Once a pre-fab is placed in the level, the editor places the tiles in their “pre-fab” order. The designer can now alter the tiles as they see fit.

Special tiles will be used similar to its smaller counterpart. They are place in the level by the designer and have the same functionality as a normal tile. When in the editor a special tile is not editable with the exception of being able to be rotated, just like a normal tile.

#### **8.4.4 Texturing**

Not explored at this time.

#### **8.4.5 Interaction of Special Tiles and Templates With Other Special Tiles and Templates**

Templates will be able to tile with themselves as long as the arranged tile in the template can logically be placed with the next template, otherwise tiles will be needed to complete the transition.

Special tiles are special cases. In order to describe a special area, multiple special tiles may be needed to complete the effect. Therein the special tiles may only work with the next special tile in the sequence, but the special tile edges must eventually return to the base tile-set specification for further tiling.

#### **8.4.6 Terrain Bitmap Masks**







#### **8.4.7 Navigation Bitmap Parameters**

A 60 x 60 pixel bitmap associated with its tile parent for A.I. pathing determination based on polygonal properties. Different colors flag a polygon for the type of pathing which can be associated with it. Note, any smaller in size causes pixel bleeding into another polygon.

(255,255,255) Passable







All characters and mobs can traverse with exception of water mobs.



 (0,0,0)	Impassable	No characters or mobs can traverse.
 (102,255,255)	Flyable	Only characters with flying abilities can travel.
 (255,0,0)	Small Mob Passable	Only small mobs can traverse.
 (255,204,51)	Large Mob Passable	Only large mobs can traverse.
 (51,153,255)	Shallow Water	All characters and water mobs can traverse.
 (0,0,255)	Deep Water	Water Mobs and flying mobs only.

#### 8.4.8 Material ID Bitmap Parameters

A 60 x 60 pixel bitmap associated with its tile parent for flagging material with their proper sound triggers based on applied materials. Different colors flag a material applied to the polygon for the type sound trigger which can be associated with it.

 (51,204,51)	Grass	Triggers a grass/grass-like sound.
 (102,102,102)	Stone	Triggers a stone sound. Rock surfaces, dungeon floors, etc...
 (204,204,102)	Dirt/Gravel	Triggers a dirt/gravel sound. Paths, river edges, etc...
 (153,102,0)	Wood	Triggers a wood sound. Floors, mostly.
 (0,51,255)	Water	Triggers a water sound. Rivers, lakes, puddles, etc...
 (0,0,153)	Metal	Triggers a metal sound. Grating, plating, etc...

#### 8.4.9 Bitmap Creation

Apply a UVW Map modifier with planar mapping and make a new map channel.

Using a Multi Sub-Object material with the appropriate parameters set, either the Navigation or Material settings or proper map channel, apply it to the tile.

Apply an edit mesh modifier to the tile next. Assign the appropriate ID's to the polygons necessary for which ever bitmap you intend to create, based on the bitmap parameters.

Next, in the Render to Texture Menu, turn off Unwrap textures. Choose your output path and file name and select the proper map channel. Add a diffuse map and make the size 60 x 60 and render.

With a new material, apply it to the tile. Be sure to have the material set to the correct map channel.



## 9 Armor Creation

### 9.1 Armor in RPGs

In a RPG, armor is a basic and necessary part of the loot system. Players need to be able to earn glorious looking armor that improves their appearance and their abilities. When we say armor, we are referring to pieces of equipment the character wears, such as chain mail coats, plate mail, leather tunics, etc.. Because of the importance of armor in the loot system, SF must offer enough variety in the armor offering to multiply out into a large potential set of combinations. This allows players to individualize their characters.

### 9.2 Core Issues

In the exploration, we attempted to address the core issues related to armor. These are:

- How granular can we get with armor pieces before we hit art pipeline, or performance issues?
- Will armor z-fight with the flesh model and how do we prevent it?
- How will we create armor sets across the different races and sexes?
- How much layering and overlapping of armor can we do?

The next sections will cover these issues.

### 9.3 Exploration Outcome Overview

Before we detail the answers above, let's summarize what was learned from the exploration.

- Surfaces per character must be strictly controlled as it will severely affect frame rate in situations where many characters are on the screen. Surfaces associated per character must be minimized. Towards this using Material ID flagging must be kept to cases when net surfaces in view will not be increased.
- Z-fighting is not an issue given our distance from camera.
- Steps must be taken to ensure skin-deforming flesh doesn't pop through armor.
- Flesh models can be externally referenced into armor MAX files to aid armor modeling and skin checking.
- Armor can be kept in a master MAX file and then externally referenced into separate files for creating levels of detail and racial modification.
- Bone partitions have a serious effect on frame rate when there are lots of creatures. Through use of hardware accelerated bone shaders, partitions can be vastly reduced. Our characters will need to leverage some form of hardware accelerated skin shader.
- Artists can flag faces with different material IDs and these resulting Gamebryo Tri-meshes can be parsed and turned off through code.

### 9.4 Armor Segments and Core Pieces

There are several reasons for limiting the granularity of armor.

- The more armor is segmented, the more tri-meshes visible per character. If a target goal is no more than 500 surfaces/frame; than having 100 characters on screen at once gives us a goal of 5 surfaces/character. More on this later.
- More pieces equals more art files to create levels of detail for, race/sex adjust, deliver and track. Since this multiplies the art load, it is good to segment armor judiciously.
- Having fewer armor segments reduces, but, of course, cannot avoid the mixing and matching between unpleasing combinations of armors.



With that said, here are the basic armor pieces:

- **Flesh** -this is not actually the armor, but the flesh model that is still visible when wearing the armor. Even a full suit of armor will still have some of the face showing, for instance. As we will detail below, the flesh model is customized to optimally fit each of the major armor styles.
- **Armor Suit** -this is mostly what you think of when you think “suit of armor”. It consists of the armor covering the chest, shoulders, and various amounts of the legs and arms depending on its style. More on the base armor styles later.
- **Gauntlets** -this is a loose term used to encompass any armor associated with the forearms and hands. The basic level of gauntlet covers the hand and the forearm just shy of the elbow. Some examples in this class include: gloves, plate mail gauntlets, bracers, and arm-guards. How much forearm the gauntlet covers can vary. The gauntlet surrounding the forearm can have “holes” in it as in an arm-guard situation where you have straps wrapping around to an armor pad.
- **Boots** - Boots cover the feet, on up to the knees. Boots may include knee-guards. Armor Suits do not.
- **Helmet** - this may include crowns, full-helms, tiaras, etc...



### 9.5 Layering

As we mentioned above, there are 3 main concerns with layering of armor: z-fighting, skin deformation discrepancies and wasted polygon fill. Of these, skin deformation misalignments are the most important and propel most of the layering strategy given below.

As a general rule, armor pieces only allow layering when a design requirement is placed on the layer beneath, that it be low in relief, and devoid of bulky details. The best example of this is the sleeves and leggings of the suit armor. If a character is wearing Chain mail leggings and sleeves, the gauntlets and boots need to fit over the chain. To do this, the chain mail needs to be low relief. From a construction standpoint, this “skin tight” armor will be derivative work based off of the flesh model so that the polygonal lay is as close as possible. The Gauntlets that layer over these sleeves and leggings must be thicker and higher in relief. So long as this requirement is met in the geometry, this layering is allowed.

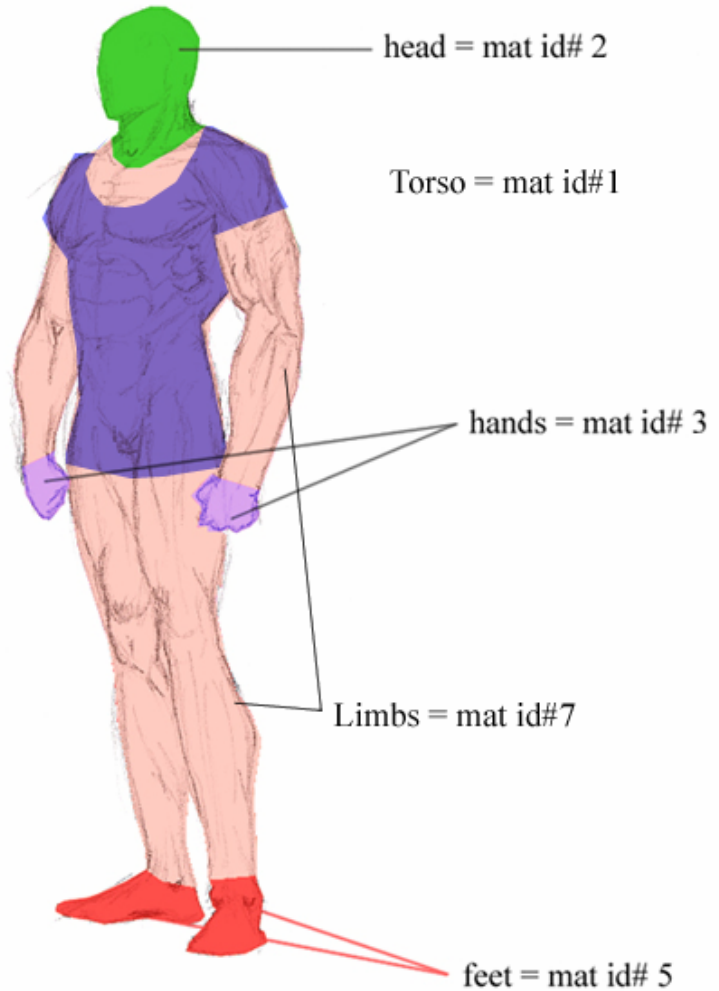
### 9.6 Tri-Mesh Culling

The hands and feet are areas of the flesh model that will be covered by armor 95% of the time. These areas also have high vertex density. These areas of the flesh model will be given materials IDs in MAX as in the diagram to the right. What this accomplishes is it forces a new trimesh in Gamebryo that can be turned off. The hands and feet in the high LoD represent about 650 of the 2500 polygons so culling these out offer a substantial savings. Since they will be culled 95% of the time, the number of surfaces visible will not be increased almost all of the time.



Similarly, the legs and arms are given a single ID so that these can be removed when the character wears a full suit armor style.

In the light suit style, only the torso ID flagged tri-mesh is hidden.



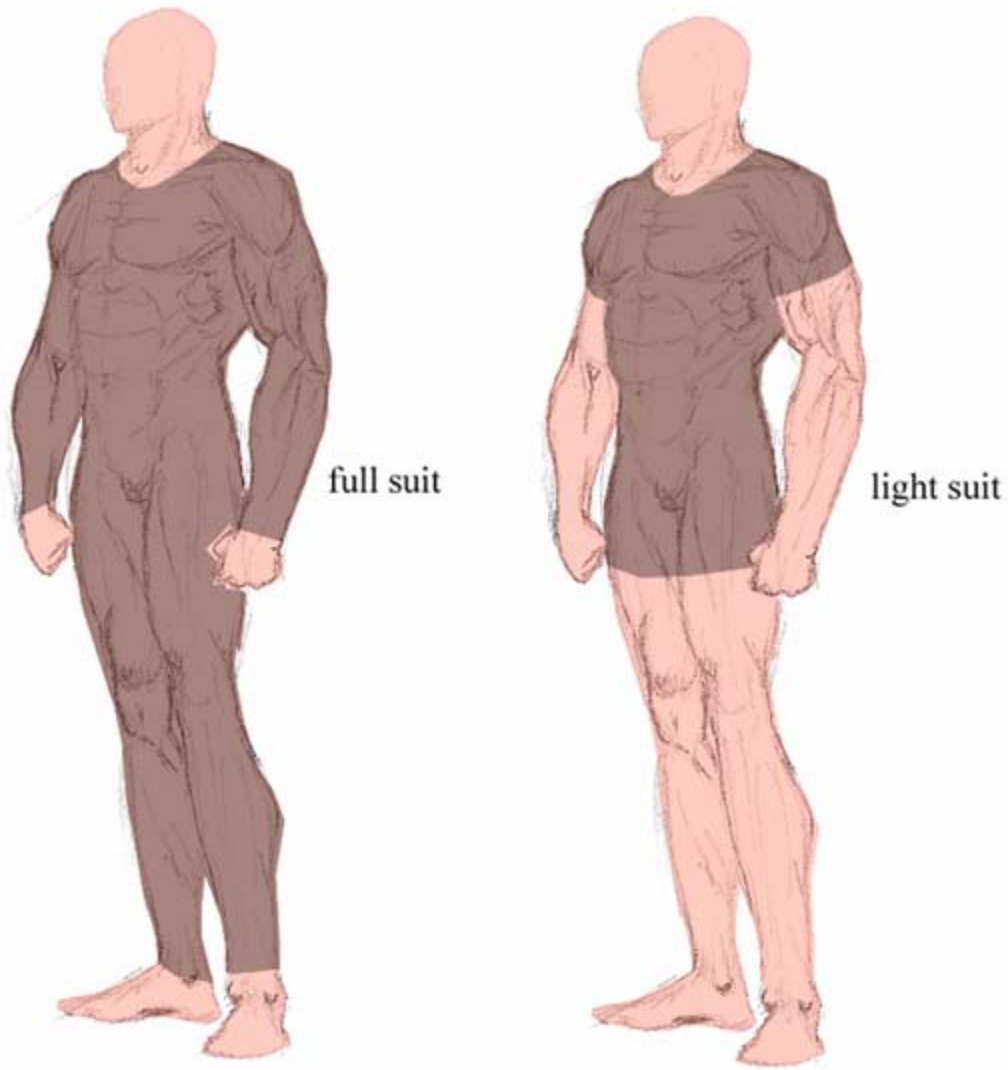
### 9.7 Armor Suit Styles

Because of the polygon fill waste of layering entire suits of armor over the flesh model of the character, a strategy was needed to reduce polygonal layering as much as possible.

To do this the polygons of the flesh model have two main material IDs, one for the torso and one for the limbs. These IDs correspond with the two armor styles shown below. The torso ID represents the polygons we want to hide when the character wears a light suit. When a character wears a full suit, both tri-meshes are hidden. This gives us four surfaces visible with the full suit, and five with the light suit. Previously we considered using a discrete model for these two polygon configurations, but since these would offer the save surfaces visible PLUS have a load hit, using polygon ID flagging looks to be the way to go.

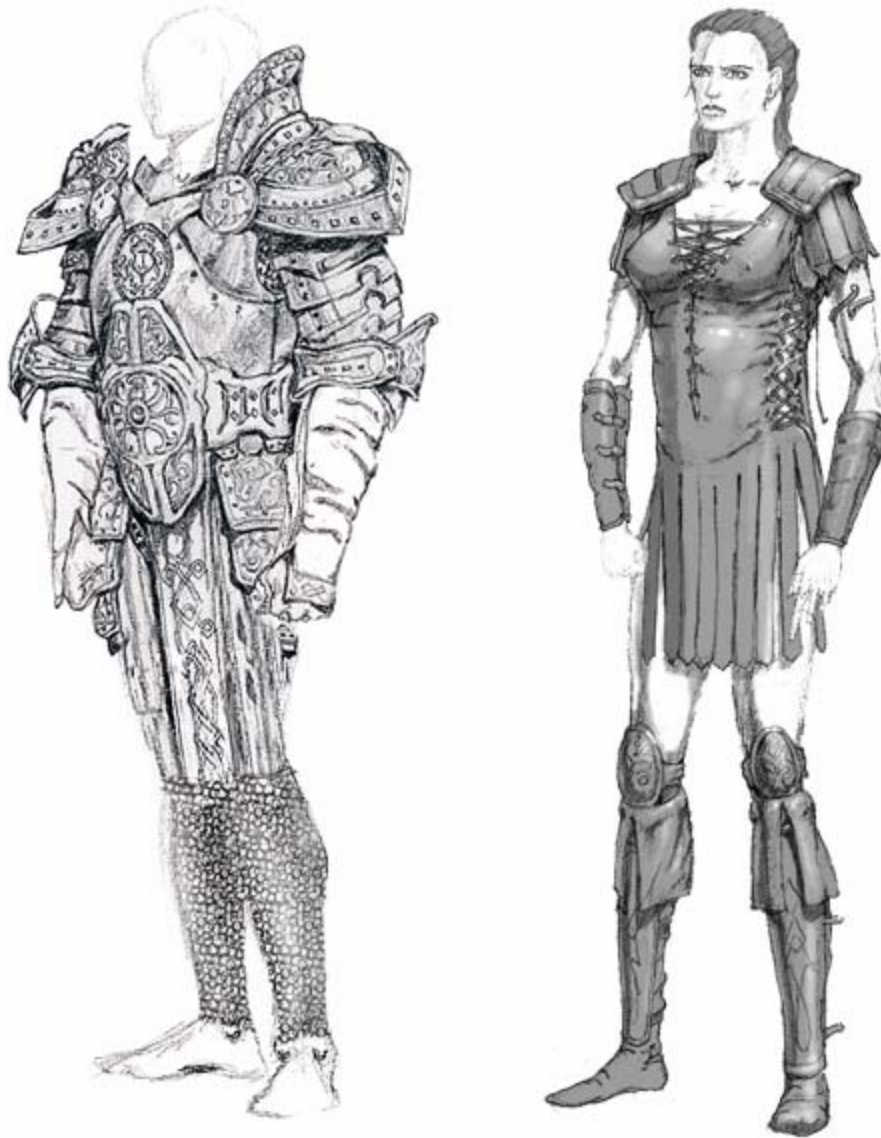


Here is a diagram of these armor suits:



As you can see, these armor suits offer various combinations of bare arms and leggings. Wizard robes, full-plate, and most armor will fall under the full-suit category. The light suit will be used for leather armor and other tunic-style armors without leggings or sleeves.

Here are some concept sketches that fall into these categories:



### **9.8 Armor Design Requirements**

Many requirements of the armors have been mentioned already. Here is a listing of them, plus others.

#### **9.8.1 Requirements**

- Armor suits must layer on top of any flesh model below them without the flesh model popping through when the skin deforms.
- Boots and gauntlets must be thick enough to layer on top of armor-suit leggings and sleeves respectively.
- Armor suits can be bulky above the elbow, but below the elbow must be “low-relief” to fit under gauntlets. This fairly well restricts armor sleeves and leggings to be skin-conforming cloth or chain.
- Elbow guards belong to the armor suit, not the gauntlets. They are part of the armor suit.
- Knee pads belong to the boots, not the suit.



### 9.8.2 Remaining issues:

- Concept of complete armor suits. This is a matching set of gauntlets, armor suit, boots, and helmet. Do the designers want this?
- Robes are full suits. What items can robe wearers use? Boots? Gauntlets? Helmets? Gauntlets propose the only problem in that gauntlets are bulky and may not fit under the sleeves of a wizard's robe.
- Capes. We want them. How do we animate them? How do they integrate with the armors without sticking out far from the shoulders as they do in, "Dark Age of Camelot"?
- How do armors fit other races and sexes?

### 9.9 Armor and the Art Pipeline

The following criteria should be adhered to when creating armor pieces.

- Armors have a poly budget that is proportional to the flesh model. A fully armored character should not exceed 4000 polygons in the highest level of detail. The next level of detail down is 2600 and then 1300 polygons.

Gauntlets                      Maximum of ~600 polygons per pair

Boots                              Maximum of ~600 polygons per pair

Helmet                            Maximum of ~300 polygons

Armor (Full suit)              Maximum of ~2500 polygons

Armor (Half suit)              Maximum of ~1500 polygons

- Armors must skin deform well with each other. It is suggested that once an armor model is created, that it be attached to the skinned flesh model so it thresholds to the skin's vertex weightings. The flesh model is then deleted. The armor is then tweaked.
- Whenever possible, it is suggested that the artist use the flesh model as a starting block for armor pieces so that the vertex lay be as similar as possible for ease in skinning.
- Armors get one surface each.
- Armors will be animation mapped from the flesh model so that we don't have to deliver animation sequences for the armors. This will require that the armors are not missing any bones found in the master animation file (unless we can code around this Gamebryo AnimTool expectation). Similarly, since we are just animation mapping, armor cannot have any new bones not found in the character files. If this gets to be a problem with Shoulder pads, then the master animation files will need to have shoulder bones and these will need to be animated in the master animation files.



## 10 Weapons



## 11 Items and Objects



## 12 Lighting Requirements



## 13 Particle Effects



## 14 Bestiary

