Do not forward this document.

Brent J. Zorich

www.brentzorich.com
brentzorich@yahoo.com

Meeting (#6): Best Practices Review

Date: February 5th 2009

Present: Bertand Ong, Jeffrey Lim, Tommy Burnette, John Sanders, Mark Cronin, Micheal Koperwas, Roberto Calvo, Parimal Aswani, Russell Paul, Jeffrey Odell, Dominic Hamon, Robert Clarke, Brent Zorich, Marlon Montgomery, Kate Shaw, Jayesh Dalal, Monique Bradshaw, Tauni Oxborrow, Abhilash Menon, Jayesh Dalal, Lee Stringer, Paul Zinnes
Apologies: Philip Schneider

Purpose: To follow up on the previous meeting discussing the Asset Sharing Best Practices/Review of notes.

Agenda:

1. Introduction:
    a. New addition and roles

2. Proposed Timeline:
    a. Asset Collaboration Timeline.vsd

3. Mailing List/Plone site w/ timeframe into Timeline

4. Expansion Planning Update

5. Asset/Model Best Practices Comments

6. Wrap-Up

Practices for Topology and Deformation
02.05.09

Best practices of topology according to deformation:

1. When doing deformation without enveloping sculpts, it is always best practice to try to keep one seam in the center of the joint, and two above and two below. The optimization practices of the mesh should be done in such a way that any alterations being done would still maintain the same silhouette while changing the topology.
2. Automation would be an ideal situation in this case. By incorporating techniques and practices of the other divisions we can work on convergence of Lucasfilm globally.
3. What could be an ideal solution for character generation would be the incorporation of block party into a set of standards. This is currently being ported from linux to windows and can be used for generation of assets for automation purposes. What could be great is if by using the volume guide in block party we could then transfer the mesh topology as a starting base as well as generate the rigs at the same time.

Building B Floor 6 21

Best Practices Comments:

Micheal K:

Changed some images to be a generic DD, and switched to a 1 unit = 1 foot scale. Also changed an image of hard edges on flat surfaces to be more explicit as a No-No.

Possible areas of question:

- Should we discuss directory structure? i thought about removing it, but think it's worth keeping in if we can standardize that moving forward.

- ILM tradition is X-forward, a horrible carryover from Softimage days. Not something that's likely going to change. Could be scripted based on who is checking something out, but it's a weird one, i know. (see *)

- image file format in the file is PSD and Targa. For ILM it's PSD and tif; we don't use tga. That's also scriptable based on who is checking something out.

- Could the shading network requirements be reduced in the document?
i separated the docs into one for Model sharing, and one for Texture sharing.

*** Tommy Burdette:**

- Drop that old X-forward ILM/soft convention and deal with making ILM handle it properly later. Scripting a transformation like that would be trivial and could added any number of places along the pipeline so we shouldn't worry about it too much at this stage.

Philip Schneider:

- Being a representative of ILM R&D at these meetings, I have only an engineer's perspective on "best practices", rather than a modeler's. Probably much of what follows will overlap considerably the input from Michael Koperwas and Russell Paul.

  My bias is towards what works with Catmull-Clark surfaces, as that is what we use at ILM for models, pretty much exclusively. Michael and Russell can verify this, or perhaps point out the exceptions. However, from my experience the best practices for Catmull-Clark also apply to polygonal models that might be used in today's games.

- Topological Issues:

Manifold Models:

All models should be manifold and be compatible with Catmull-Clark subdivision. In practical terms this means:

- All edges must border exactly one or two faces
- No vertices may exist that are not part of any edge or face
- No two vertices may share more than a single edge.
- No "bow ties"
- No lamina faces (faces sharing more than a single edge with another face)

Faces:

Quadrilateral faces should make up the vast majority of any model. Triangles may be used where necessary, as well as faces with five or six sides; faces with a large number of vertices should be avoided.

Vertices:

Vertices with valence 4 (that is, have 4 edges emanating from them) should make up the vast majority of any model (boundary vertices should have valence 3, except of course at corners where the valence is 2). The reason for this is that a Catmull-Clark surface at and between valence-4 vertices is smooth in the same way a bicubic B-Spline surface is smooth - no rendering artifacts will be present; however, at non-valence-4 vertices (aka "extraordinary vertices"), the surface is less smooth, and this reduced smoothness can be seen as a rendering artifact. Extraordinary points are best located on a model at points where the reduced smoothness will be less apparent, if at all possible. Modelers experienced with constructing Catmull-Clark mesh models can provide their input on this issue. Frequently, extraordinary points are located at, say, corners of the eye or mouth, or within an intentional crease, etc. Vertices with valence greater than 5 or 6 should be avoided at all costs.

Geometric Issues:

- Faces of models should not have an extreme length/width ratio.
- Degenerate faces should never be present, and nearly degenerate faces should be avoided at all costs.
- Vertex locations should be given in double precision wherever possible.
- Face size can vary widely across a model, but should not vary rapidly in a local sense.
- "Hard" edges (in Catmull-Clark terms) should be avoided in favor of beveling.
- "Hard" vertices (again in Catmull-Clark terms) should really never be

used.
- It should be remembered that predefined vertex normals are not meaningful for Catmull-Clark surfaces.

Brent Zorich:

- Best practices of topology according to deformation:

1. When doing deformation without enveloping sculpts, it is always best practice to try to keep one seam in the center of the joint, and two above and two below. The optimization practices of the mesh should be done in such a way that any alterations being done would still maintain the same silhouette while changing the topology.
2. Automation would be an ideal situation in this case. By incorporating techniques and practices of the other divisions we can work on convergence of Lucasfilm globally.
3. What could be an ideal solution for character generation would be the incorporation of block party into a set of standards. This is currently being ported from linux to windows and can be used for generation of assets for automation purposes. What could be great is if by using the volume guide in block party we could then transfer the mesh topology as a starting base as well as generate the rigs at the same time.

Modeling Guidelines

Beveled Edges

Nothing in real life has a razor sharp edge (with the exception of razors), and neither should objects in the game. Additionally, an object with a more rounded edge catches highlights more than objects with sharp edges. The images below illustrate this difference:

Cube, no beveled edges

Cube, beveled edges.

In the above example, two cubes, one beveled and one not, are rendered with the same materials and in the same environment. Notice how the highlights are picked up from the lights in the second image, as compared to the flat look of the first.

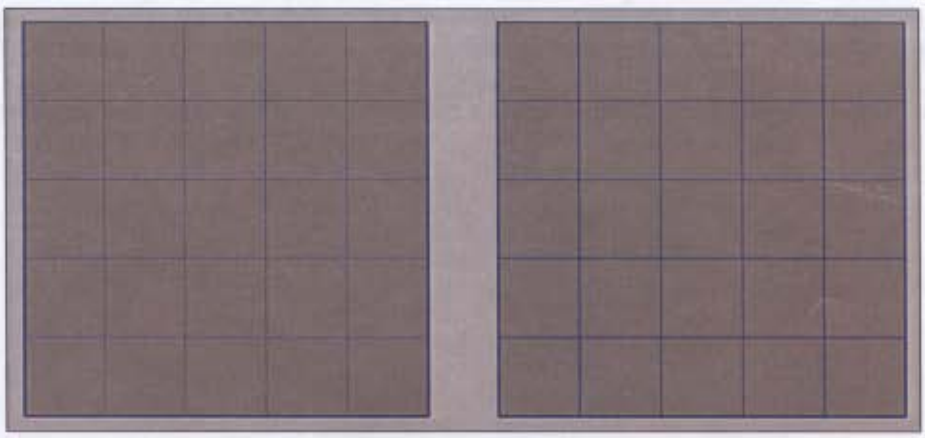
On the Indy project, it recommended that major edges be beveled whenever possible. Doing so will greatly enhance the look of the environment in-game. Knowing which edges to bevel is a subjective decision and consideration should be given to how large the bevel (in units) is and on how large of a surface it is applied.

Rules:

- Bevel when and where it makes sense, but as much as possible. Try not to bevel in areas which will be less than a pixel onscreen (like vista views, etc), otherwise it can be a waste of vertices.

Hard/Soft Edges

Only use hard edges when necessary. During export, any vertices that connect to a hard edge will be duplicated, thus increase to the total vertex count in run-time. This is done for hardware reasons.



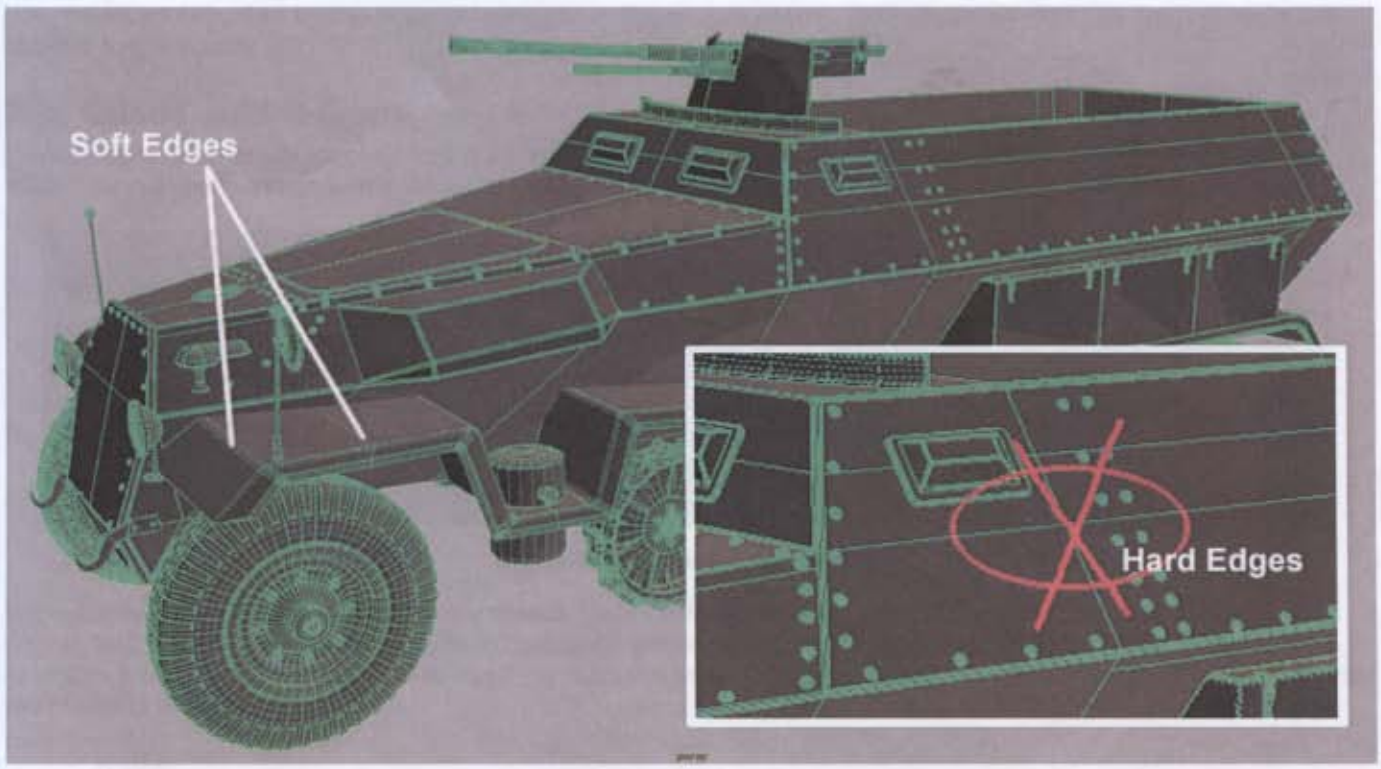
Good (left) and Bad (right) uses of edge hardness

In the example above, all inside edges are **soft** in the left image. On the right, all edges are set to hard. In the game, the vertex count in the model on the right **would be almost twice (2x) as much** as the model on the left.

Please refer to the ***Polygon Display*** (p. 4) of this document for more information about the standard Polygon Display settings for Maya.

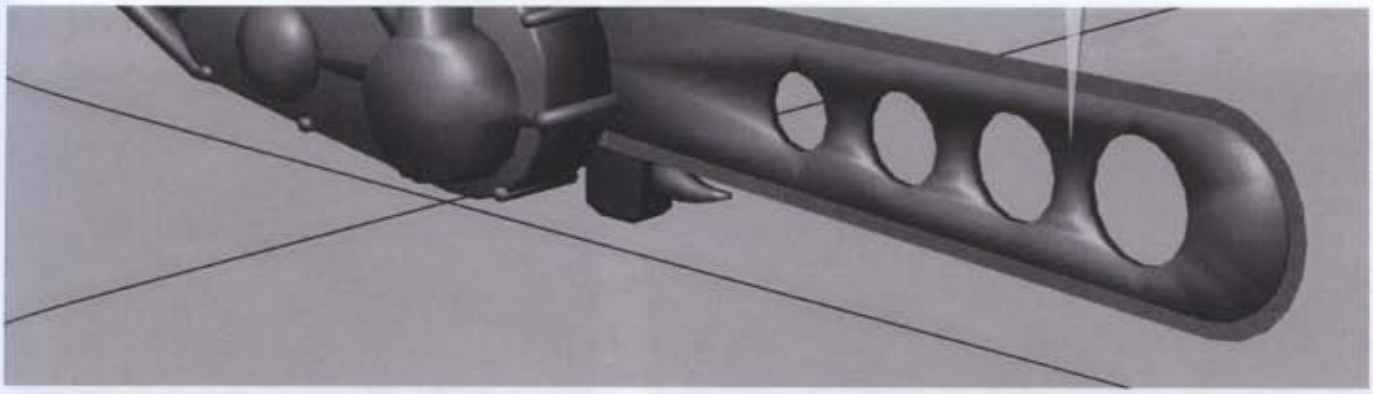
Rules:

- Only use hard edges when needed.
- **NEVER** use hard edges across flat surfaces



In the example above, the vehicle is **NOT properly making use of hard and soft edges**. The hard edges in the red circle all shared faces along the same plane. Since the relative angle between the faces is 0 degrees or less, having the edges hard creates unnecessary vertices on export. To correct this, simply select the edges in Maya, and make them soft. If the viewport settings are properly setup, they will switch from a solid line, to a dotted line

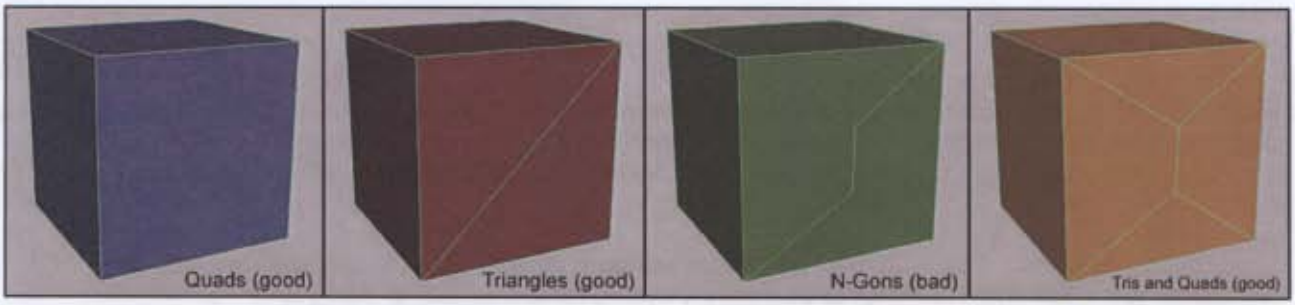
On non-coplanar surfaces, removing shading errors from a surface can be tricky and in some cases, these errors will show up when rendered in the game. For example:



Note the faces indicated by the white arrow head in the picture above. Setting edge weighting appropriately can alleviate these trouble areas.

Tris, Quads, and N-Gons

Maintaining a good mesh topology not only reduces rendering artifacts, but it also produces a more optimized mesh during export. Whenever possible try to connect n-gons to other faces to produce triangles and quads.



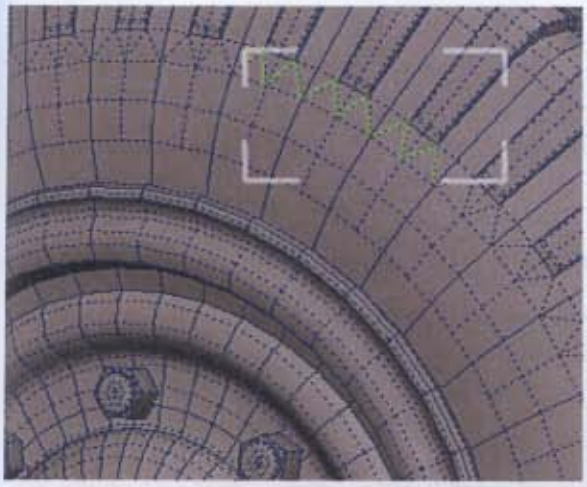
Examples of quads, and tris, n-gons

In the above example, the boxes display **quads** (faces with 4 verts or edges), **triangles** (faces with 3 verts or edges), and **n-gons** (faces that have less than 3, or greater than 4 verts or edges). The last image on the right illustrates an acceptable correction to the n-gons on the previous image. Note how the n-gon faces were split into two triangles and two quads.

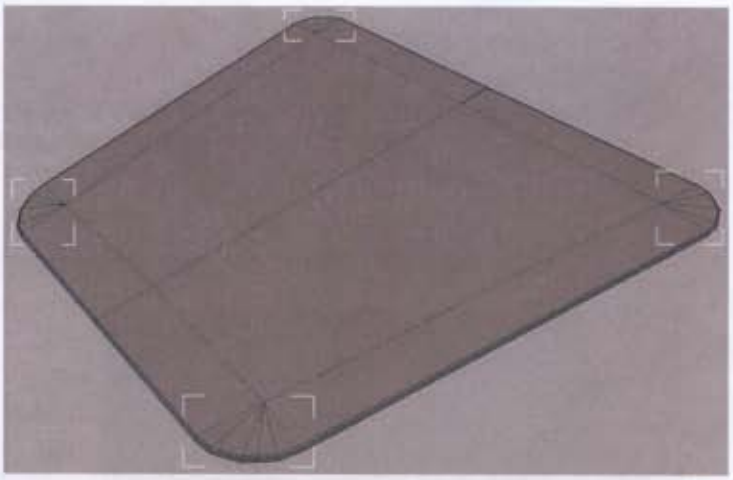
Rules:

- Quads are the most preferable way of modeling a mesh. Only use triangles when they are necessary (rounding corners, eliminating n-gons, etc)
- No n-gons (polygons with more than 4 vertices) are allowed.

Examples of good triangle usage:



Good use of triangles to join N-gons in a mesh



Good use of triangles to round off corners and edges

UV Layout

A good UV layout serves two purposes: to allow better quality from the assigned texture and to help cut down on the overhead memory cost. If a UV layout is inefficient, the final quality suffers, and the object generally uses more memory at runtime. When creating UVs for an asset, it is important to follow these guidelines:

Rules:

- Minimize number of UV islands as much as possible.
  - This helps to reduce overall memory overhead at runtime. However, if this results in stretched or skewed UVs that affect the appearance of the texture on a surface, then it is appropriate to detach UV islands to alleviate this.
- Overlapping of UV islands is an acceptable method of maximizing texture space and reusing parts of a texture UNLESS the Asset Spec indicates that the UVs should be setup for Light Maps. Going outside the 0 to 1 space is an acceptable method of setting up a UV island to tile.
- The asset spec may call for more than one UV set to be created. ONLY create the number of UV sets specified in the asset spec.
  - This is a direct impact on the overall memory footprint of the object and MUST be followed at all times.
- Unless specified in the UV Notes section of the asset spec, the names of the UV sets do not matter.

U and V defined

U is defined as Width and V is defined as Height. This is important to consider when referring to the Asset Spec, as it will identify the appropriate texture map size in both dimensions. As a rule, non-square textures should be oriented vertically rather than horizontally as this is more efficient for our renderer.

Non-Overlapping UVs

Non-overlapping UVs (UVs in 0 -1 space, with no UVs overlapping others - see image below) are commonly used for any type of textures that require atlasing, or any textures containing baked lighting information (lightmaps,

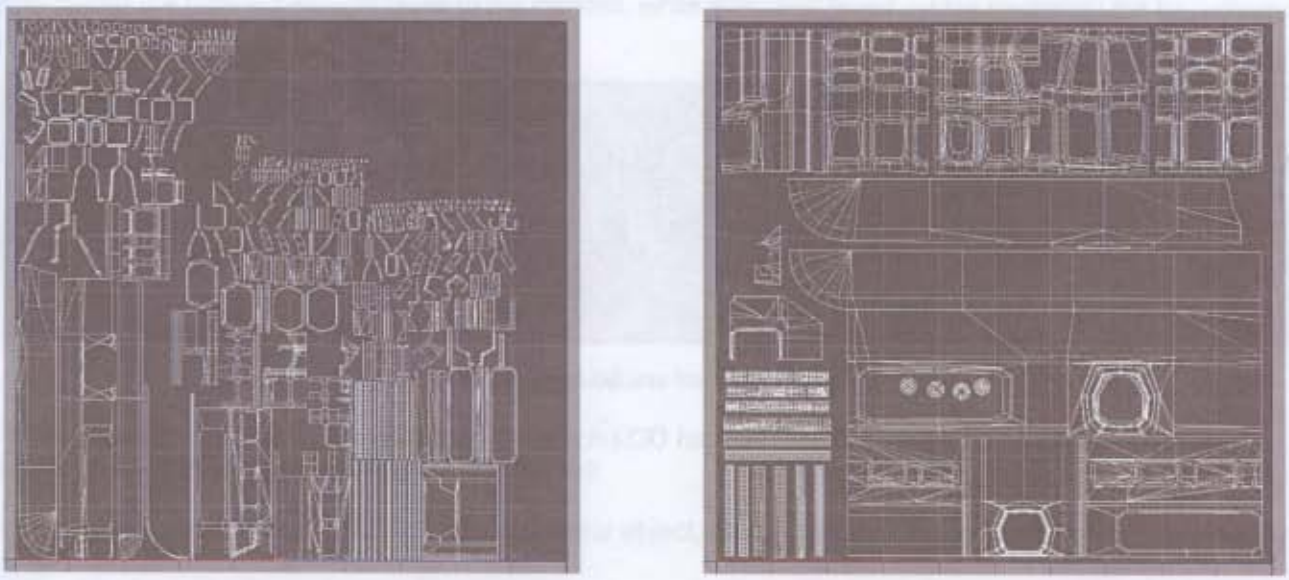
Ambient Occlusion, normal maps, etc.). If a particular UV set needs non-overlapping UVs, it will be called out in the Asset Spec.

UV Map Channels

The color map applied to an asset should be installed on the default UV map channel. Any subsequent UV channels used, as perscribed in the Assets Specifications, may be added after the default channel.

UV Layout – Best Practices

The images below illustrate not only good and bad UV layouts, but also non-overlapping UVs.



Example of a bad UV layout (left) vs. a good UV layout (right)

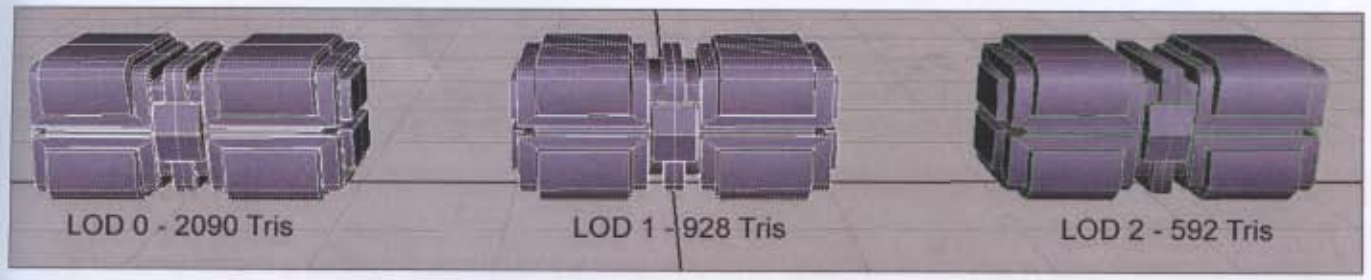
In the above example, the image on the left was generated using Maya's automatic mapping algorithm. Notice how many discreet UV islands there are, as well as the amount of wasted space in the layout. In the image on the right, the UVs were laid out by hand, maximizing the visual parts of the model, and making as much use of the UV space as possible.

Example illustrating visual artifacts of poor UV layouts vs. good UV layouts.

In the above example, take note to the seams and artifacts present in the image on the left. While the image is a different object then in the previous example, it still illustrates how poor UV layouts can create disjointed seams when a map is applied to the model. In the image on the right, there are no visible seams, as the UV layout was corrected to optimize UV islands and to capture the most amount of lighting detail.

Levels of Detail (LODs)

When drawing objects at far distances, it is often necessary to swap out the higher resolutions for models, and replace them with simpler representations. This process, called Levels of Detail, allow the renderer breathing room to render the high-res objects close to the camera, while spending fewer cycles rendering the background objects.



Example of good LOD use from Star Wars: TFU

In the example above, note that even though the each LOD has roughly half the triangles as the previous, the general form and silhouette of the object is preserved.

When modeling Levels of Detail (LODs) for a particular object, please observe the following guidelines whenever possible:

Rules:

- In order to achieve the maximum payoff between LODs, the vertex count between each LOD should be reduced by around 50%.
    - For example, the highest LOD for an object may be 100 verts. The next would be 50, and so on.

- If possible, reduce the number of materials between each LOD. Huge performance gains can be made by simply dropping the number of materials on the object by 1.

The maximum number of assigned materials for an objects LOD will be specified in the Asset Spec.

Texturing Guidelines

Texture Files

All textures should be saved as a **PSD** (default Photoshop format) in order to preserve the individual layers that make up the composite texture. Do not flatten **PSD** files.

Texture files should also be saved out as 72 dpi **TGA** files at 24 bit, unless and alpha channel is used. If an alpha channel is used, the file should be saved at 32bit.

Alpha Channels

Alpha channels may be used in textures only if specified in the Asset Specifications. Alpha can be used for either transparency/translucency, or as a method of storing other map types, such as Specular, or Self Illumination.

Texture Tiling

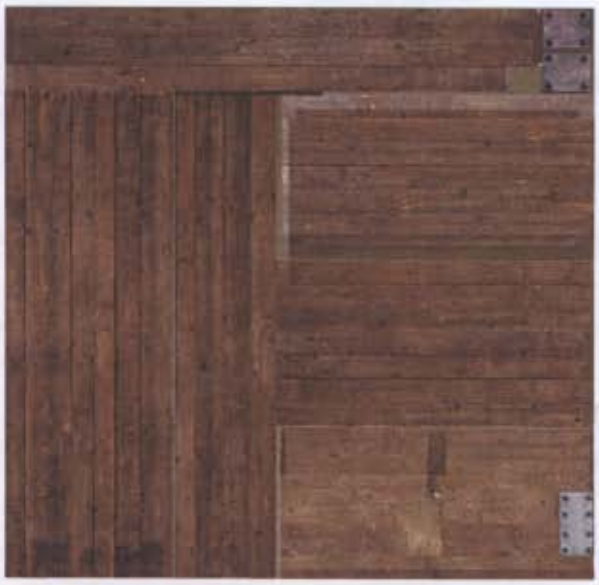
Tiling of textures across a surface is acceptable unless otherwise specified in the Asset Specifications. Having UVs outside the 0 to 1 space (when tiling a texture) in the UV Editor is also acceptable, unless otherwise specified.

Color Maps

When painting color textures, please follow these rules to get the most out of the texture:

- All textures must be level-balanced and checked under a neutral lighting rig (see section on Color Balancing for a how-to description).
- Any and all lighting should be removed from a color texture. A color texture should only describe a surface's "pure" (or albedo) color (see below images).

Bad Images	Good Images



- Ambient Occlusion should **NOT** be baked directly into the color texture at any time. **If ambient occlusion is required, it will be called out in the Asset Spec and will be delivered as a separate texture.**
- In addition to the source PSD with layers, a flattened TGA file must also be delivered.

Normal Maps

When creating normal maps for an asset, please observe the following rules:

- Normal maps should always be generated in **tangent space**, unless otherwise specified in the Asset Spec.
- Unless specified in the Asset Spec, normal maps will not need a PSD file.
    - Most normal map generators output the normal map directly in to TGA format.
- The Asset Spec will indicate if the normal map should come from a high-resolution source mesh. Unless otherwise indicated, Normal Maps can and should be derived form 2D textures rather than from high-res geometry.
- High Res source should be derived from either ZBrush or Mudbox.

- Low Res maps can be generated from Photoshop using nVidia's Normal Map filter, or via CrazyBump, which gives even better results.

Materials

The textures painted for a particular model must be assigned to a material, which in turn, must be assigned to the geometry, before submission. The number of materials and textures required will vary, depending on the specifications called out in the Asset Spec.

Rules:

- ONLY create and assign the specified number of materials to the object(s).This number can be found in the Asset Spec.
- The number of assigned materials for an object has a direct effect on overall game performance. It is EXTREMELY important that this rule is ALWAYS observed.
- In order to help ease the process of getting the asset into the game engine, only choose materials from the list of supplied, acceptable materials. The following materials should be included as part of this packet:
  - colorSpecNorm_network.ma
  - colorSpec_network.ma
  - colorNorm_network.ma
  - color_network.ma

If an occlusion map is required for an asset, please hook it into the material using the Adding Occlusion tutorial at the end of this section.

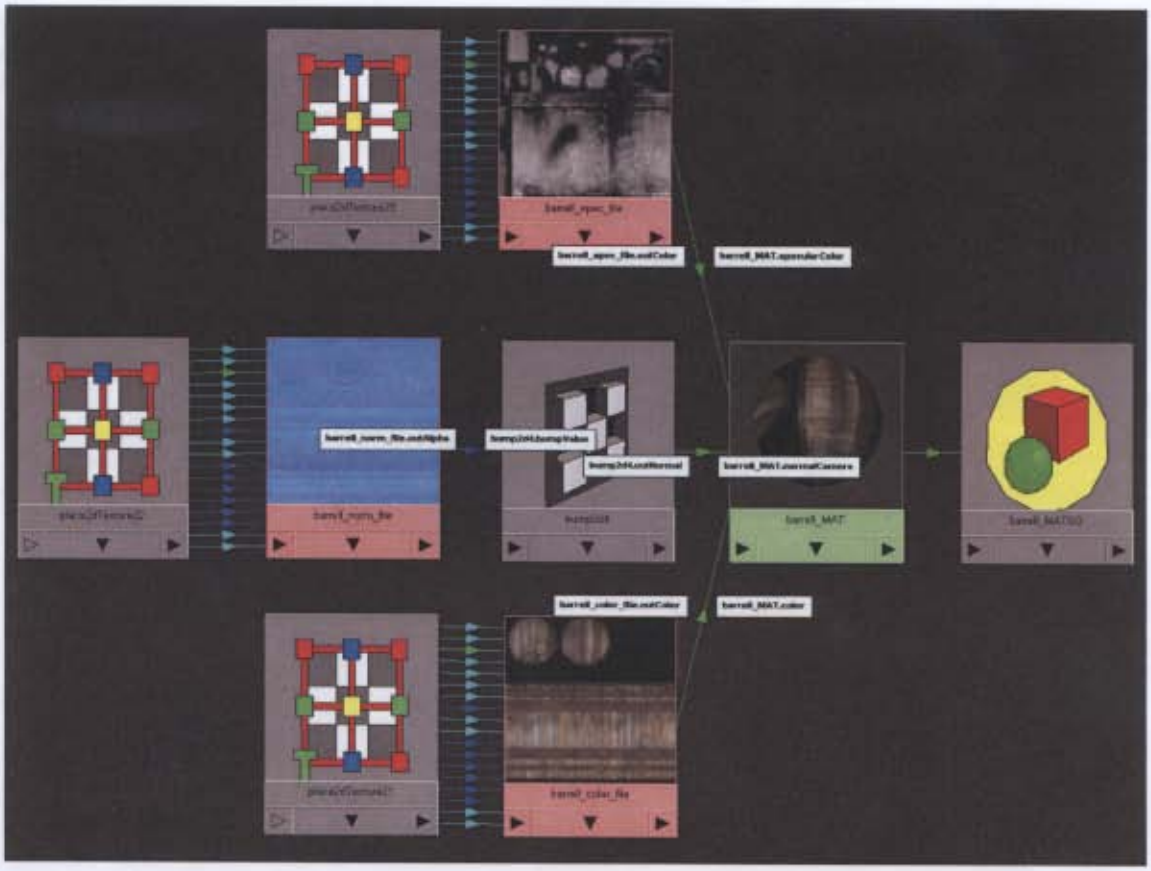
Materials/Shading Networks Explained

Below is an explanation of the acceptable materials, and how the individual nodes inside connect to each other for the final result.

colorSpecNorm_network.ma

This network consists of the following:

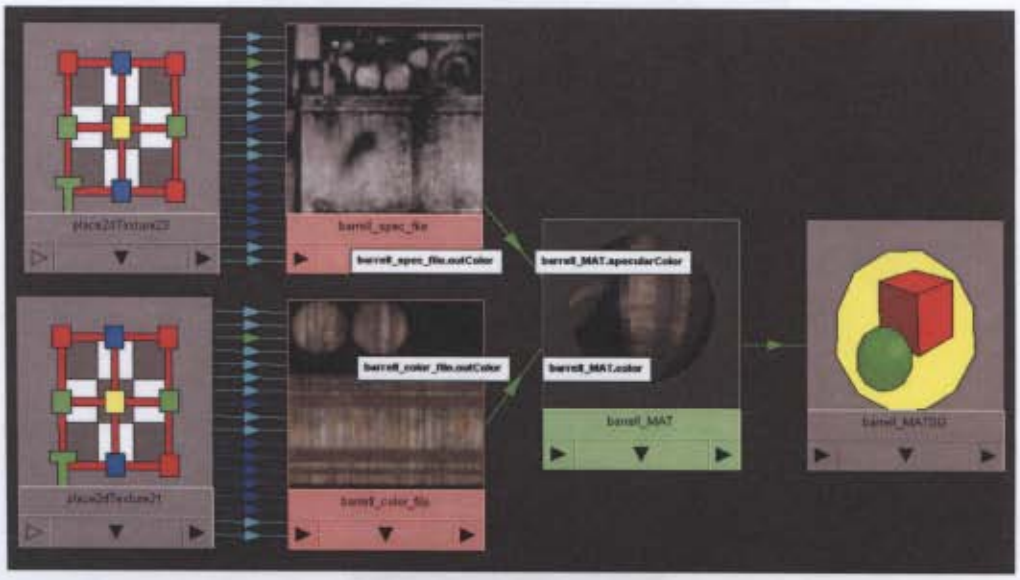
- Blinn Shader
- File node connected to the color attribute of the Blinn
- File node connected to the specularColor attribute of the Blinn
- File node connected to a Bump2d node.
- The Bump2d nodes outNormal attribute is connected to the Blinn nodes normalCamera attribute
- The Use As value on the Bump2d node should be set to Tangent Space Normals



colorSpec_network.ma

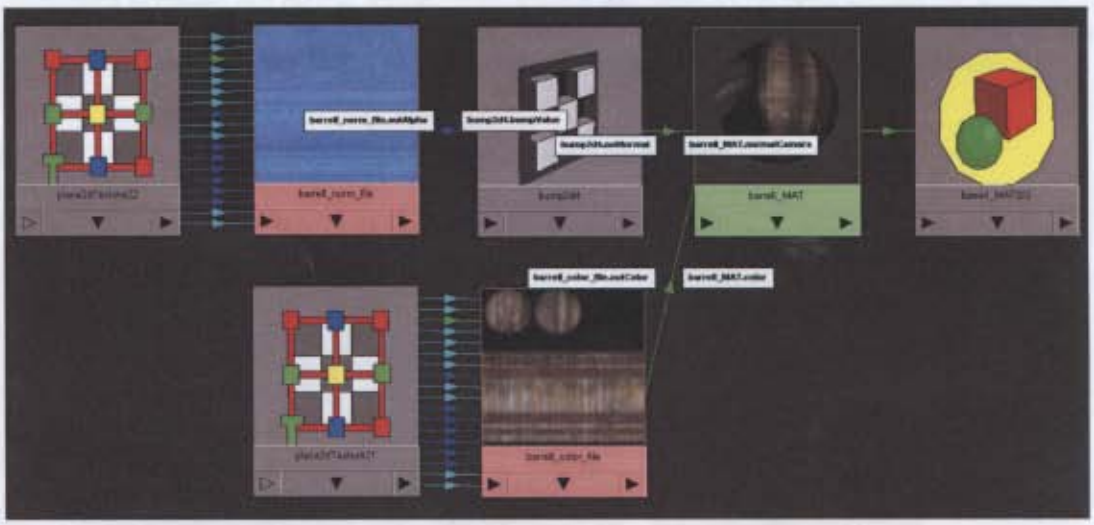
This network consists of the following:

- A Blinn Shader
- A File node connected to the color attribute of the Blinn
- A File node connected to the specularColor attribute of the Blinn



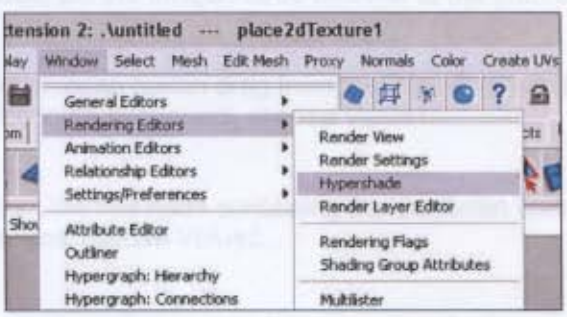
colorNorm_network.ma
This network contains the following nodes:

- A Blinn Shader
- A File node connected to the color attribute of the Blinn
- A File node connected to a Bump2d node.
- The Bump2d nodes outNormal attribute is connected to the Blinn nodes normalCamera attribute
- The Use As value on the Bump2d node should be set to Tangent Space Normals

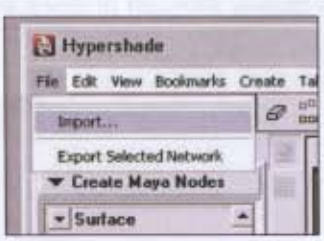


Tutorial: Importing Materials/Shading Networks

Step 1:
In Maya, on the main menu bar, click on Window > Rendering Editors > Hypershade



Step 2:
In the Hypershade window, click in File > Import...

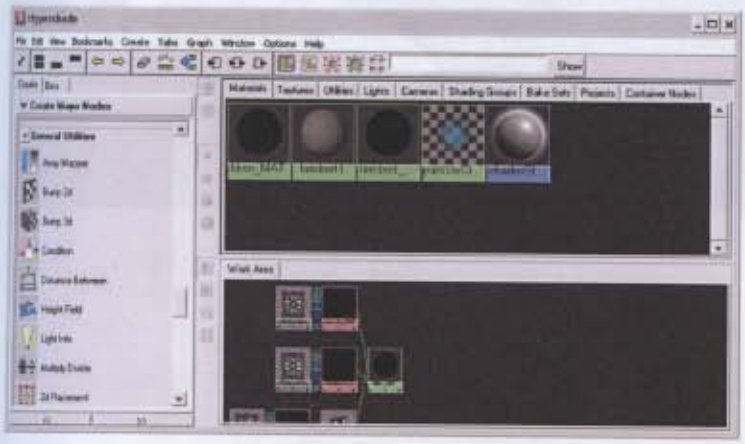


Step 3:
In the Import dialog, choose the appropriate material file (*_network.ma) and press the Import button. The shading network should now be imported into the scene and ready for use.

Tutorial: Adding Occlusion to a Shading Network

If an occlusion texture is called out by the Asset Spec, then it will need to be hooked into the existing shaders. This is a fairly easy process, as the occlusion texture is simply a multiply operation on top of the color texture. Multiplying the occlusion by the color in the shading network can be done with the following steps:

Step 1:
In the hypershade Create > General Utilities panel, create a new Multply Divide node.



Step 2:
In the hypershade Create > 2d Textures panel, create a new File node.

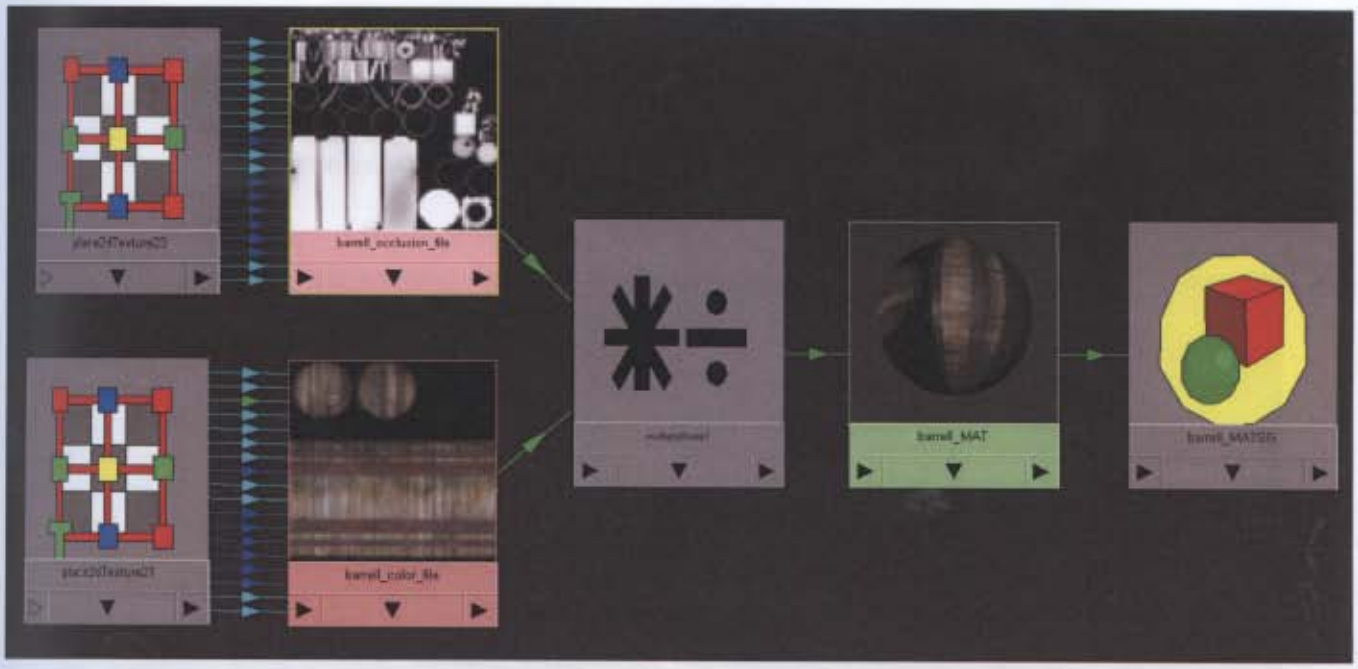
Step 3:
In the attributes for the new File node, set the Image Name channel to the path of the occlusion texture on disk.

Step 4:
In the hypershade work area, middle mouse button drag from the File node with the color texture to the Multiply Divide node. When prompted which input to connect, choose Value1.

Step 5:
Next, middle mouse button drag from the File node containing the occlusion texture, to the Multiply Divide node. When prompted which input to connect, choose Value2.

Step 6:
Finally middle mouse button drag from the Multiply Divide node to the Blinn material node.When prompted which input to connect, choose Color.

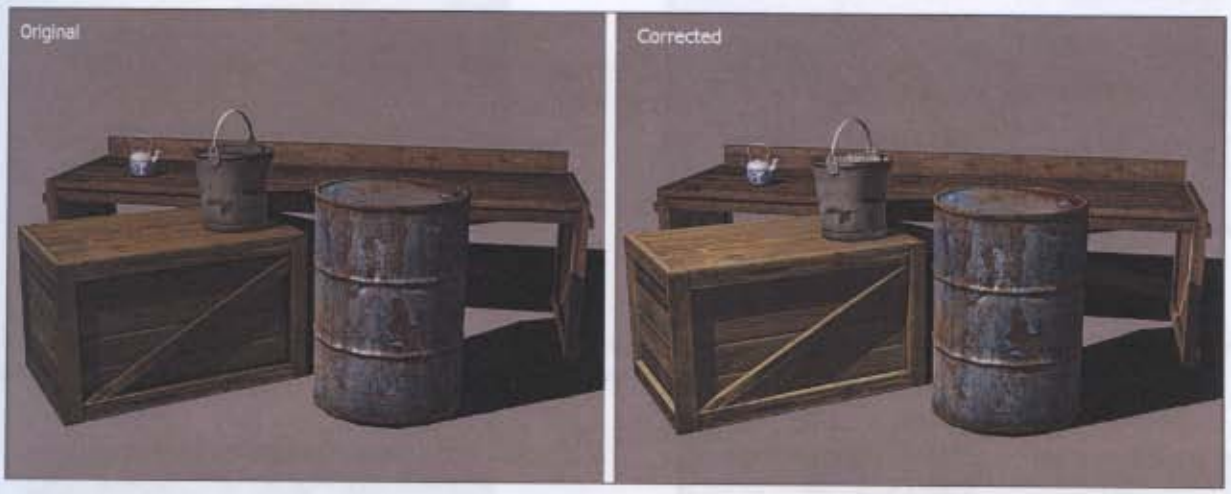
When all the nodes are properly connect, the shading network should look similar to this:


barrel_occlusion_file
barrel_color_file
barrel_MAT

Tutorial: Color Balancing

Every color texture required by the Asset Spec must be properly color balanced to ensure that all textures fit within same color range. As a result, lights will be able to hit both ends of the spectrum for the highs and lows. In turn, this makes it much easier to visually sit different objects from different artists into the same scene.

Example:

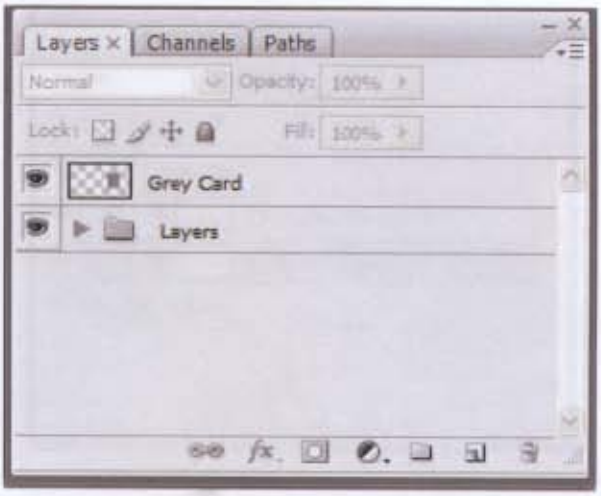


Example of non-balanced vs. balanced scene

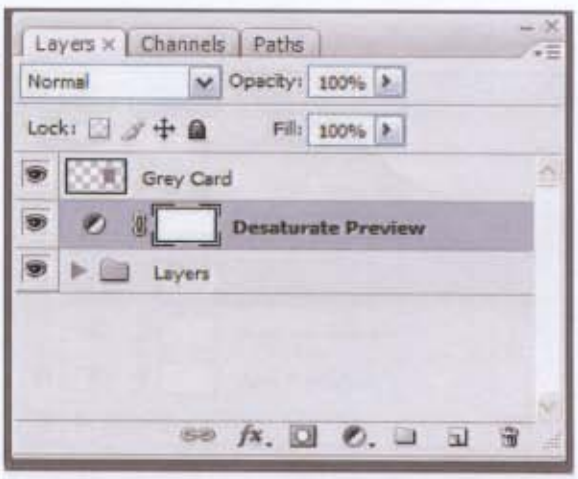
In the above example, the differences in the two scenes are subtle, but important. In the image on the left, none of the textures were balanced properly against each other, resulting in a dark, muddy mess. In the image on the right, values and hues were adjusted to help ensure the difference objects sit better together in the final scene.

Original painted texture

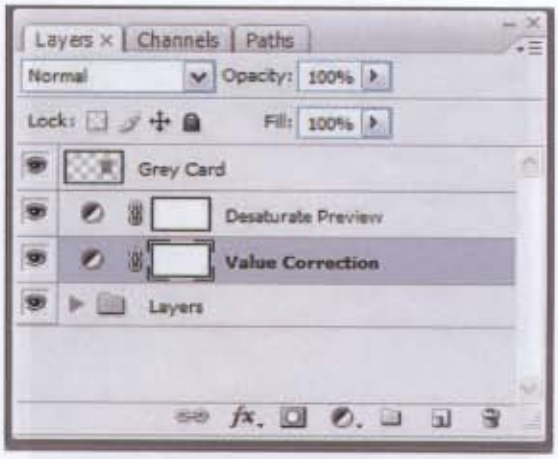
Step 1:
Create a 50% grey card (RGB: 128,128,128) on a new layer in the PSD. Call this layer "Grey Card".



Step 2:
Next add a Hue/Saturation adjustment layer below the grey layer and set the saturation to -100. Call this layer "Desaturate Preview". This allows the ability to quickly toggle between a color and grayscale image.



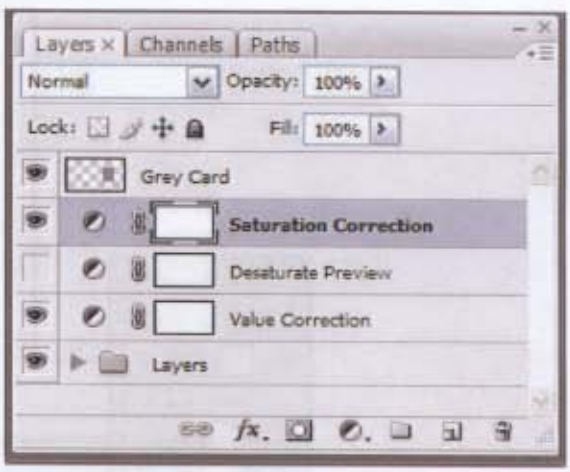
Step 3:
Create a new Curves adjustment layer called "Value Correction". Tweak the settings of the value correction layer to match the grey card as close as possible.



When tweaking the settings of the value, the goal is to get the mid-tones of the image to roughly match the value of the grey card. Refer to the above illustration.

- Generally, the blackest black of the image should be about 5-10% **brighter** than perfect black.
- Likewise, the whitest white of the image should be about 5-10% **darker** than pure white

Step 4:
Create a new Hue/Saturation called "Saturation Correction".

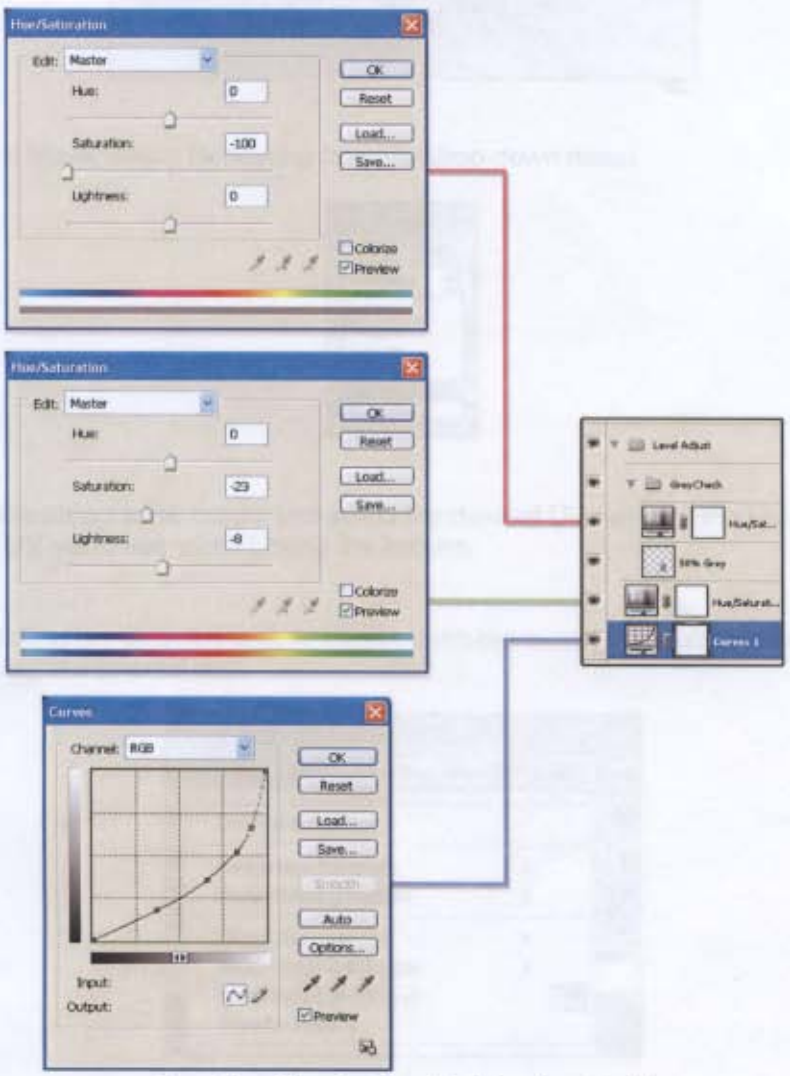


When a photo is taken of a subject, the color of the light illuminating the object (from the sun, the flash, etc) is already being added into the base (or albedo) color of the subject. The goal of this step is to eliminate as much of the color from the light source as possible. The end result allows the color of the in-game lights to more accurately illuminate the object at runtime.

Final Result
The images below illustrate a before and after of the example texture, as well as the settings used to get to the final result.

NOTE: Settings will vary depending on each texture.

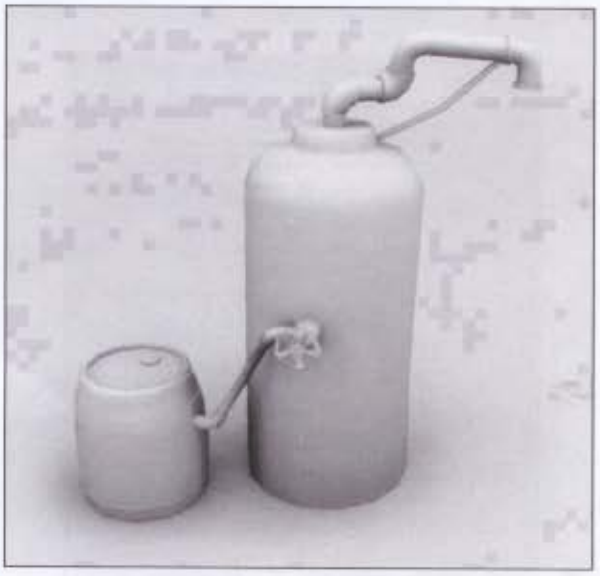
Before and after comparisons of the color image.



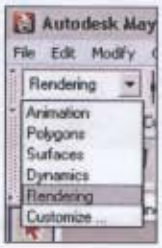
Example settings for the color balancing tutorial.

Tutorial: Generating Occlusion Maps

Generating an occlusion map using Mayas texture baking features can be fun AND easy.

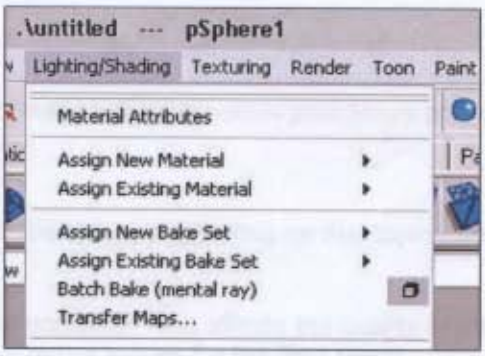


Step 1:
In the upper left corner of Maya, select Rendering from the drop-down menu.

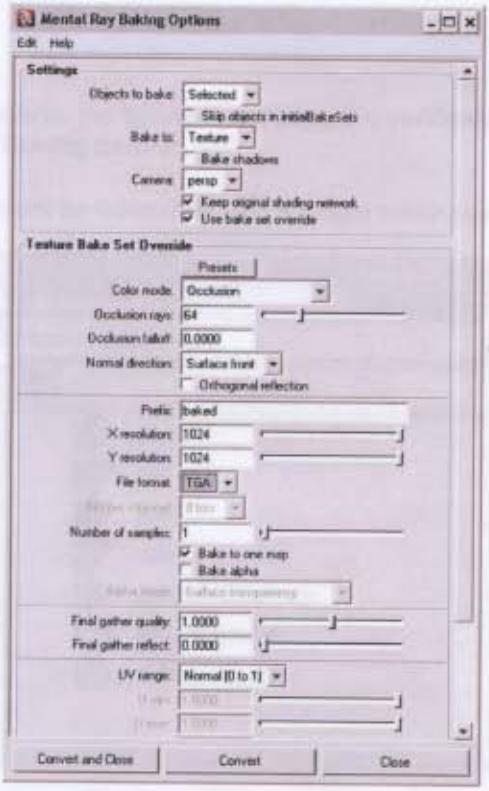


Step 2:
Right click and hold on the object to be baked and select the desired UV set from the UV Sets menu. This will tell the texture baker which UV set to use when baking the texture.

Step 3:
Ensure the objects for baking are selected and, in Mayas main menu bar, and select the options box from Lighting/Shading > Batch Bake (mental ray).



Step 4:
In the baking options dialog, set the values of the attributes to match those that are in the image below. Ensure Keep Original Shading Network and Bake to one map are both selected, and that the Color Mode is set to Occlusion.



Step 5:
Set the Occlusion Rays to a low number (32) for preview, and a high number (128) for the final quality.

Step 6:
Set the prefix to be the filename of the occlusion map. This is specified in the Asset Spec.

Step 7:
Set the proper X resolution and Y resolution for the output occlusion map. These dimensions are specified in the Asset Spec.

Step 8:
Set the File format options to TGA

Step 10:
Press Convert and Close. The output image will be saved in your Maya project folder, under:
`renderData\mentalray\lightmap`

Tips:
Generally the falloff should be set to 0, however, depending on the object, that value may need to be tweaked to get the desired result.

As stated in the tutorial, the number of occlusion rays affects the quality of the final image, and also render time. Use smaller values for quick previews, larger values for the final result.

Sometimes there may be black seams baked into the map of the object. If this is the case, adjust the Fill texture seams attribute (not shown in the above image) to fill in the gaps.

Submission

Maya Scene

Before submitting a file back to LucasArts, the scene **must** pass the verification process. The included verification script will check for the following issues:

- All objects in the scene must be combined into a single mesh (see image below).

Example of an outliner from a "clean" scene.

- Top-most transform should share the same name as the Maya file.
  - If scene is called **pan_whiskeyCrate.ma**, then the top-most node in the scene must also be called **pan_whiskeyCrate**.
- Objects and pivots should be at the world origin (0, 0, 0).
- There should be **no lights** in the scene
- The mesh in the scene **must be made of polygons**. Sub-division surfaces and NURBS surfaces are not allowed.
- There should be no **lamina faces** (two or more faces that share all the same edges).
- Every face **should have some surface area above the tolerance of .00005 (1e-5) centimeters**.
- There should be **no edges of zero length**.
  - Two unwelded/unmerged vertices, connected by an edge, sharing the same space.
- There should be **no unshaded faces**.

- There should be **no empty UV sets**.
- No objects should be attached to the default **lambert1 shader** (initialShadingGroup).
- No objects are allowed to have any shaders **EXCEPT lambert, blinn, or phong**.

These parameters can be checked by running the verify scene script sent with this packet. If there are any issues that generate errors, they will be displayed in an error report UI.

If errors are encountered before submission, it is up to the artist to correct them before resubmitting the asset. **Any asset containing errors will be rejected immediately and sent back for correction.**

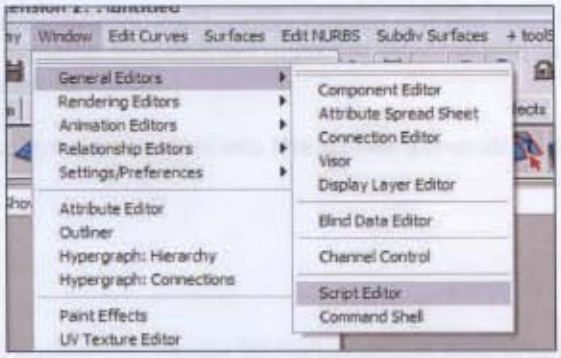
In addition to the error check, the artist must ensure the following:

- Maya scene and textures must be saved with the filenames provided in the Asset Spec.
- There **should not be any history on any nodes**.
  - History can be removed by clicking on **Edit > Delete All by Type > History**

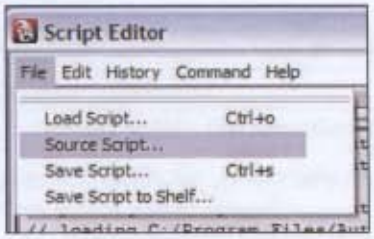
Tutorial: Using the Verify Scene Script

To use the verify scene script, it just needs to be sourced from within Mayas script editor.

Step 1:
Open Mayas script editor. This can be found by clicking on Window > General Editors > Script Editor.



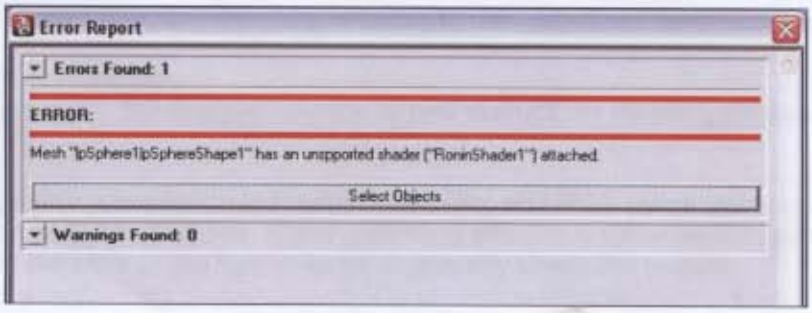
Step 2:
In the Script Editor window, click on File > Source Script.



Step 3:

Navigate to the scripts folder of your packet and open lec_verifyScene.mel. This will start the verification proce

If an error is found during verification, it will be displayed in the error reporter window as either an error or a warning. If the error is an object that can be selected (such as an unshaded face), selecting the Select Objects button will select the offending objects for that error, allowing for the artist to easily correct.



After a single error is corrected, repeat steps 1 through 3 to see if anymore errors are in the scene. The Maya scene is ready for submission if there are errors reported in the error report window.

Textures

Before submitting a PSD file back to LucasArts, please observe the following rules:

- Textures should be named and size according the specifications called out in the Asset Spec.
- **Ensure that a PSD, as well as a TGA are being submitted.**
- **Do NOT, under any circumstances, completely flatten out the PSD file.**
  - Further down the pipeline, an artist at LucasArts may have to tweak a setting or two in order to get the most from the texture.
- If submitting a normal map, **be sure to include the hi-res geometry/source image** used to generate normal map.

Submission Checklist

Before submitting the files back to LucasArts, please be that all of the following criteria have been met:

Requirement	☑
Are all required files finished an approved for submission?	
Are all required files named correctly and in the correct directory structure?	
Did all Maya files pass submission verification with zero (0) errors?	
Are all color maps properly color balanced?	
Do the poly counts of all models fall within the specified range?	
Does the number of assigned materials for all models match the numbers of allowed materials in the Asset Spec?	

Glossary

Asset Package –
The whole package of materials given to the artist for an asset. Includes the **asset spec**, this document, stub files in the correct directory structure, shaders to use, reference materials, and scripts.

Asset Spec (or Asset Specification) –
The list of requirements, deliverables, reference, and instructions detailing the requested asset.

Color Balance –
The process by which a textures black, white, midtones, and RGB colors are normalized to proved a consistent color space for all assets. Color balancing effectively removes the color of light from the images, allowing the color of the light in game to properly shade the texture.

Lamina Faces –
Any two or more faces that share the same edges.

LOD (or Level of Detail) –
A lower resolution version of a particular asset. Used as a performance-enhancing placeholder in the game when an asset is far away from the camera. **LODs** are generally 0-based. This means the *highest* polygon model is at LOD[0]. The next lower resolution version is referred to as LOD[1], and so on.

Material –
The Maya material and its associated **shading network**.

N-Gon –
A polygon that can contain 5 or more edges and vertices. A hexagon, pentagon, etc.

Quadrangle (or Quad) –
A polygon containing 4 vertices and 4 edges. A square.

Shader –
See **material**.

Shading Network –
The nodes and attributes connected that define the behavior of a Maya **material**.

Triangle (or Tri) –
A polygon containing 3 vertices and 3 edges. A triangle.

UV –
A 2-D coordinate specification, similar to (x,y), used to map 2-D images to a 3-D model. Units along **U** specify the width, whereas units along **V** indicate height.

UV Set –
A discrete UV layouts. Models for the game can have up to 4 different UV sets to achieve desired shading results.