Marvin Landis

Normal Map Create

.PDF by
basilio noris aka dies-irae

Introduction

Modern Real-Time application look good, but pre-rendered animations are far ahead in terms of image quality, models complexity and lightning techniques.

A great handicap for realtime graphics is the polygon count. RT engines can yield a limited amount of polys at acceptable framerates (25-30fps or more).
Many methods have been developed for polygon reduction, but so far, the simplified, or lowpoly, models dont look as nice as the hipoly ones.
Color or bump textures are used to add detail to the models, but the results are far from satisfying.

A recent technology (that will appear in the next generation video games) uses vector normals maps on a lowpoly model to compute what shape his hipoly relative would have. The technique has many flaws but the quality of the models has made a great leap while keeping the always problematic polygon count low.

Starting out as an exporter for ATI's NormalMapper (a toolset for developers to generate and test normal maps) the plugin has grown to a complete lw solution for rendering lowpoly models with hipoly normal maps.

Getting started

If you’re like me, you are reading this document just to know HOW TO DO IT wondering how long it will take to get there. I’ll try to make it as painless and swift as possible:

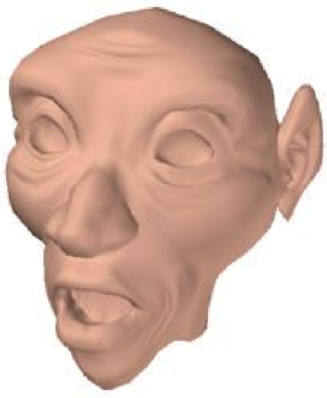
STEP 0: Adding the plugins

For now, just add the 4 plugins (*Modeler \ Plugins \ Add Plugins*) I’ll explain later what each one of them does.

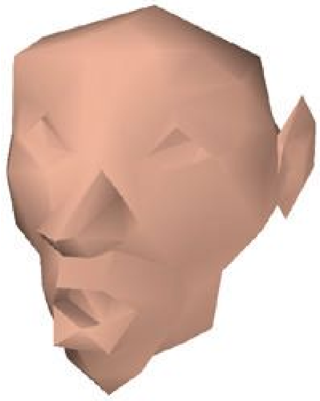
STEP 1: Preparing the model

You’ll need 3 things.

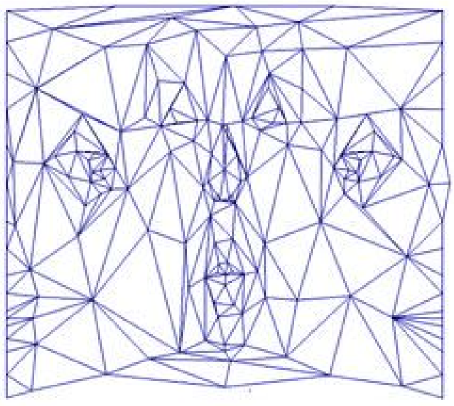
The HIPOLY model



The LOWPOLY model

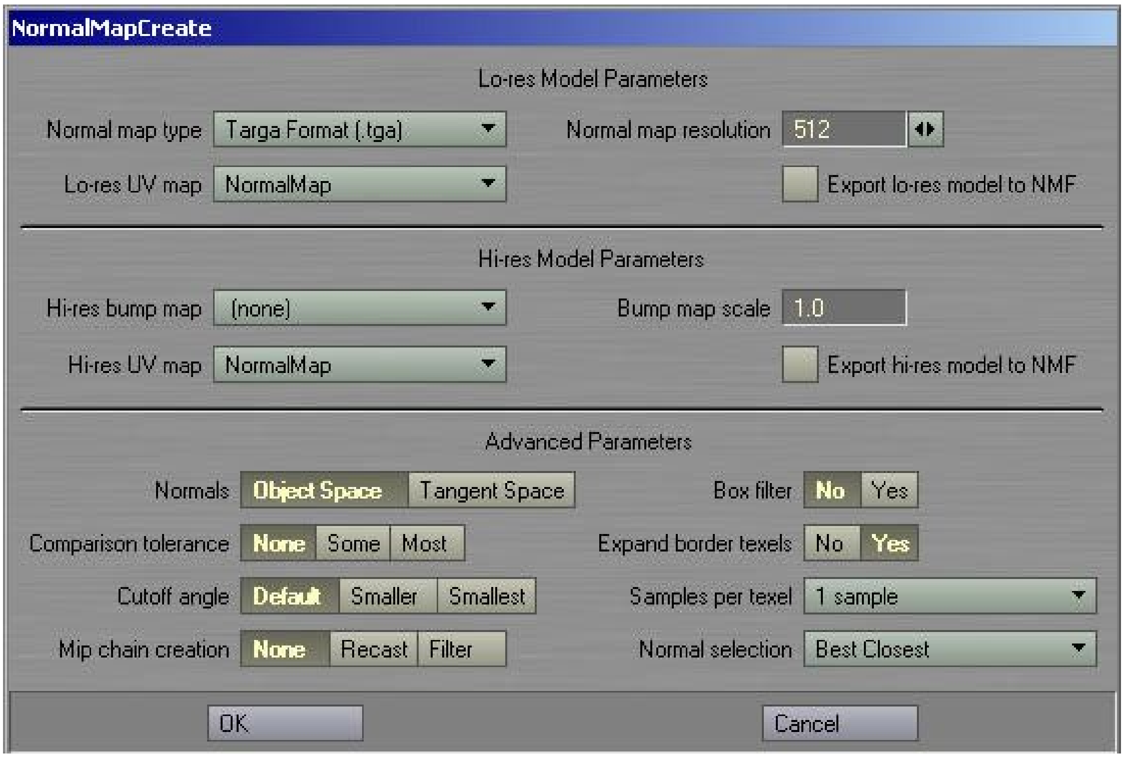


An UV MAP for the LOWPOLY model



Have the LOWPOLY in the *Foreground Layer* and the HIPOLY in the *Background Layer*.
Make sure not to have any endomorph map selected. Switch to *(base)* if you have morph maps in the model (*M* button on the lower right part of the screen).
Set the HIPOLY surface to SMOOTHED (leave it flat if you’ll need the LOWPOLY to be flatshaded)
Launch the NormalMapCreate plugin.

STEP 2: Exporting the N-Map



NormalMapCreate

Lo-res Model Parameters

Normal map type: Targa Format (.tga)

Normal map resolution: 512

Lo-res UV map: NormalMap

☐ Export lo-res model to NMF

Hi-res Model Parameters

Hi-res bump map: (none)

Bump map scale: 1.0

Hi-res UV map: NormalMap

☐ Export hi-res model to NMF

Advanced Parameters

Normals: [✓ Object Space] Tangent Space

Box filter: [✓ No] Yes

Comparison tolerance: [✓ None] Some Most

Expand border texels: No [✓ Yes]

Cutoff angle: [✓ Default] Smaller Smallest

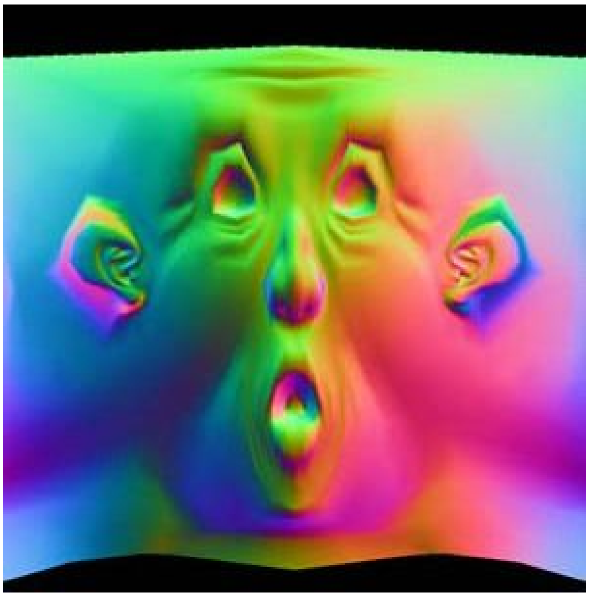
Samples per texel: 1 sample

Mip chain creation: [✓ None] Recast Filter

Normal selection: Best Closest

You can leave the settings as they are, what you might want to change is the Normal Map resolution (1024 works fine). I'll explain later the use of the bump map and the other options.

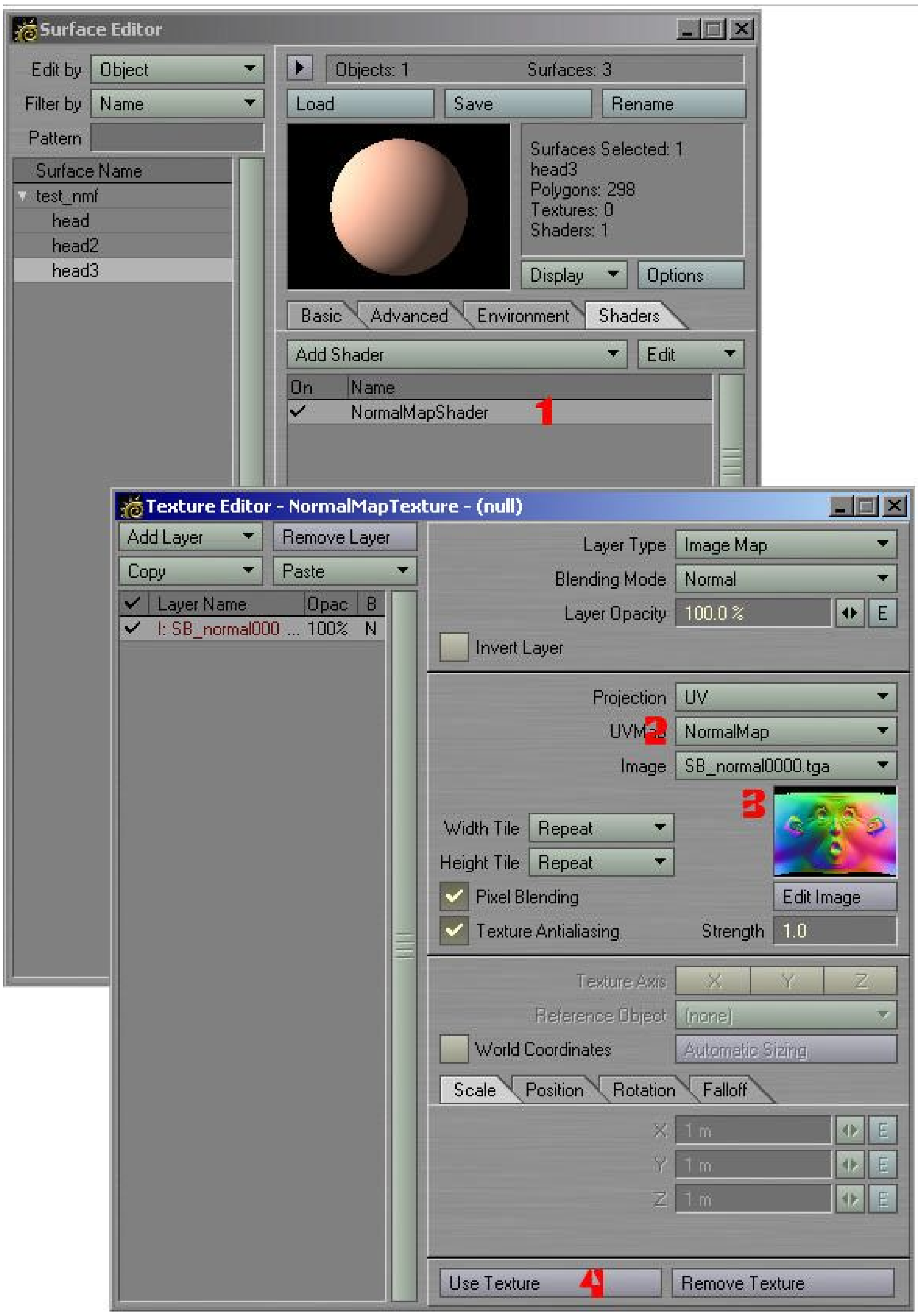
Press ok and save the image, you should get something like this:



You can close *modeler* now and load the LOWRES object into *layout*.

STEP 3: Loading the N-Map

In *layout*, have the LOWRES object loaded and open the *surface editor* for the object surface(s).



Surface Editor

Edit by: Object
Filter by: Name
Pattern:
Objects: 1 Surfaces: 3

Surface Name
test_nmf
head
head2
head3

Surfaces Selected: 1
head3
Polygons: 298
Textures: 0
Shaders: 1

On | Name
✓ | NormalMapShader

Texture Editor - NormalMapTexture - (null)

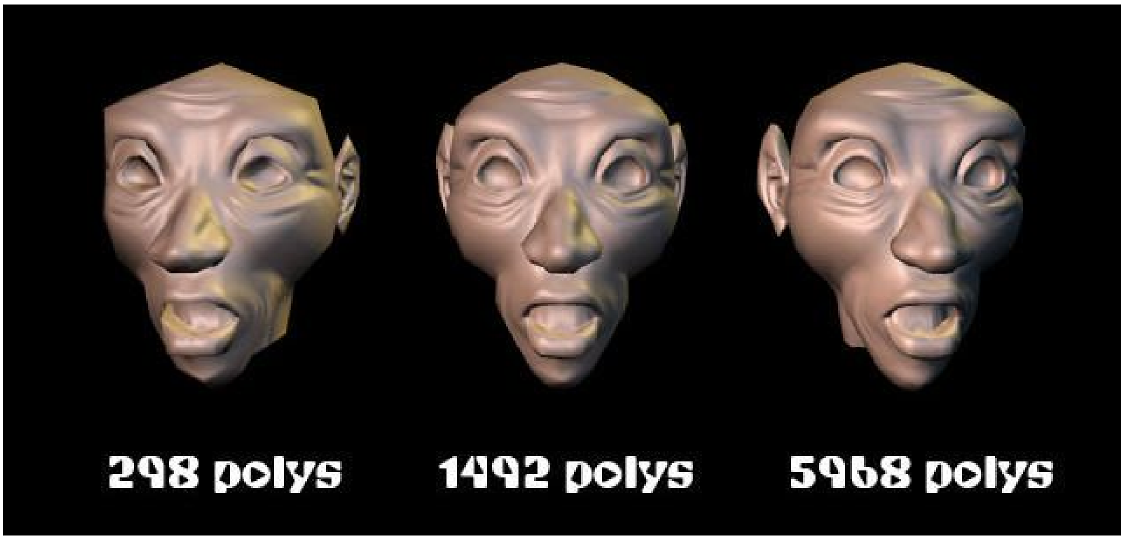
✓ | Layer Name | Opac | B
✓ | I: SB_normal000 ... | 100% | N

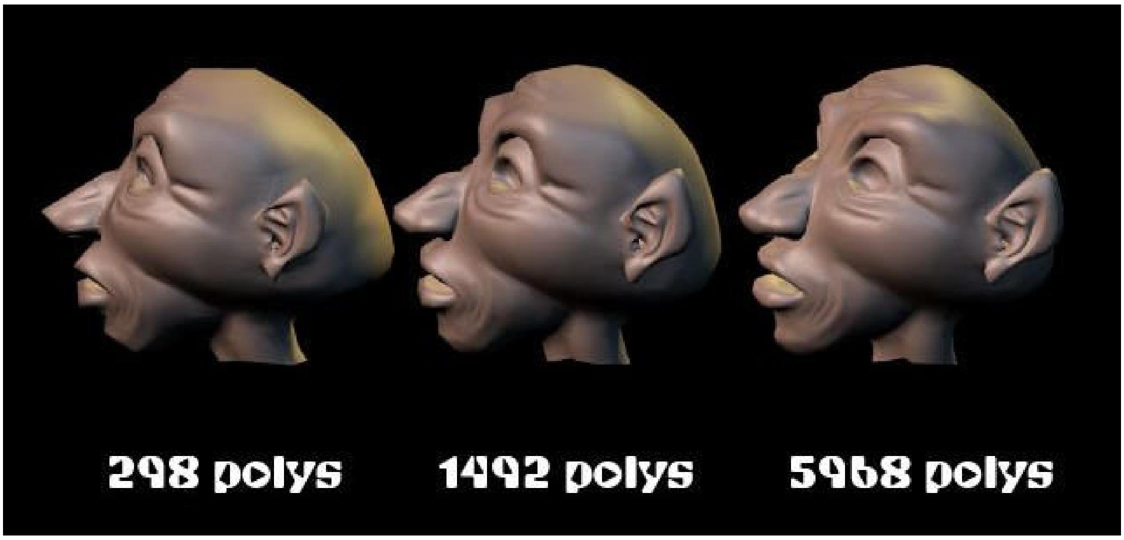
Layer Type: Image Map
Blending Mode: Normal
Layer Opacity: 100.0 %
☐ Invert Layer
Projection: UV
UVMap: NormalMap
Image: SB_normal0000.tga
Width Tile: Repeat
Height Tile: Repeat
☑ Pixel Blending
☑ Texture Antialiasing
Strength: 1.0
Texture Axis: X Y Z
Reference Object: (none)
☐ World Coordinates
X: 1 m
Y: 1 m
Z: 1 m

under the *shaders* tab add the NormalMapShader plugin (1) and doubleclick on it. In the texture editor that opens select the desired UV Map(2), the N-Map image you want to use(3) and use texture.

STEP 4: Rendering

This is it! Just press F9 and enjoy…





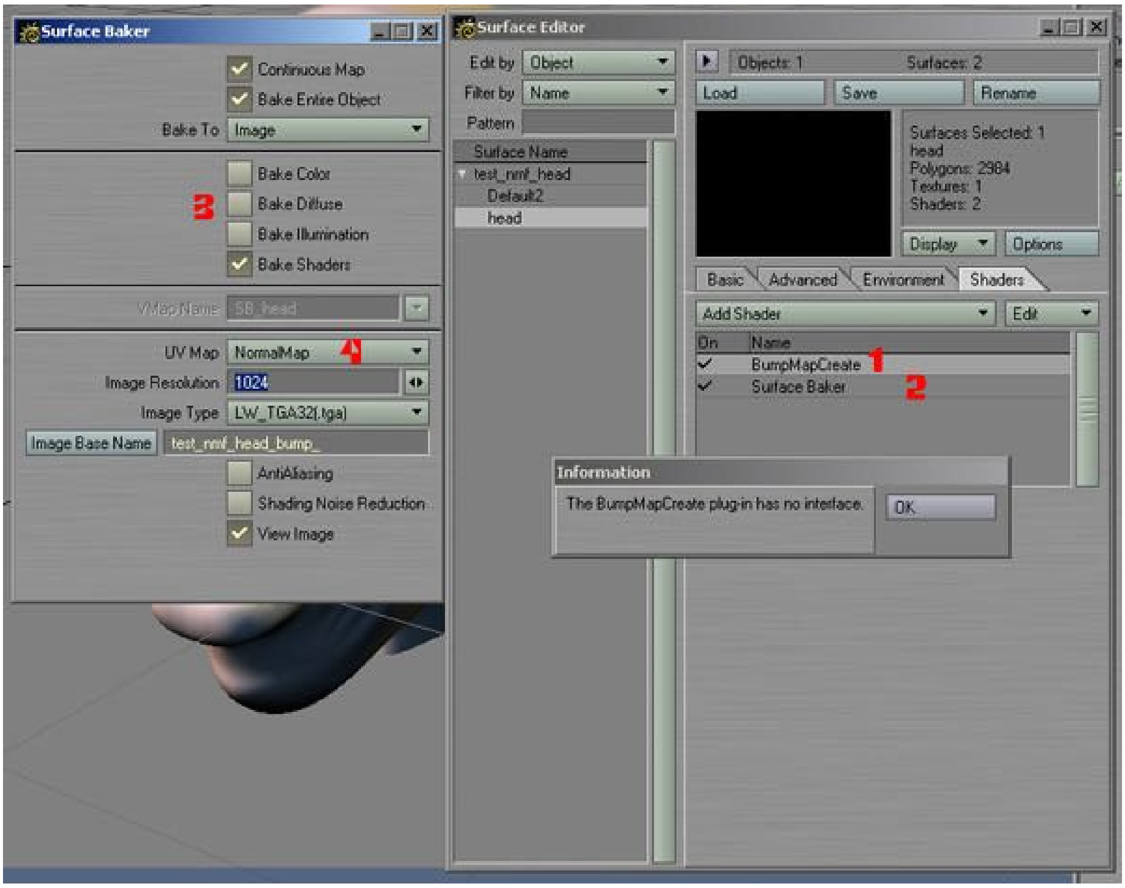
an important note about the bump map. The bumpmap is computed over the local normal of the pixels, but now the engine reads the normal from the N-Map, so it will ignore the bump data. To use bump maps, you'll have to use BumpMapCreate and the bumpmap option in NormalMapCreate. Go to the BumpMapCreate section below to know more.

Indepth

BumpMapCreate (Layout)

This plugin is used to bake the bump channel layers into a usable bump map. Since the bumpmap doesn't affect the N-MapShaded surfaces, you'll have to apply the bumpmap BEFORE the N-Map creation.

Create an UVMap for the HIPOLY model if you don't have one yet.



Surface Baker

☑ Continuous Map
☑ Bake Entire Object
Bake To: Image
☐ Bake Color
☐ Bake Diffuse
☐ Bake Illumination
☑ Bake Shaders
VMap Name: SB_head
UV Map: NormalMap
Image Resolution: 1024
Image Type: LW_TGA32(.tga)
Image Base Name: test_nmf_head_bump_
☐ AntiAliasing
☐ Shading Noise Reduction
☑ View Image

Surface Editor

Edit by: Object
Filter by: Name
Objects: 1
Surfaces: 2
Surfaces Selected: 1
head
Polygons: 2984
Textures: 1
Shaders: 2

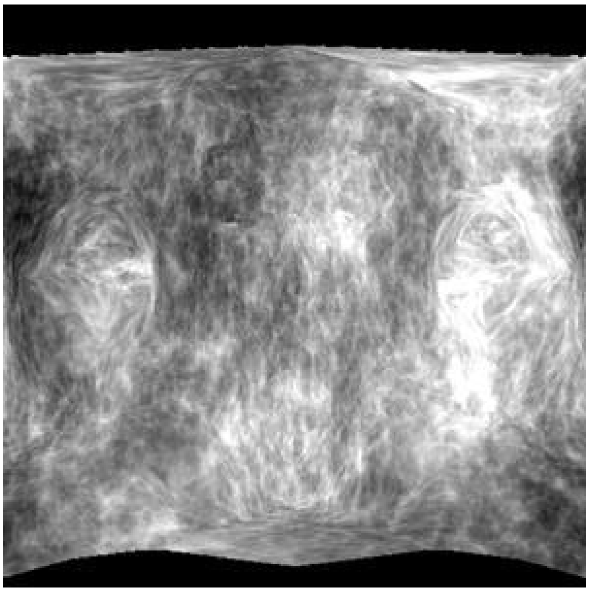
On	Name
✓	BumpMapCreate
✓	Surface Baker

Information

The BumpMapCreate plug-in has no interface.

Add the plugin to the surface(s) and add *surface baker* as well.
Open surface baker options and check only BAKE SHADERS.
Select the HIPOLY UV map and do a render, you'll obtain a baking of the bumpmap channel.
The resulting image might be too bright or too dark thus losing detail. You can use the *ImageViewerFP* exposition options or more easy, change the *DIFFUSE* value of the surface. You can always tweak the general value once you use the bump map.

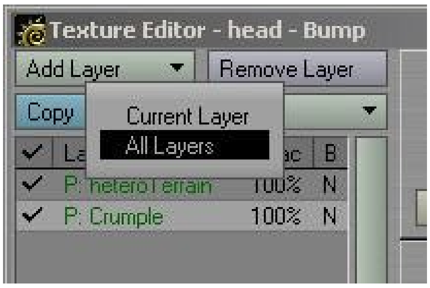
(this image was rendered with a *DIFFUSE* of 50%, a 100% diffuse rendered out as a white image)



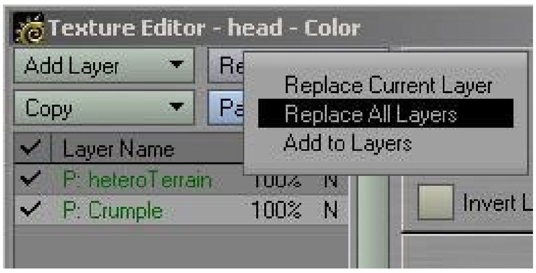
Right now *ALPHA blend* mode is not supported, so those layers wont be taken into account. (There seems to be some issues with layers with negative values)

If you do need to use *ALPHA blending*, you can use another method that doesn’t need the plugin.

STEP 1: Copy all layers from the surface's *bump channel*

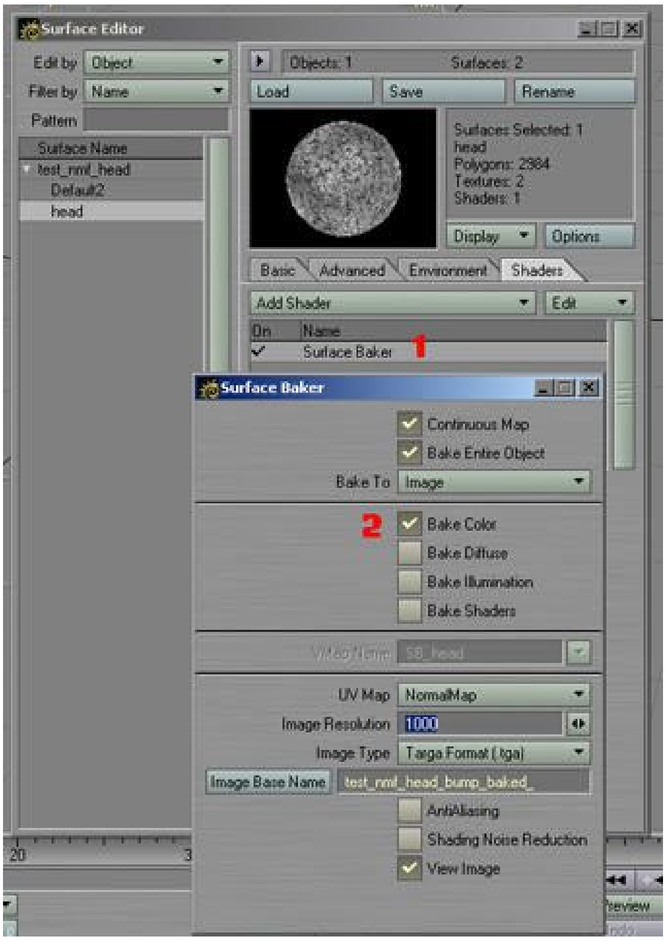


STEP 2: Paste all layers into the surface's *color channel*



then set the basic *surface color* to Black and the *texture layers color* to White, (or vice versa if you're using negative bumps). The *texture value* setting in the bump channel can be produced using different shades of gray. (for example: a 30% *texture value* becomes a 76 76 76 *color*)

STEP 3: Add *Surface Baker* and check only BAKE COLOR, be sure to select the correct UV map for the HIPOLY object.



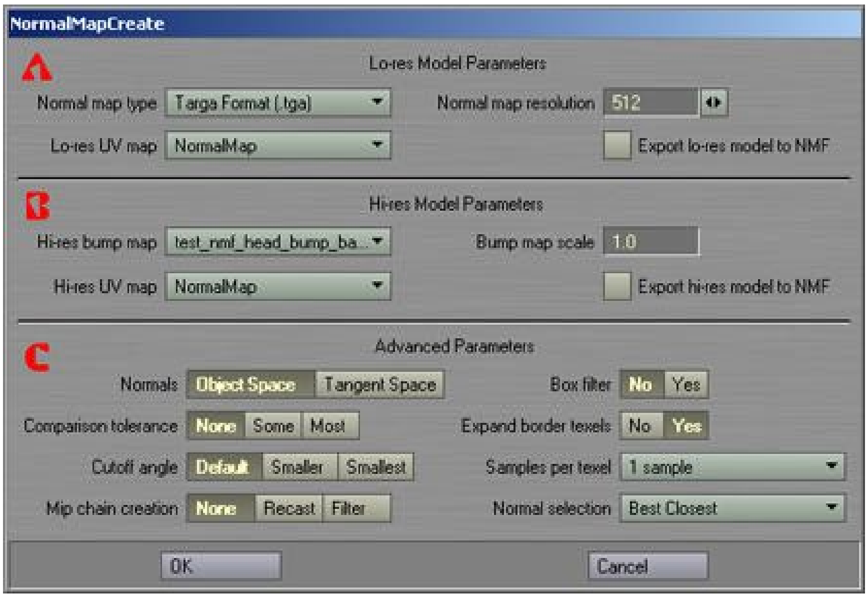
Edit by: Object
Filter by: Name
Surface Name
test_nml_head
Default2
head
Objects: 1
Surfaces: 2
Surfaces Selected: 1
head
Polygons: 2984
Textures: 2
Shaders: 1
✓ Surface Baker

☑ Continuous Map
☑ Bake Entire Object
Bake To: Image
☑ Bake Color
☐ Bake Diffuse
☐ Bake Illumination
☐ Bake Shaders
VMap Name: SB_head
UV Map: NormalMap
Image Resolution: 1000
Image Type: Targa Format (.tga)
Image Base Name: test_nml_head_bump_baked_
☐ AntiAliasing
☐ Shading Noise Reduction
☑ View Image

You can then render out the texture. If you use AntiAliasing, the render can take a lot of time.

NOTE: The method is quite elastic, you can chose to use the *luminosity* or *diffuse* channel rather than the *color* one, each method needs some tweaking to get the right result.

N-MapCreate (Modeler)



Normal map type: Targa Format (.tga)
Normal map resolution: 512
Lo-res UV map: NormalMap
☐ Export lo-res model to NMF
Hi-res bump map: test_nmf_head_bump_ba...
Bump map scale: 1.0
Hi-res UV map: NormalMap
☐ Export hi-res model to NMF
Normals: Object Space
Box filter: No
Comparison tolerance: None
Expand border texels: Yes
Cutoff angle: Default
Samples per texel: 1 sample
Mip chain creation: None
Normal selection: Best Closest

The Lo-res Model Parameters

Only thing to notice the "Export lo-res model to NMF" option. This exports an .nmf file which can be used with the ATI normal map tools. Use it if you want to play a bit with these utilities. The rest is pretty straightforward.

The Hi-res Model Parameters

We've just seen how to create a bump image map from the bump channel.
Just select the desired bump map file (you'll have to load the image from the image editor BEFORE you launch the plugin to see the image in the dropdown menu)
The bump map scale is used to tune down (or less likely up) the bump. If you created yourself the bump map, you probably wont need to change this setting, but if the bump is too strong, you'll have to set this option to lower values. You can of course use negative values.

NOTE: the bump images created with BumpCreate normally needs a *Bump Map scale* of 0.2

Once again, you can export the NMF object to be used with ATI's normal map tools.

Advanced Parameters

Most of these options are interesting only if you plan to play with ATI's normal mapper tools, otherwise you probably wont need to change these settings
Here's a quick description for each one:

NORMALS: decides which coordinates space should be used. TANGENT is needed for ATI's NMFView. (see below)
BOX FILTER: applies a blur filter on the final image.
COMPARISON TOLERANCE: removes imprecisions on the floating point operations
EXPAND BORDERS: expands the normal region. It can cause some garbage with ATLAS maps, but somewhat improves the normal map quality.
CUTOFF ANGLE: determines if the normal from the HIPOLY faces roughly the same direction of the LOWPOLY. This eliminates some artifacts in some cases but can generate more in others. The smaller the angle the closer the normal in the high resolution must be to the normal in the low resolution model.
MIPMAP CHAIN CREATION: used for game developing, if you want to implement mip mapping. RECAST recomputes the image for every mipmap level. FILTER only blurs the map more and more.
SAMPLES PER TEXEL: more samples are used for every pixel, resulting in some kind of anti aliasing. The computing time increases linearly. Can cause garbage.
NORMAL SELECTION: decides which normals from the HIPOLY will be chosen for every point in the LOWPOLY UVMap when several normals are available.

(more informations can be found in the readme file of the ATI NormalMapper toolset.)

N-MapShader (Layout)

the plugin is pretty simple, add it and open it, it works exactly as the standard lightwave *Texture Editor*. Chose the N-Map file and the UVMap for the LOWPOLY object and render.

N-MapColor (Layout)

This plugin is a quick method for creating a Normal Map for an object. Add it to the shaders and use SurfaceBaker to save it as a texture map just like the BumpMapCreate plugin.
The map thus created is not dependent from the LOWPOLY method (i.e. you'll have to adjust the LOWPOLY UVMap to match the N-Map), and the results are far from the NormalMapCreate ones. The good point is that the method is lightning fast.

ATI Normal Mapper

ATI's Normal Mapper is at the origin of this set of plugins, you might be interested in playing around with it. You can download it from
http://www.ati.com/developer/tools.html
be sure to check the **NormalMap.ppt** file (you can download PowerPointViewer from www.microsoft.com) and the **readme.txt** file

Creating a New Map.

If you want to try it out, use N-MapCreate and check the export to NMF options.
This will create 2 .NMF objects. For example lores.nmf and hires.nmf
Open a dos command prompt (win+r "cmd") and type
`"normalmapper lores.nmf hires.nmf 1024 1024 normalmap.tga"`
where "1024 1024" is the texture size and "normalmap.tga" is the image name.

Viewing the objects.

You can see the .nmf files plus a tga normal map (either created with ati's own program or with N-MapCreate) with NMFViewer.
A note if you created the map with N-MapCreate. Lightwave uses a last to first row order when saving TGA's. If you want to use them with NMFViewer, you'll have to flip them. If you open and resave the TGA's with photoshop (or psp or almost every other image editing program) the image will be saved in first to last row order and the viewer will read it correctly.

Limitations and Issues

An important note, the raytraced or mapped shadows do not take into account the normal of the pixels, therefore the shadowing will be calculated over the low poly model, which might cause artefacts, especially with sharp shadows.
The specularity, reflection and diffuse color are calculated using polygons normals, so those are the parameters that take advantage of the N-Map information.
As already said, the bumpmap channel stops working once the NormalMapShader is activated. If you change the bump map, you'll have to recompute the N-Map, a process that can be quite boring but without solution.

NormalMapShader not rendering

There seems to be a problem on some LW systems. The NormalMapShader doesn’t seems to work and renders out only flat objects. The problem apparently resides in LW. A reinstall of lw fixes the problem. Another solution is to add the N-Map to the bump channel. I cannot say why this should work but it does the job ;)

Useful links

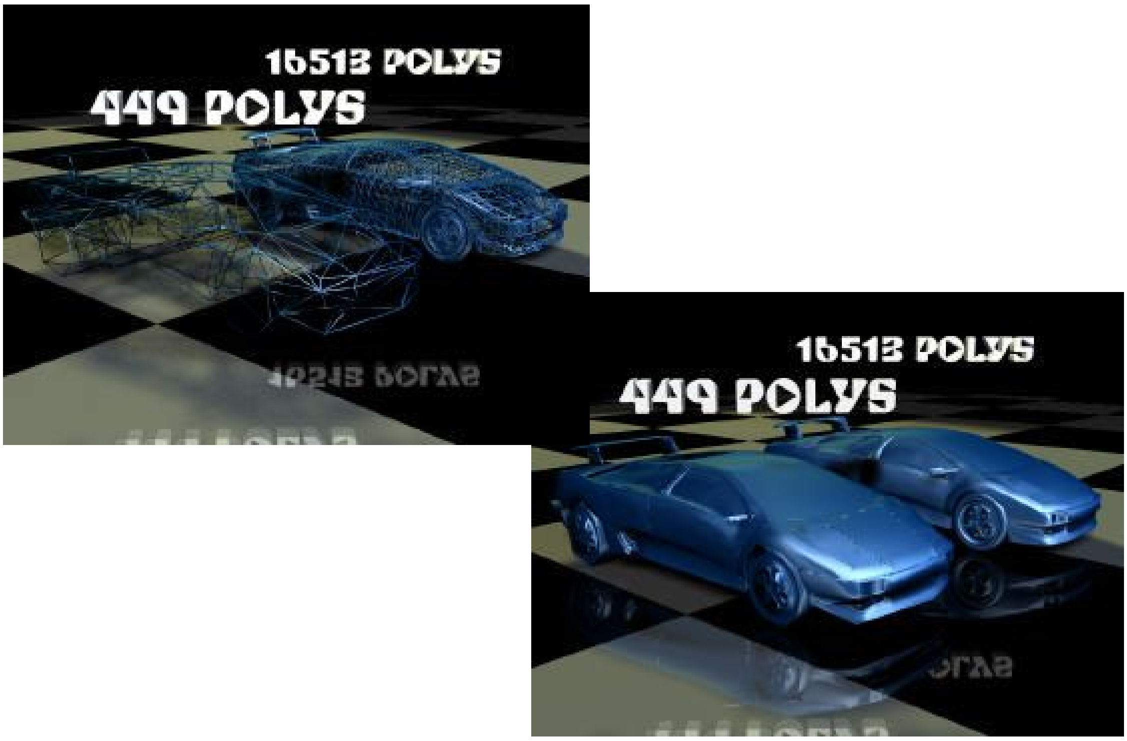
The Plugin Home Page	http://amber.rc.arizona.edu/lw/normalmaps.html
Marvin Landis Plugins	http://amber.rc.arizona.edu/lw/
This document URL	http://www.dies-irae.com/NMCManual.pdf
THE lw plugins page	http://www.flay.com
ATI's home page:	http://www.ati.com
ATI's Normal Mapper	http://www.ati.com/developer/tools.html
CGTalk Forum	http://www.cgtalk.com
StrokeIt	http://www.tcbmi.com/strokeit/

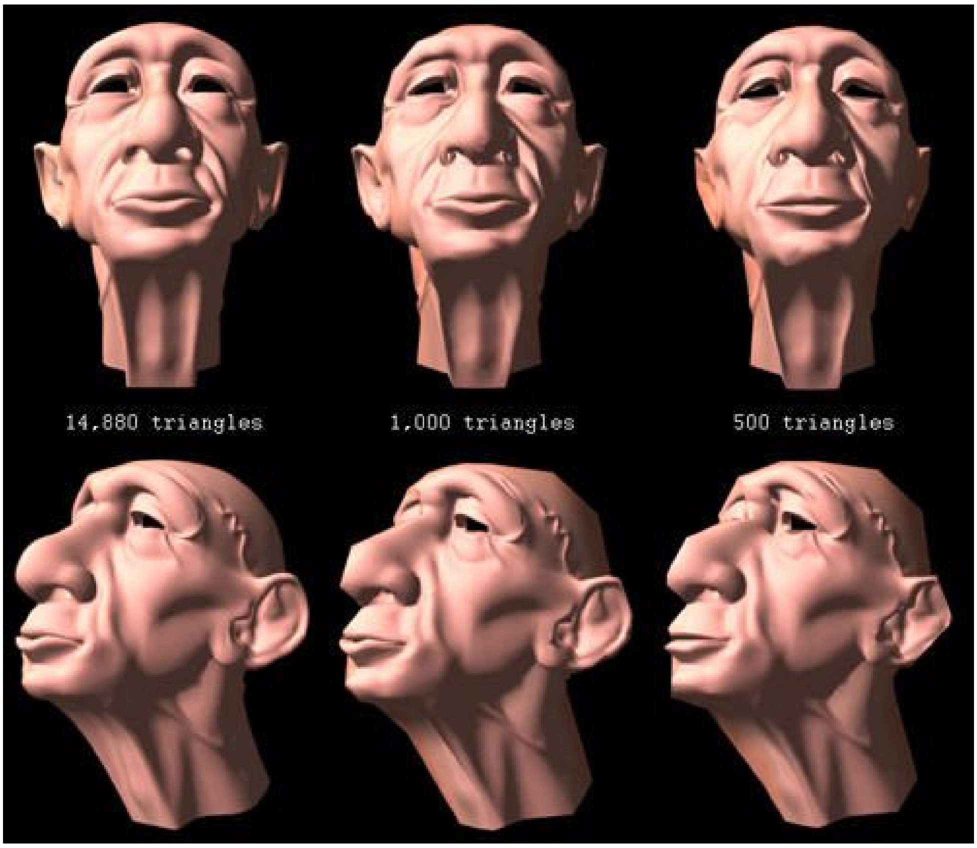
Your very humble meself (in case you might need anything)	dies-irae@dies-irae.com

Thanks and Credits

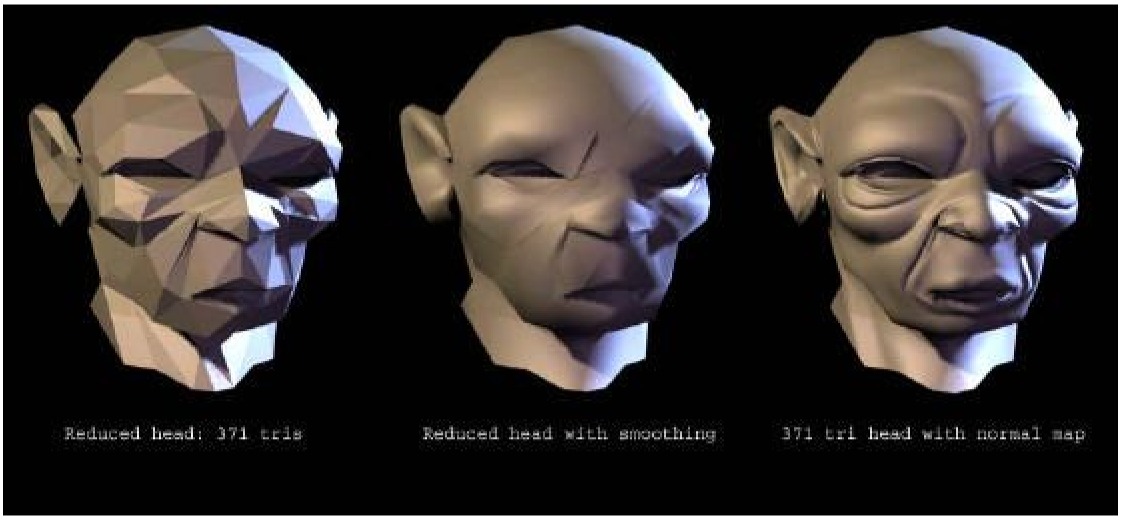
Marvin Landis, for writing this wonderful plugin!
Michael Blackbourn, for the great example images
PJ, for aiding in making this document
Geng@, same thing
You, for reading this far…
Sam, for cooking while I spent my time on the pc.
Kino, for teaching me how to play quake3

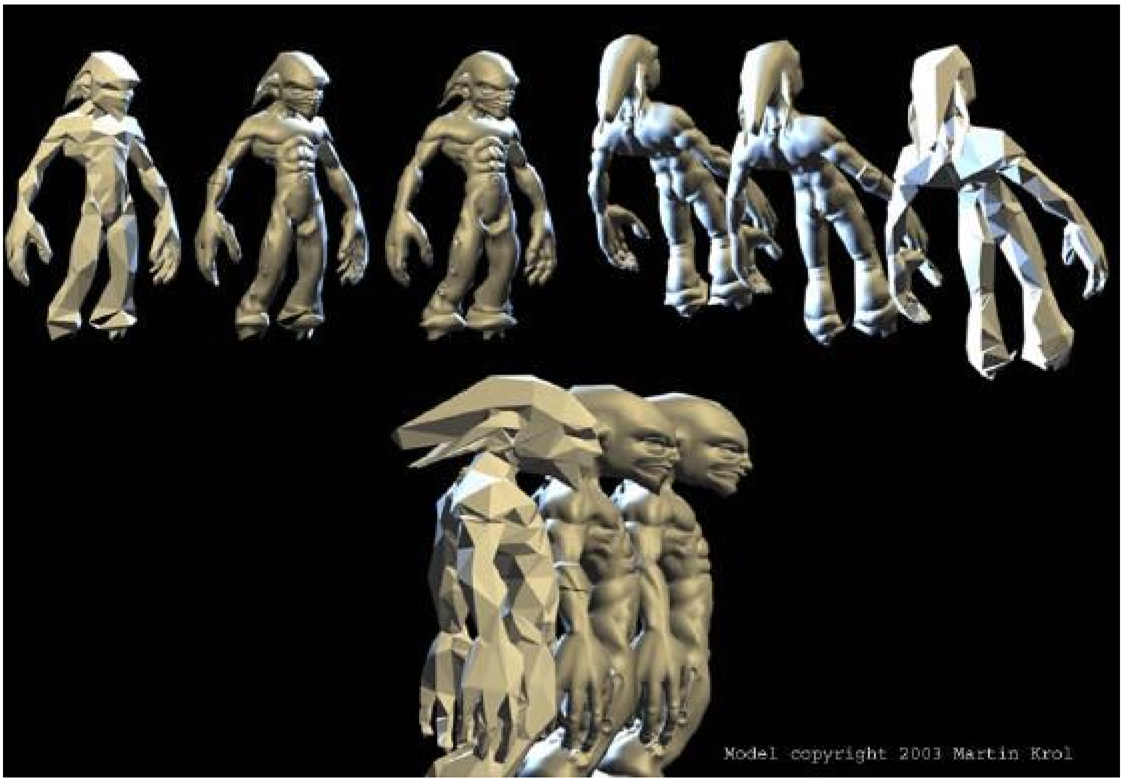
Some examples





gramps model by David Maas. Renders by Marvin Landis





images courtesy of Michael Blackbourn