VERSION 3

xN NORMAL MAPPER

Table of Contents

1. Introduction

1.1 What is xNormal?

xNormal is an normal mapper[1] tool designed to generate DOT3 bump / displacement / ambient occlusion / base texture baking maps from a very high poly and to map it into a low poly one. With this technique you can simulate millions of polygons in the low polygon model, because emulates the high polygon modelling using bump mapping. It also somes with useful tools like height map to normal map conversor, normal map to cavity map, height map to occlusion map, cone map computation, etc...

It can read / write the most used and popular formats in the market so you won't need any exporter.

xNormal is designed to be very easy-to-use and it doesn't mess with complicated options.

Some quick features:

Multiple input / output format support
Normal / displacement / occlusion / baked base texture maps fast generation
“Floating” geometry support
Complete C++ SDK and plugin system
Uniform two-sided / Non-uniform (cages) ray distances / MatchUV / ray blockers support
Allows UV Mirroring / Wrapping / Cylindrical coordinates
Supports quads and triangular faces
Easy to use
Advanced GPGPU rendering acceleration
Interactive 3D previewer with multi-monitor, advanced shaders, transparency, full-dynamic soft translucent shadows and displacement support
Tangent and object space normal map support
Complete documentation and practical examples
Advanced batch rendering including isolation and masking
Per-user serializable settings and partial UI skin customization
Automatic and visual installer / uninstaller
Exportation / Importation of tangent basis and models
Film quality (HDR[2] IEEE754 precission, supersampling) results
Fast “technical service” and direct talk with developers
Full localized application (support for multiple languages)
Advanced multicore / hyperthreading CPU support

1 **Normal map.** See http://en.wikipedia.org/wiki/Normal_map link for more info.

2 **HDR**. High dynamic range. xNormal support both input and output HDR image formats for accurate results. For more info see http://en.wikipedia.org/wiki/High_dynamic_range_imaging

1.2 Is this program free?

Yes, it is. If you like this application feel free to credit/mention it in your program. Here is the partial license (please take a look to the xNormal_legal_unified.rtf file in the docs dir after installation for further details) :

1.3 Hardware and software requisites

PC Windows x64 version:

You'll need a x64-compatible CPU, 2Gb of RAM and Microsoft Windows 7 or above operating system

The interactive 3D viewer requires a graphics card with almost DX10 Shader Model 4 hardware-accelerated shaders able to use almost a 1024x768 screen resolution. 512Mb of VRAM or above is required.

For optimal settings we recommend 4Gb RAM, a multicore-CPU and a graphics card SM4 with almost 1Gb of VRAM.

1.4 Installation

Windows PC version (Windows Installer):

Run setup.exe and follow the indications of the screen through the wizard setup. You will be asked for a target path and the application will install there.

1.5 Updates and contact information

You can always download the latest version of this software from:

http://www.xnormal.net

Also, if you want to contact me for bugs, feedback or suggestions please send us an email to:

contact@xnormal.net

1.6 Formats supported

The current 3D file formats supported are:

- .FBX
- .3DS
- .DXF
- .LWO
- .OBJ
- .MS3D
- .DAE
- .PLY
- .ASE
- .MESH
- .OVB
- .SBM
- .X
- .mesh
- .LXO
- .SIA/.SIB.
- .OFF

The current image file formats supported are:

- .BMP
- .TGA
- .JPG
- .TIFF
- .PNG
- .DDS
- .EXR
- .HDR
- .RAW
- .PSD
- .WEBP

2. Usage

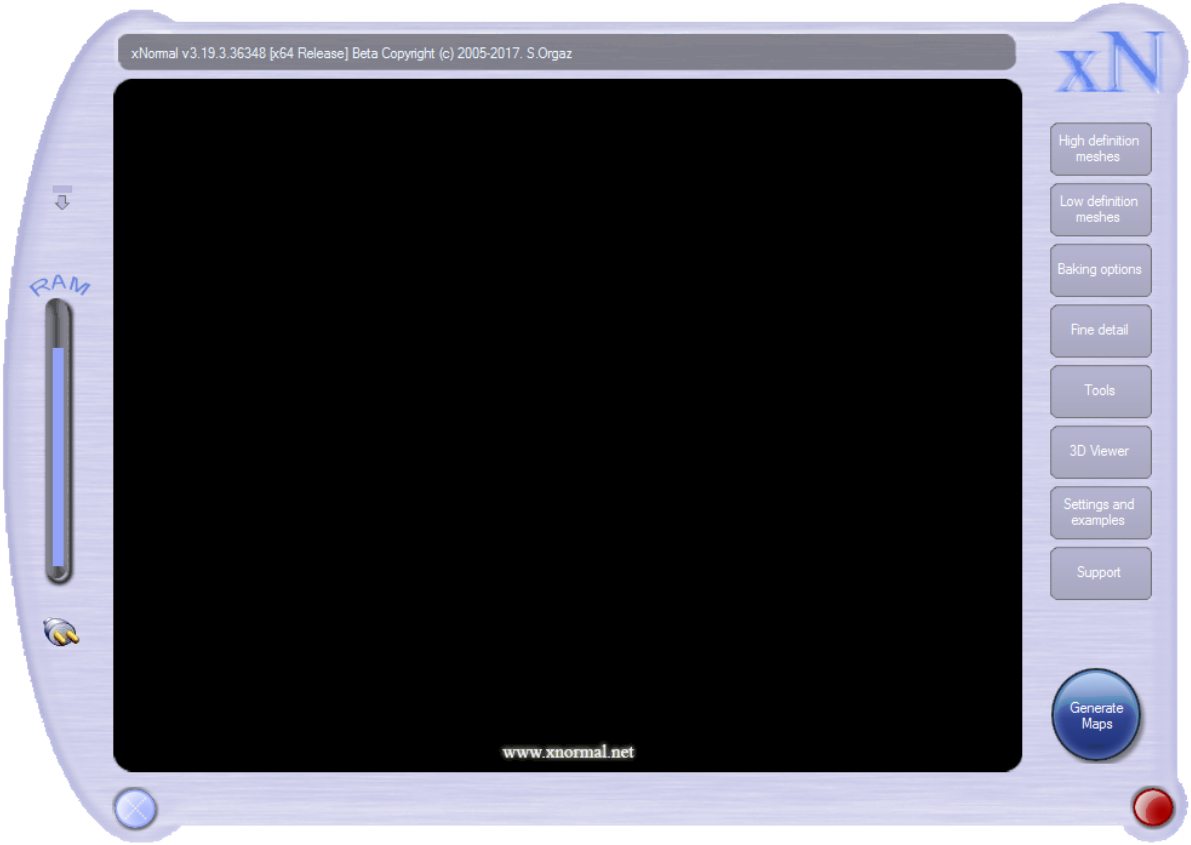
2.1 How to run the program

Windows PC version:

To run xNormal just go to Windows **Start Menu**->**Programs**->**xNormal** or go to your desktop and find the xNormal icon.

2.2 General description of the interface

This is the general aspect of xNormal:



On the right, you have the main tab controls: High polygon model, Low polygon model, Normal map settings, fine detail control, the viewer, settings/examples and support.

On the bottom, you have the Close button, the progress bar, a blink indicator to know if the program is halted and the generate button.

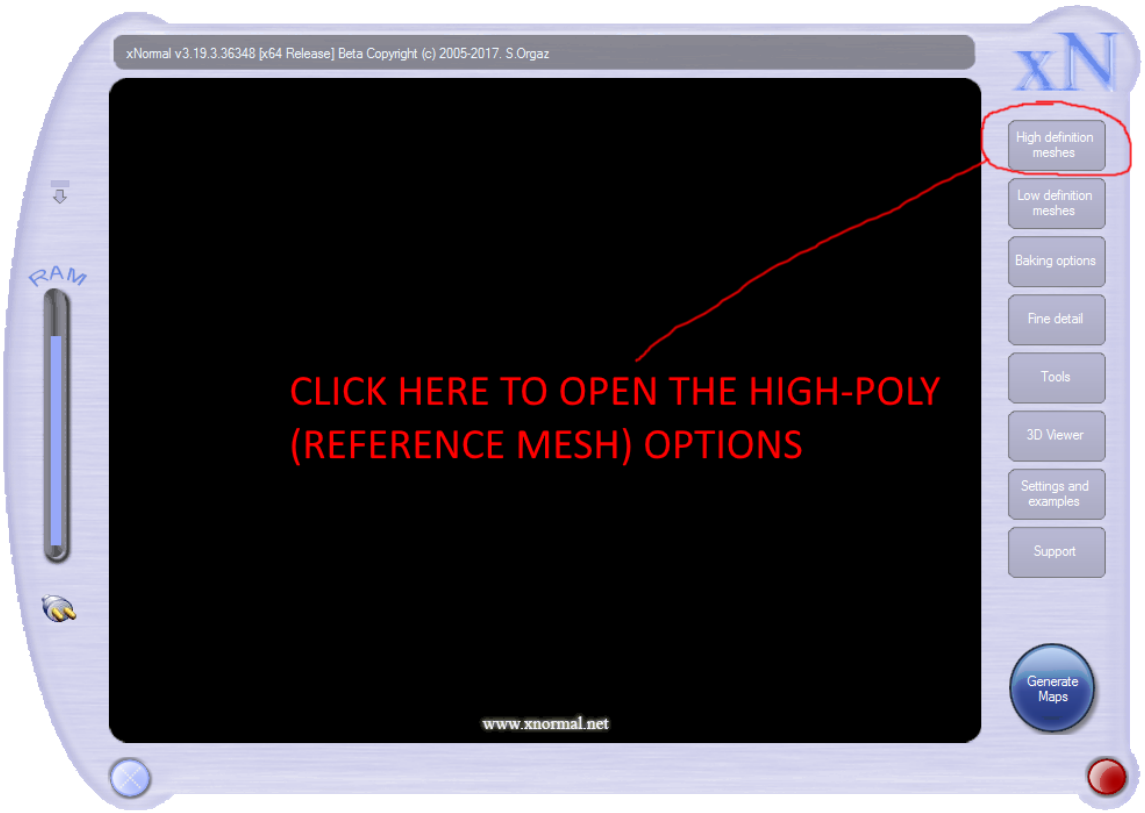
On the left, you have a free RAM indicator very useful to know if you are almost outta memory and the plug-in manager.

On the top-left corner, you have the icon to minimize the window.

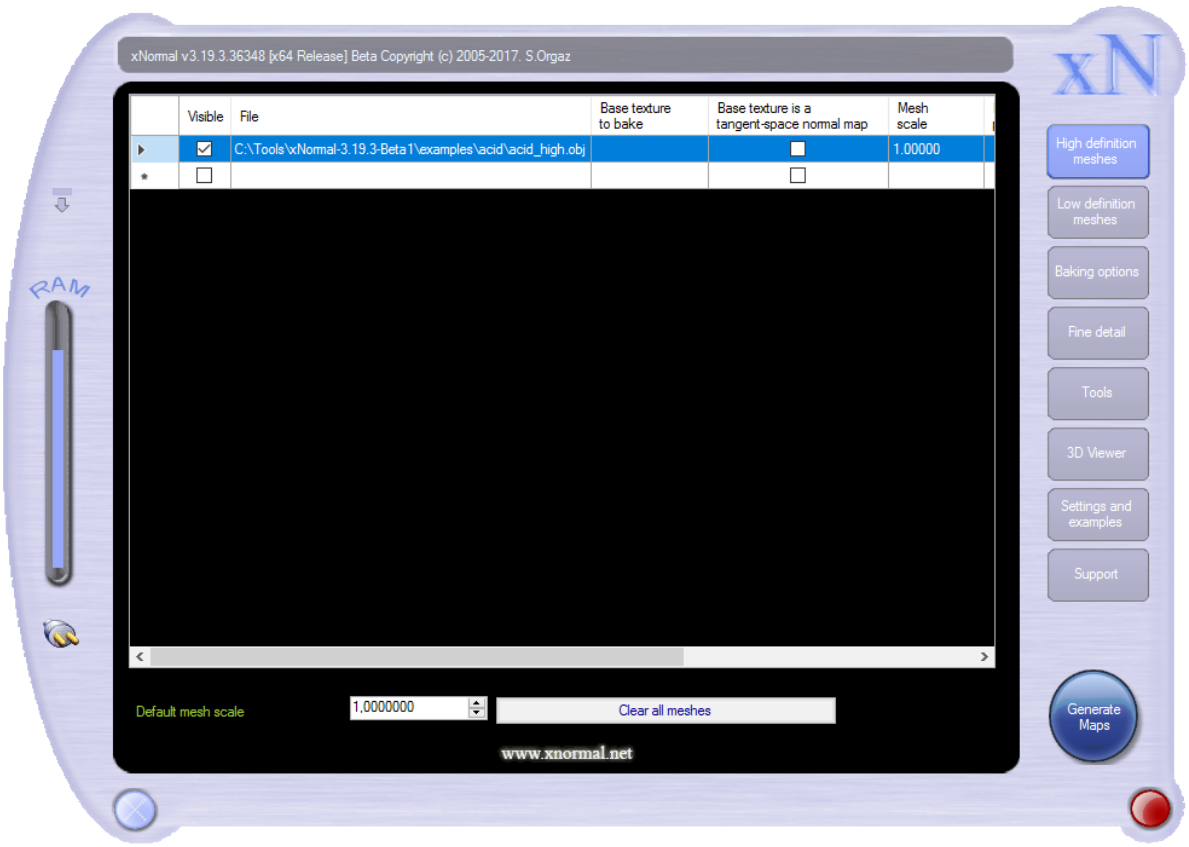
Now let's see all these parameters in detail :

2.3 High polygon model

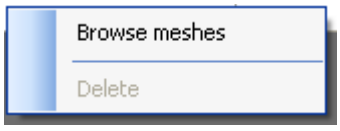
Go and click in “Highpoly model” tab:



The high-polygon model options will appear:



This is reference model, the high polygon file we want to gather the normals from. You can browse it on disk or network pressing the right mouse button to show the context menu:



Notice the program allows you to select more than one file in case you need to load multiple part meshes.

The “visible” checkbox controls if the mesh will be taken into consideration for raycasting (visible on) or if not (visible off, so won't be “solid”).

The high polygon model MUST contain vertex positions. If you want to use user-defined-per-vertex normals then, obviously, you need to export them. If no normals are present will be automatically calculated.

You can include UVs, but won't be used generally (unless you specify the “bake highpolygon base texture to lowpolymodel” feature), so you will waste disk and RAM space if you export them.

xNormal supports QUAD FACES and TRIANGLES. It does NOT support N-gons.

One thing to take into consideration is that you must be cautious with the number of polygons and vertices the high model :

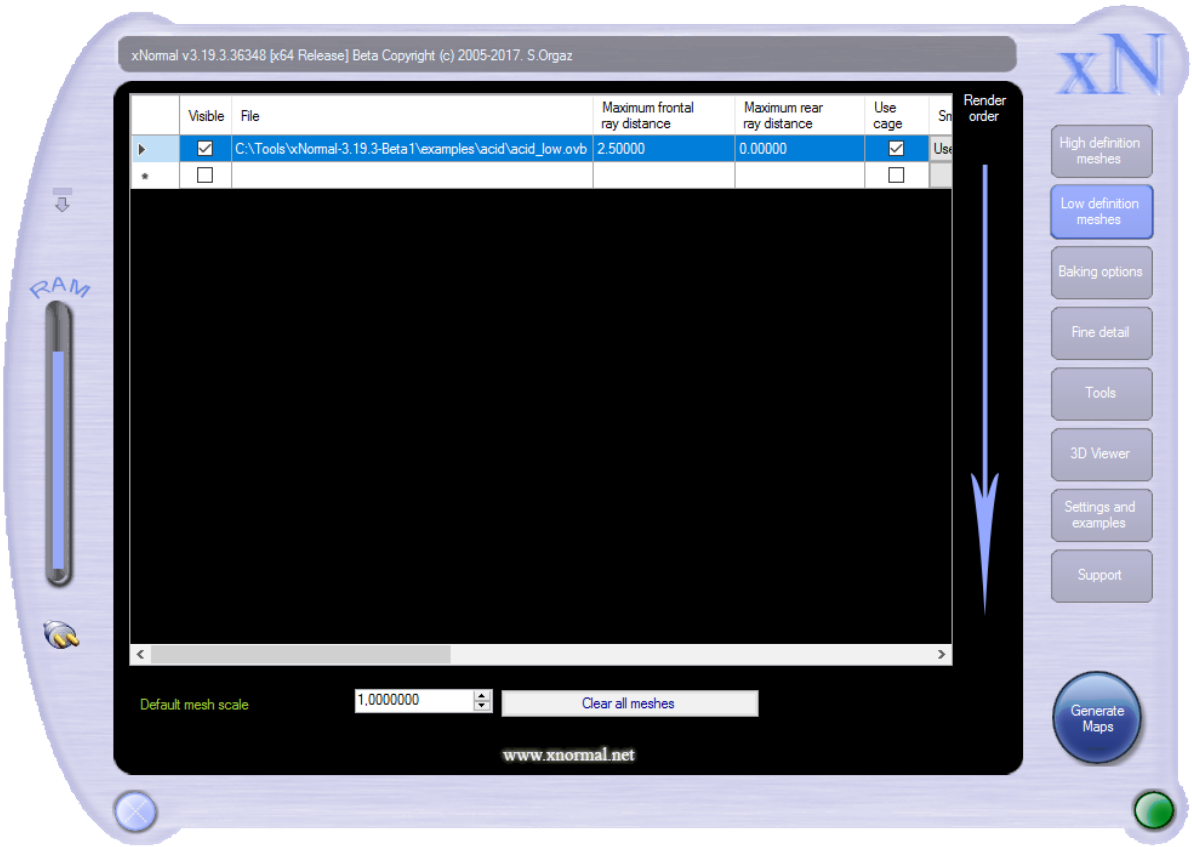
- **Memory**. You’ll need a lot of RAM to load the meshes and create the acceleration structures. Consider to split in multiple meshes your high polygon model and create the normal maps by parts. See the UI memory indicator to know how much memory is used at the moment.
- **Time.** As many polygons and vertices has your high model, the more time to read and process them.

Our advice is to export only vertex positions for the high-poly mesh and allow xNormal to automatically calculate the normals to save disk space and memory.

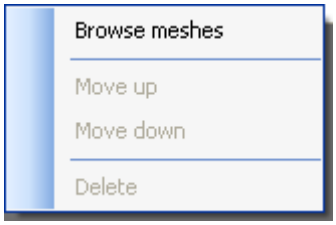
2.4 Low polygon model

Ok go to the “Lowpoly Model” and click it:

Now the low polygon model’s options will appear:

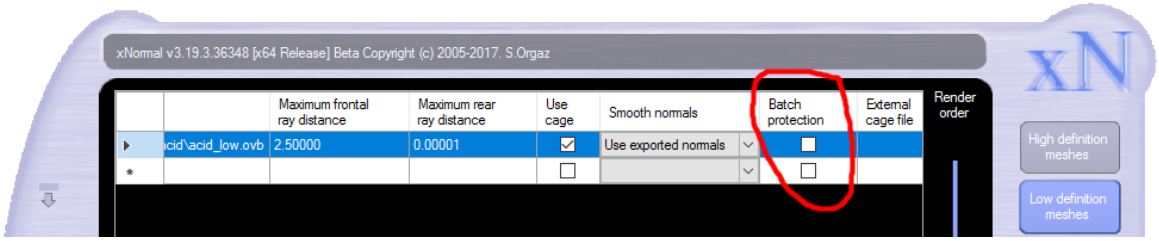


This is the working model, the low polygon file we will use in our 3D engine. We will apply the generated normal map to it to simulate the aspect of the high polygon model. You can browse it on disk or network pressing the mouse right button to show the context menu



Notice you can sort by columns and move elements up / down, because the order is IMPORTANT. Batch - rendering will start at TOP, moving to DOWN overwritting all the previous pixels by default.

If you want to protect some area of the normal map just check the “Batch Protection” option, so the pixels corresponding to that mesh file won't be overwritten by the next batch:



This option allows you to isolate / mask / protect portions of the normal map against writting.

Notice you can also hide a mesh checking/unchecking the “Visible” option, so will / won't be rendered into the normal map. This option is used too by the xNormal 3D viewer to show / hide the meshes. Hidden meshes won't be loaded neither rendered or painted.

The low polygon model MUST contain vertex positions and UVs. Positions are always needed for raycasting. UVs are required to fit the output normal map into the low model. If you want to use user-defined-per-vertex normals then, obviously, you need to export them or to check the “Smooth normals” option so will be auto-generated using face averaging.

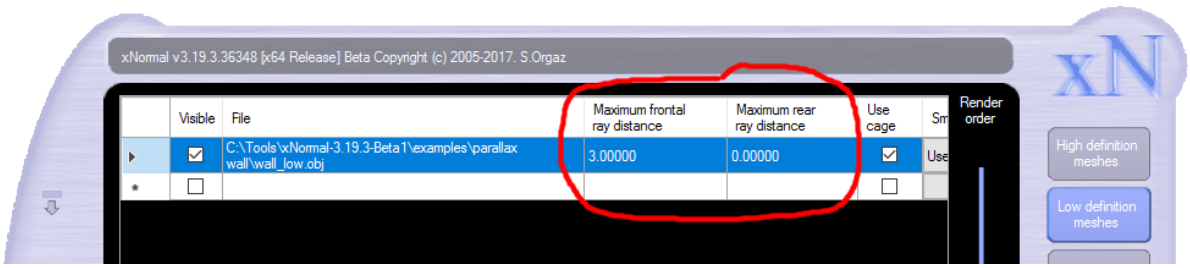
The model must be aligned with the high polygon one. Must have the same position and orientation when you export them.

xNormal supports QUAD FACES and TRIANGLES, it does NOT support N-gons.

Take into consideration too that some formats don't perform coordinate translation to a common system... OpenGL-based viewers/engines use a right-handed coordinate system, while Direct3D ones use a left-handed one.

2.4.1 Uniform ray distance and cages

This is the parameter that you must specify here:



As you can see, you can control outter (front) ray extent as well as inner (back) ray distance indepently. Using 0.0 you can avoid the ray to go front or back. If you set both front and back ray distances to 0.0 an error window will appear.You need to set almost one to greater than zero.

xNormal ALWAYS uses a ray-hit-furthest approach. If rays fail, then the program uses ray-hit-closest-double sided approach. To achieve this, you must specify a GOOD distance... As initial aproximation, place here your model radius divided by 20.

This parameter is CRITICAL. You have several options to set it right. Let's see some:

2.4.1.1 Measuring the ray distances with your favourite modeling application

Open your favourite modelling program and calculate it. Some 3D apps include ways to use a “tape” to measure an aproximated maximum distance between the lowpoly model and the highpoly model.

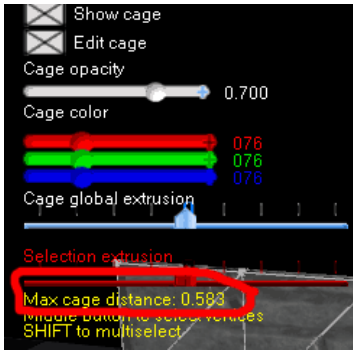
Using this method you can see the values to put for the front and back rays.

2.4.1.2 Measuring the ray distances using radius aproximation and “edit cage” feature

Load your lowpoly model into xNormal. Launch the 3D viewer:

Approximate it setting the object's radius divided by 20. This is usually a good number to start, but is far from perfect.

You can also show the cage and check the “edit cage” option so you could see the maximum cage extrussion precise value:



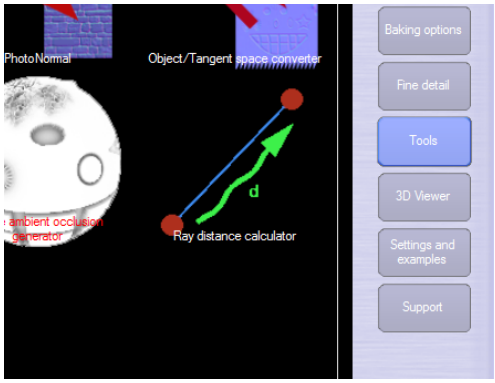
2.4.1.3 Measuring the ray distances with the “closest hit if ray fails” option

Check the “Closest if ray fails” option and set distances to a very low value**.** Put both front(outter) and back(inner) distances to very few (like 0.001). Then check the “Closest if ray fails”. This should work for the 90% of the meshes unless you have have floating geometry!

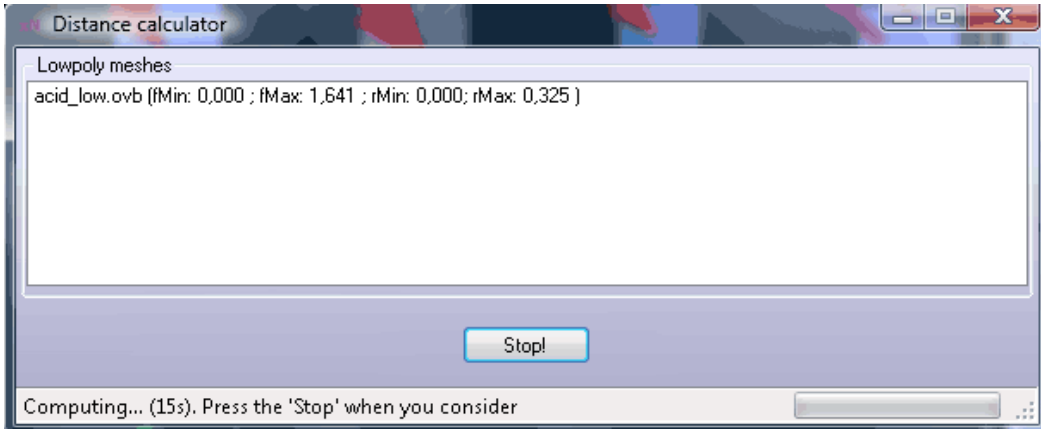
In that way, even if you specify an incorrect ray distance xNormal will use the closest surface available (which should be ok for simple objetcs like spherical objects).

2.4.1.4 Measuring the ray distances using the Ray Distance Tool

You can use the Ray Distance Tool to measure automatically the uniform ray distances. Go to the tools tab button on the right and click over it:

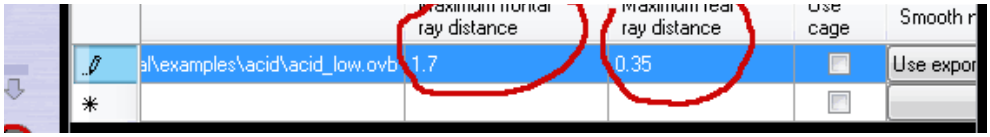


Then, a dialog will appear with all the lowpoly meshes that you have in the current settings. Just press the "Go!" button and xNormal will calculate the maximum and minimum frontal and rear ray distances:



The fMin is the minimum frontal ray distance (avg) between that lowpoly mesh and the highpoly meshes.
The fMax is the maximum frontal ray distance (avg) between that lowpoly mesh and the highpoly meshes.
The rMin is the minimum rear ray distance (avg) between that lowpoly mesh and the highpoly meshes.
The rMax is the maximum rear ray distance (avg) between that lowpoly mesh and the highpoly meshes.

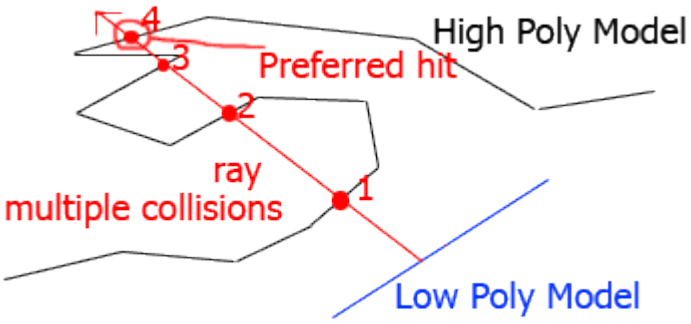
The computation won't finish until you press the "Stop!" button. The more time you allow the tool to work the merrier. I recommend to use almost 15 seconds to be precise.
Once you know the approximate distances, just copy the fMax and rMax (a bit augmented) to the lowpoly corresponding mesh slot:



2.4.1.5 Measuring the ray distances using “cages”

Use the “cages” method**.** See above. This is the ultimate method, but requires some work.

Problems can appear when you have some kind of “floating geometry” or a non-convex model like this:

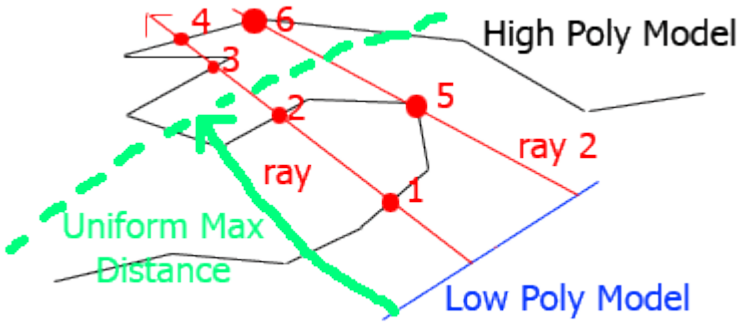


Notice the result can vary a lot depending on the value you put... See the image again... If you specify a short ray distance, you will get the hit #1 while if you specify a very large distance, you will get the hit #4.

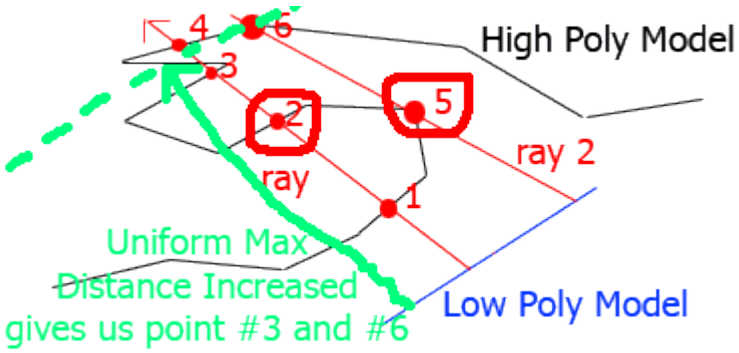
This value also helps a lot to optimize the raytracing... If you specify a good short one, tons and tons of posible collisions could be early discarded so it will take less time to render the normal map.

So, again, to conclude ... Set this parameter CAUTIOUSLY, specially if you have problems with the “floating geometry”, but don't worry too much if you don't set a correct one because xNormal will use the closest hit if the “Maximum Ray Distance” is incorrect or will use the lowpoly model normal if things become really weird.

Now see the following image:



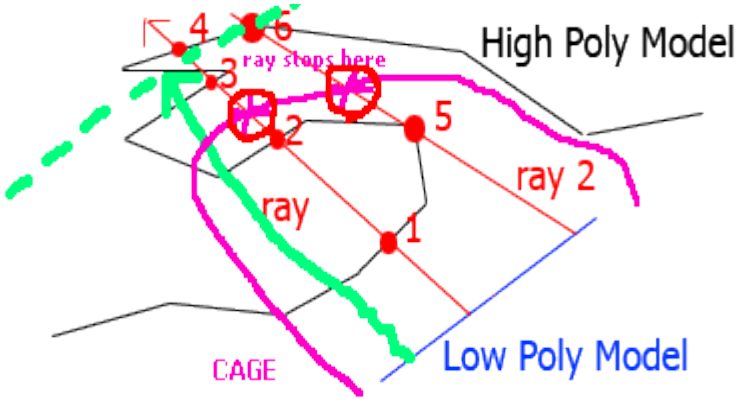
Imagine we set a constant max ray distance (the marked in green in the picture) to reach from ray #1 the hit point #2. Okie.. All fine... Now see ray #2 and imagine we want to reach the hit point #6... OOOH! The defined max ray distance in green will give us the stupid point #5 and NOT the one we want (the #6)... So that's a problem becase if we increment a few the ray distance we will get the hit point #3 (and not the #2 we wanted) and point #6. See this in the image:



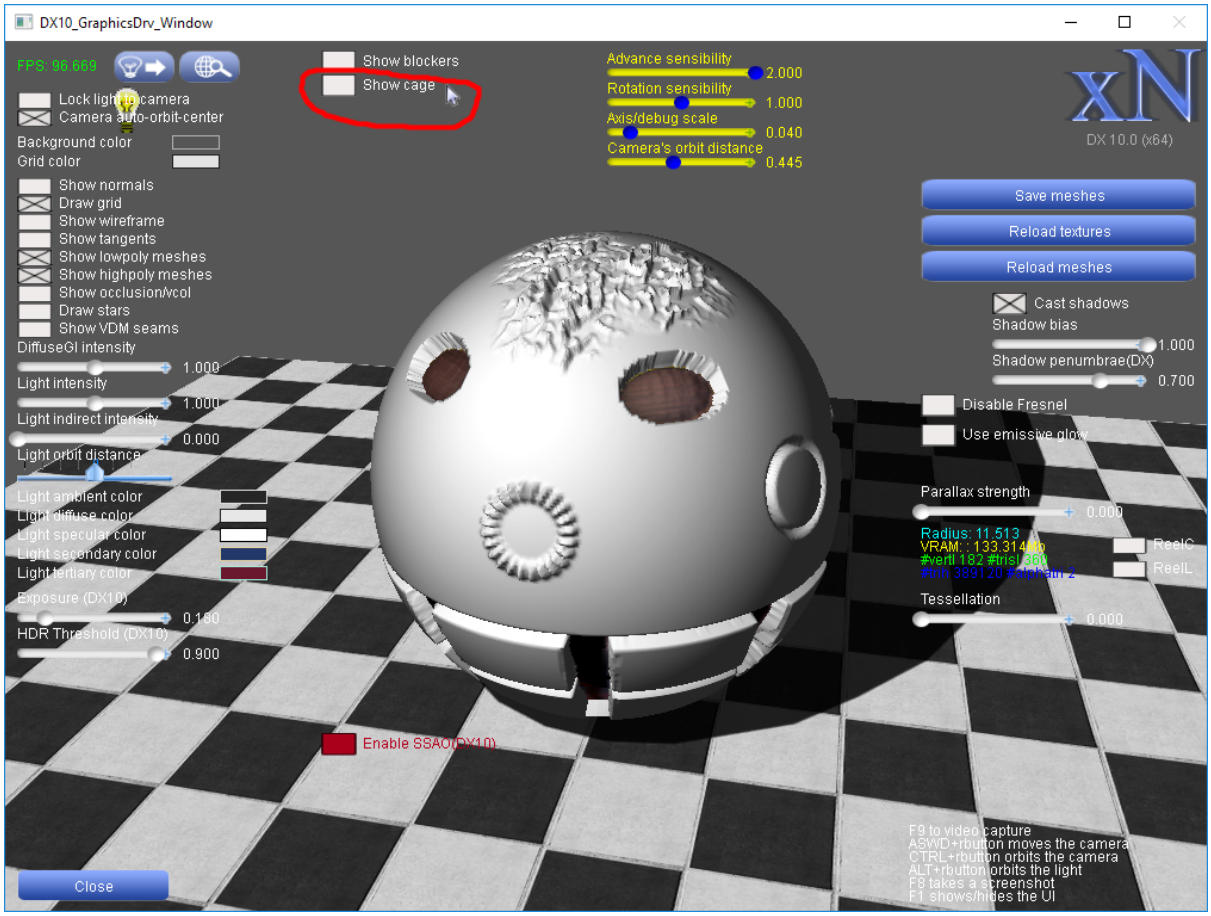
Too bad! We need other way! ... Constant (uniform) max ray distance is not always the solution, specially for very complex meshes like the image one....

xNormal allows you to specify “cages” to solve this problem... A “cage” is just a manual-designed polygonal mesh extrussion to specify the maximum ray cast distance in a non-contant or non-uniform way.... They work like “anti-portals” if you seen any 3D room engine (like Quake 3 one). They “stop” the ray at certain distance.

See the image:

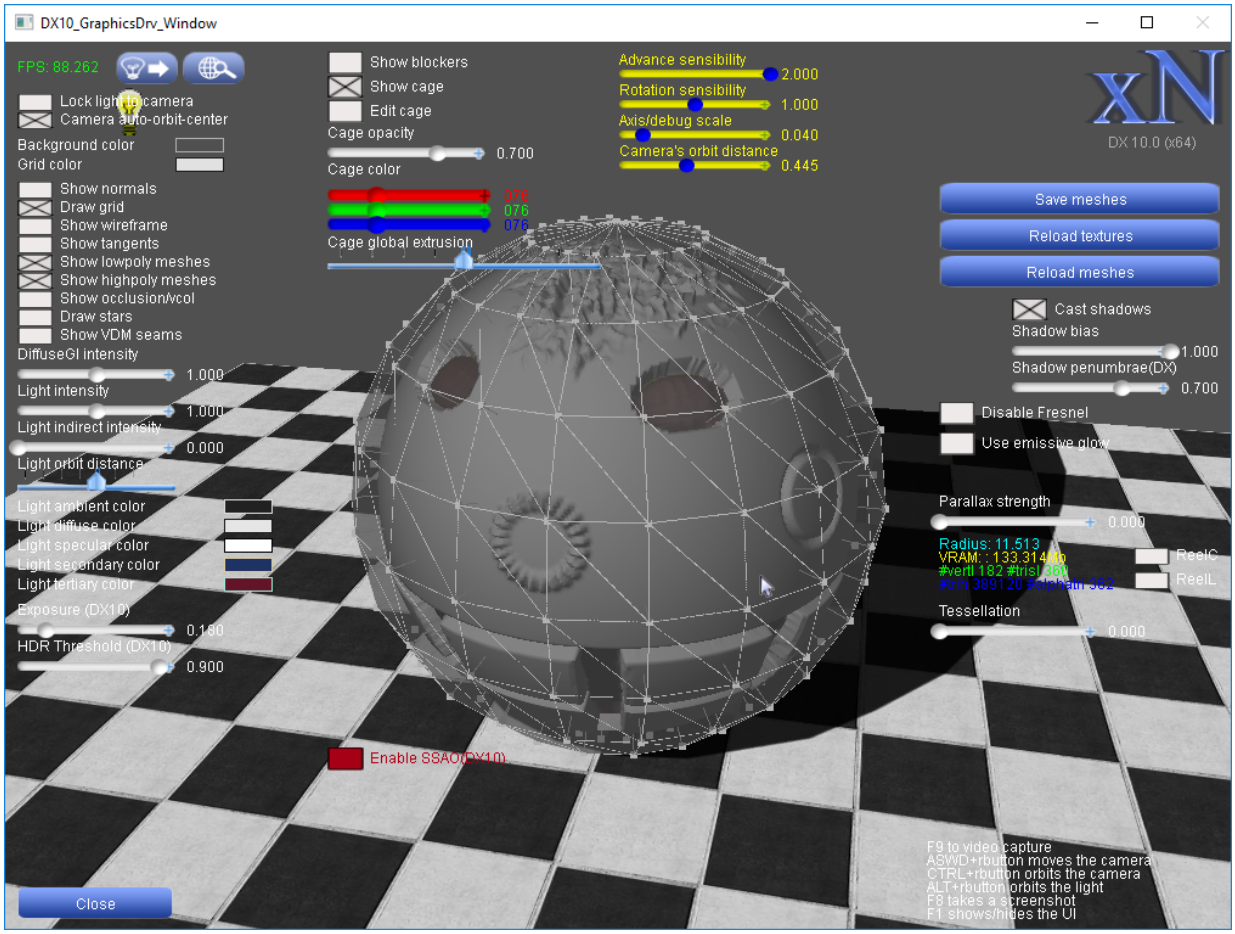


This needs a bit more of work because you need to set the cages manually. To do this, open the 3D viewer and check the “Edit cages”, some bars will appear then:



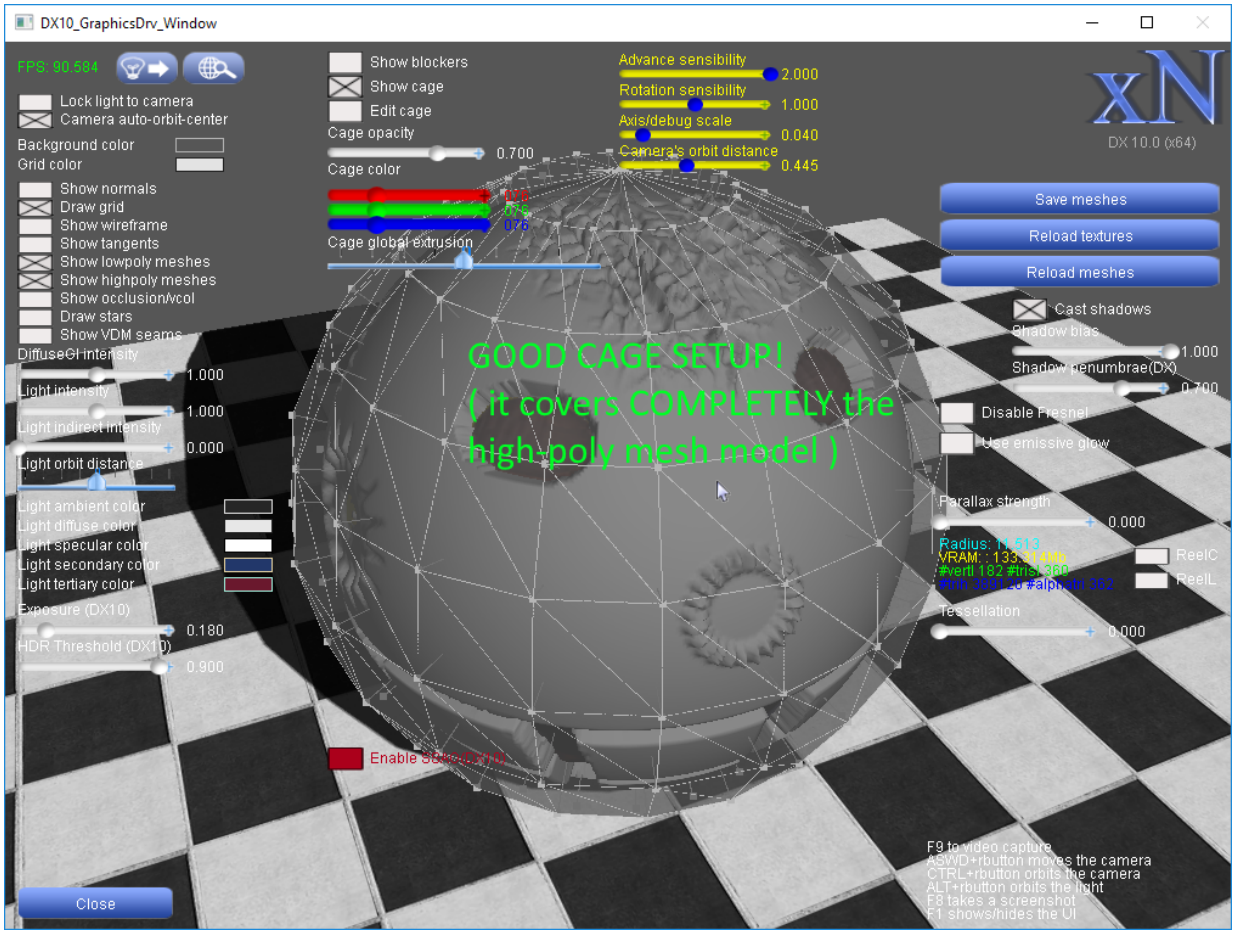
As you can see, you can control both front and back extrussion cages, color, opacity, etc.... Rays will be fired from the cage to the lowpoly model.

This is a typical aspect of the cage:



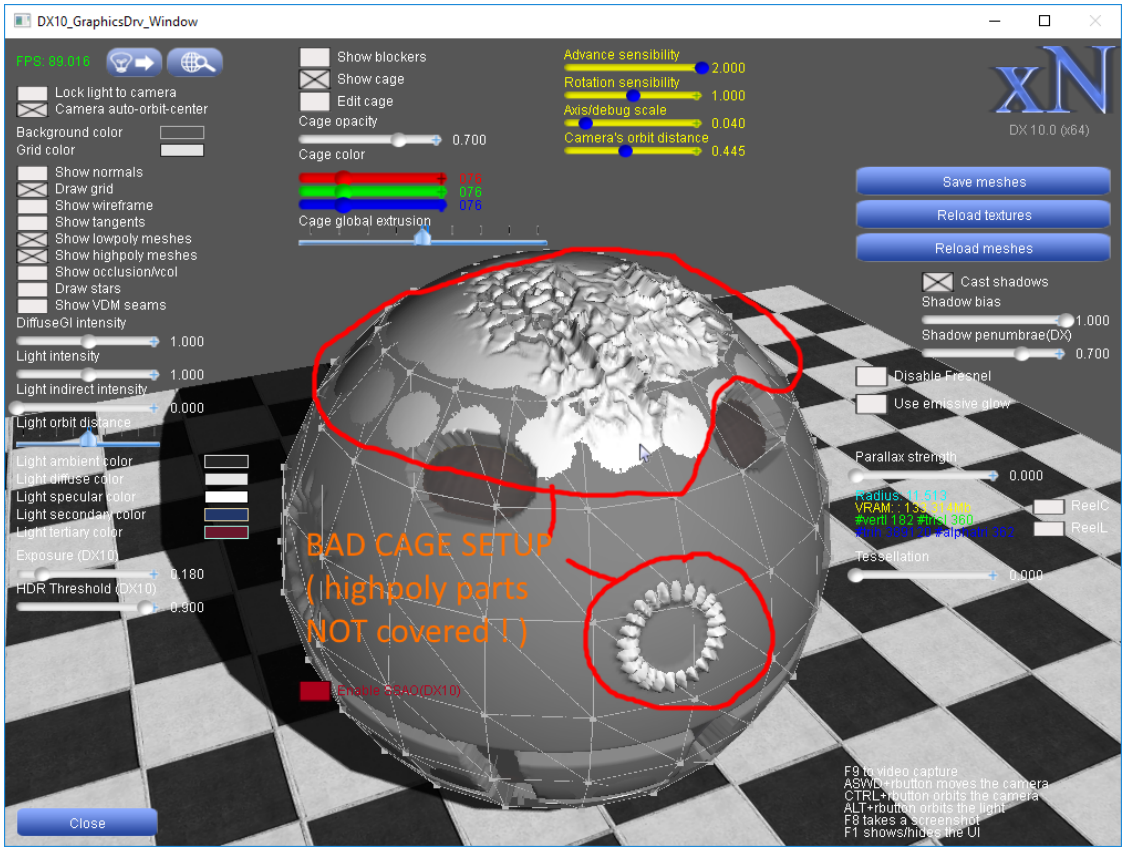
The front cage delimits the maximum ray distance PER VERTEX/FACE. Rays will stop when they reach the front cage.

Notice usually you WANT the front cage covering the entire high-poly model as you can see here:



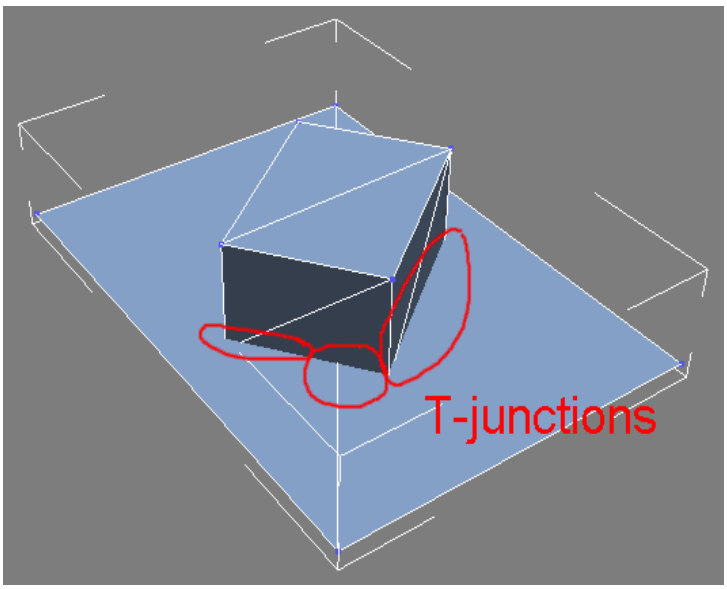
This is a WELL set cage, because the high-poly model is covered fully by the cage.

On the other hand, you could have something BAD like this:



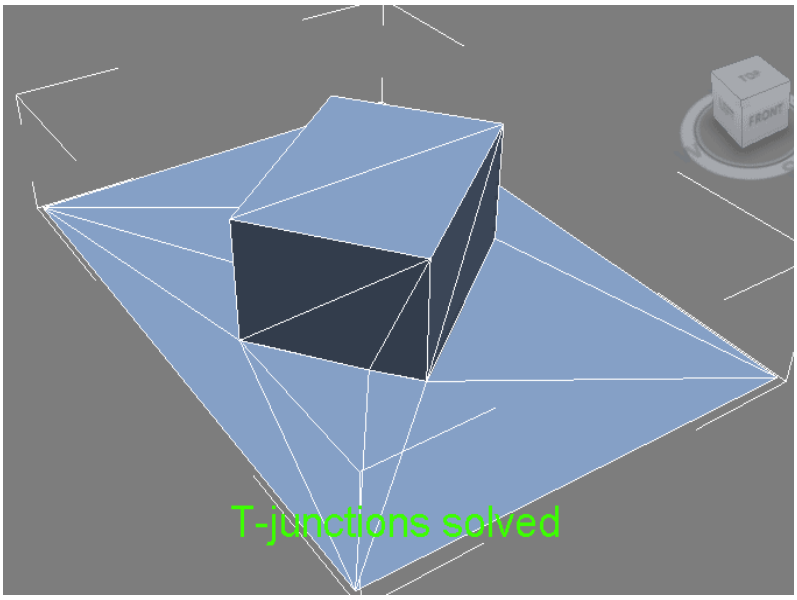
You can see this is a NOT good cage setup because some parts of the highpolygon model are not covered by the cage (in the image the ones NOT inside the red circle).

Other thing to consider to make a good cage is that the lowpoly mesh should NOT contain T-junctions:



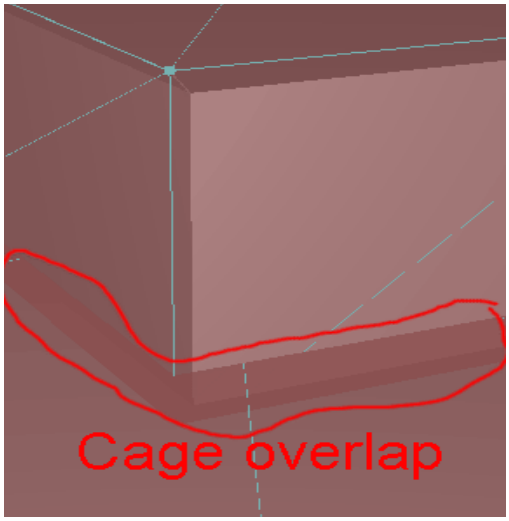
T-junctions will cause Zfighting and the cage's faces could overlap causing bad ray distance measurement! Avoid T-junctions always!

A way to solve the T-junctions is to tessellate a bit the surface, putting some extra vertices and re-making faces in this way:



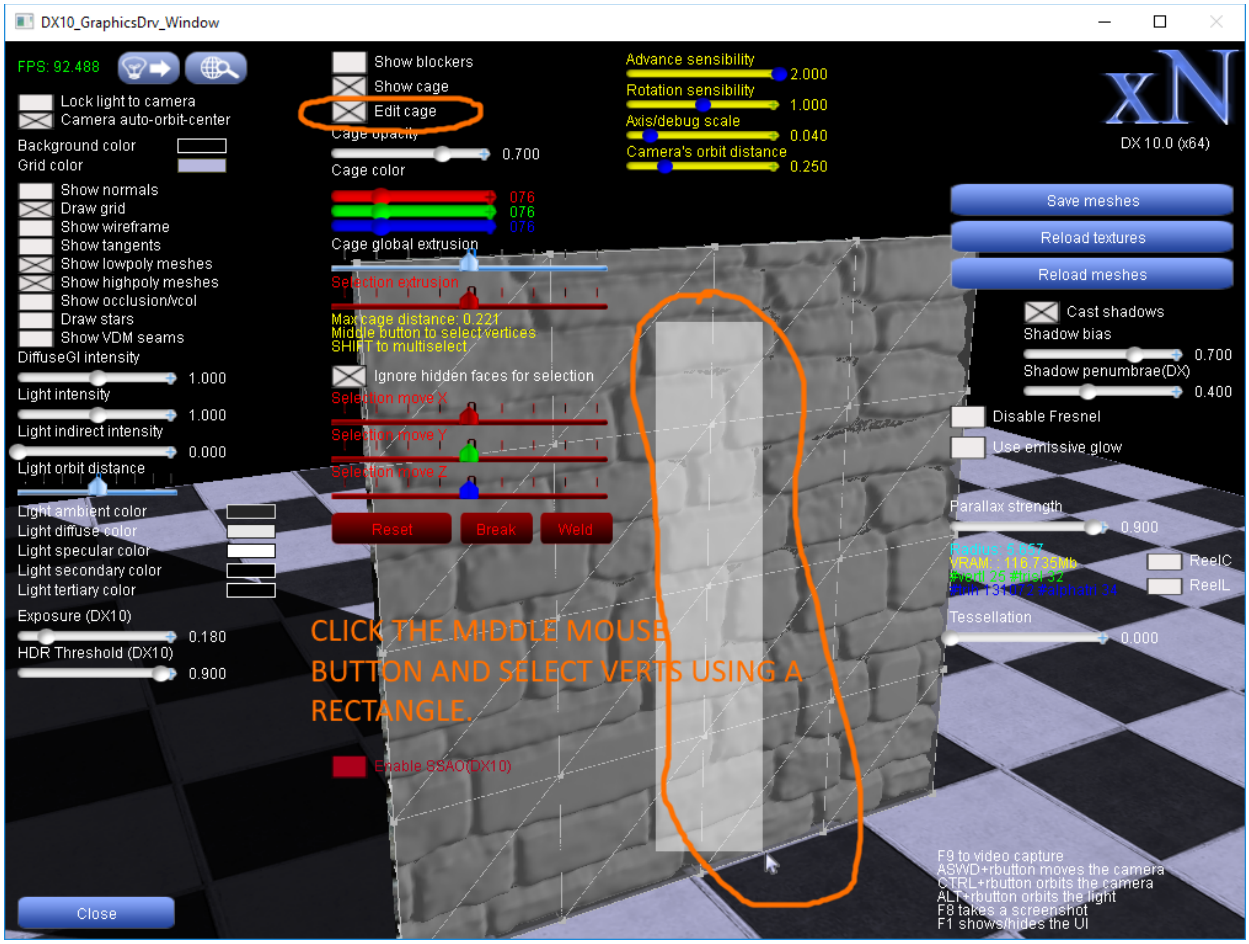
T-junctions solved tessellating the surface and re-making faces near the conflictive points.

T-junctions will make the rays to overlap on the cage...



which can lead to very bad artifacts in the normal map. On the other hand they can cause an ugly Z-fighting effect when using 3D engines with low precision depth buffers. Soooooooooo, an advice: remove as many T-junctions as you can from the lowpoly mesh or you'll get some ray distance artifacts due to cage overlap!

To edit the cage can be a bit tricky... Basically you can hide the lowpoly mesh unchecking the “Show lowpoly meshes” to edit it:



To select vertices just click over them or hold MIDDLE mouse button and move to define a 2D selection rect.

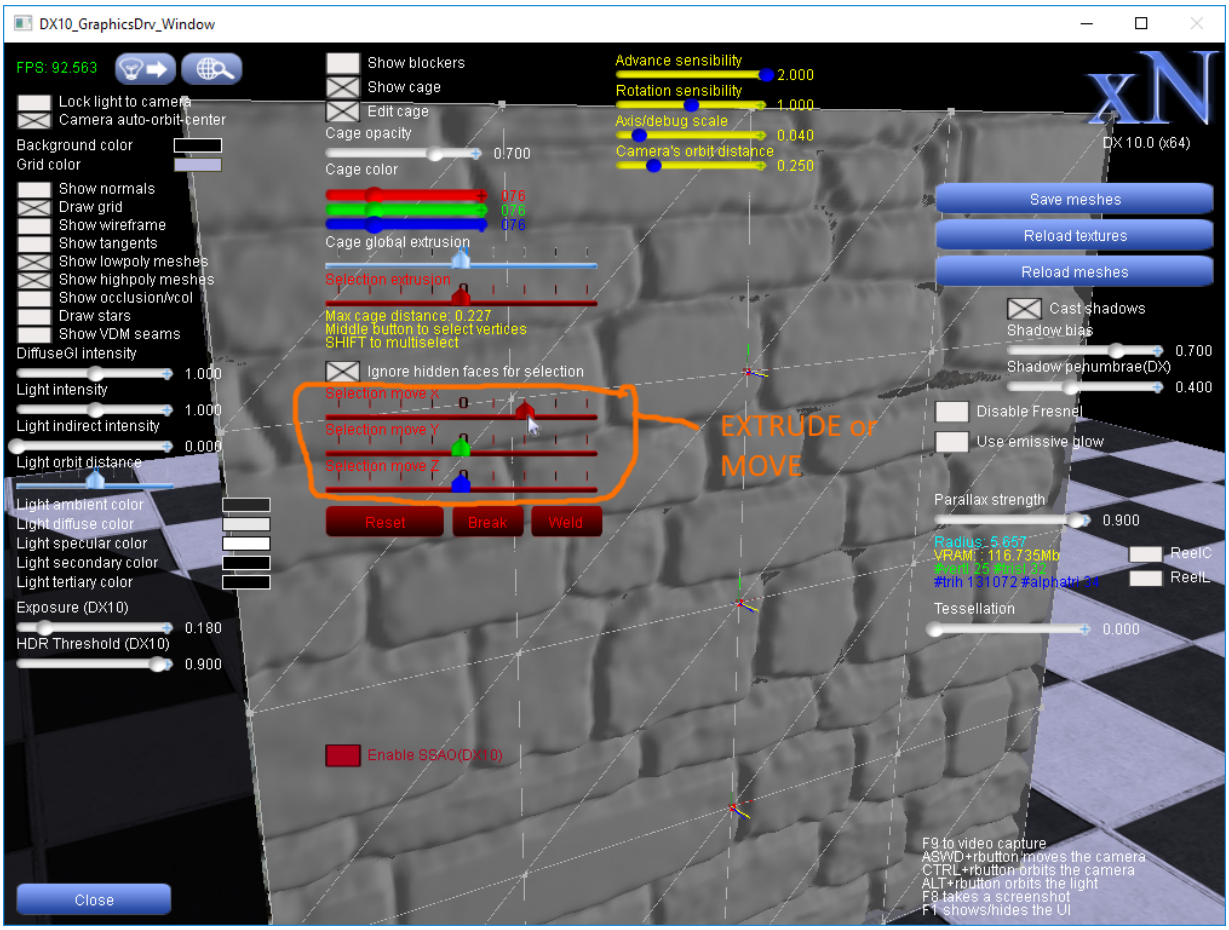
To add / remove vertices from/to the currect selection, use hold the SHIFT and click over the vertex.

Clicking over the empty space can clear the selection.

Notice too xNormal draws small lines to indicate the point-to-point extrussion (a line from the cage to the original lowpoly model vertex).

To avoid to select non-visible vertices you can check the “Ignore backface vertices”, see:

Once you selected the cage vertices you can “extrude” them along the vertex normal or move them free using the “move” sliders:

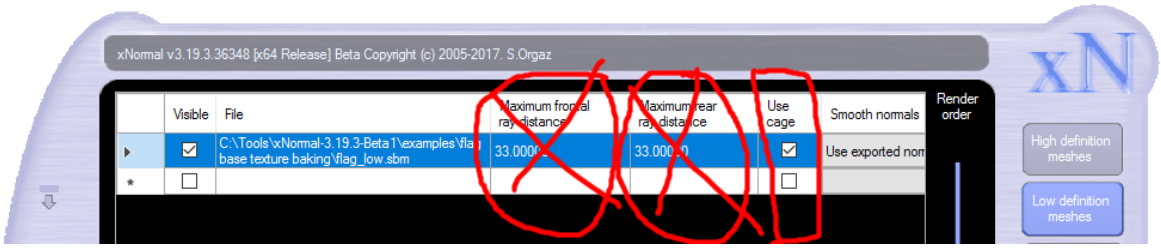


This allow you not only to control the ray distance with the extrussion... You can also tweak the normal direction of the rays! I highly recommend this tutorial about cages made by Ben Manthis :

http://www.poopinmymouth.com/tutorial/normal_workflow_2.htm

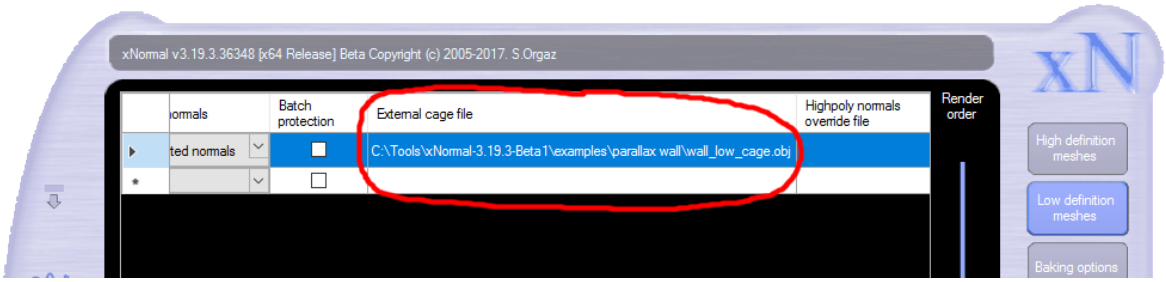
To save the cages click over “Save meshes” and select a format that allows to save cage info (for example, the OVB or the SBM format). If you use formats without “cage” data information (like 3DS, OBJ, etc...) you won't be able to restore the cages you edited... That's why you have to save the meshes in a format that allows to load/save cage data like, again, the OVB or SBM format.

To use cages for the normal map generation just check in the “use cage” option in the low-poly model tab, so they will ignore the front and back maximum uniform ray distance and use the cage you created. See the image:

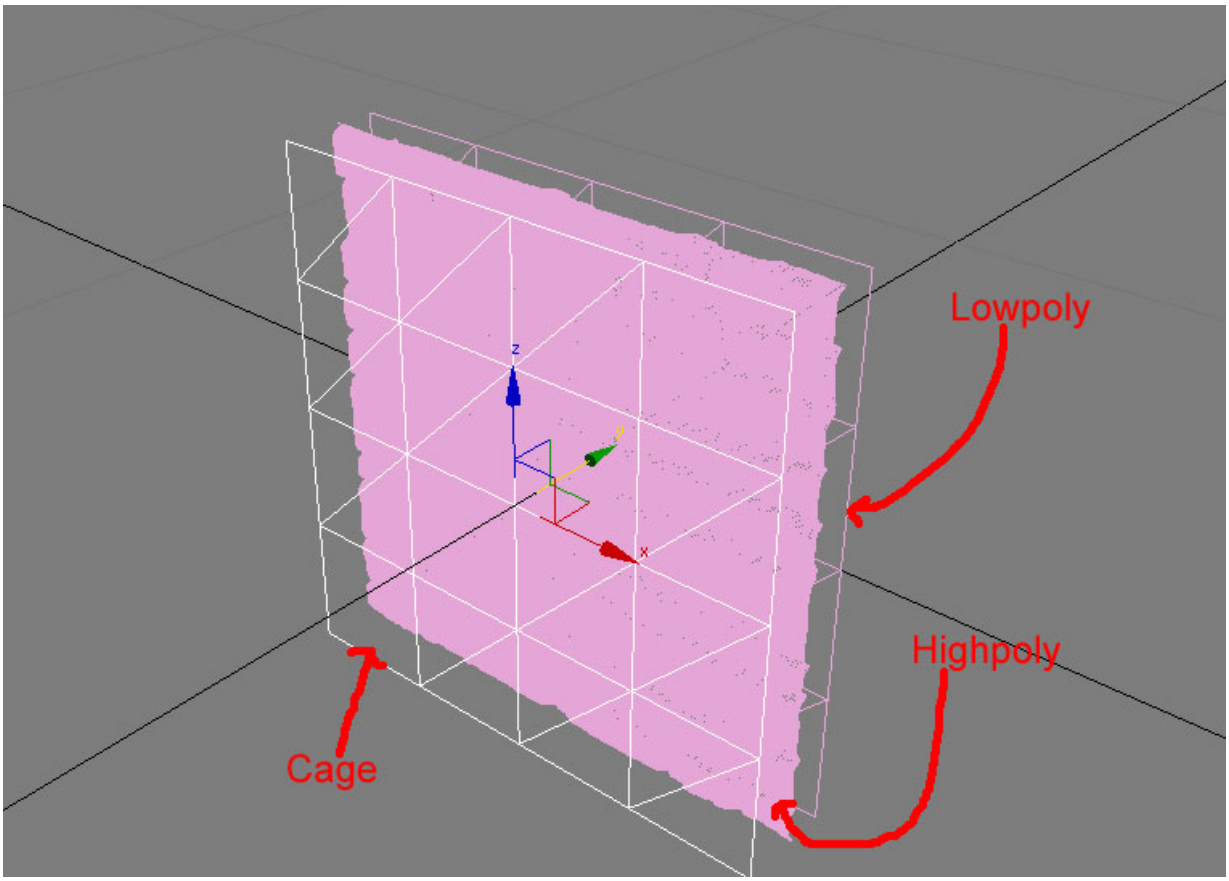


Notice only a few mesh formats can load/store cage information (for example, the OVB). If you try to use cages with, for example, a format like the OBJ or the 3DS which don't contain cage info, a default value (0.0f) will be used for cages! So sure if you check the “use cage” option you selected a format that contains cage vertex extrussion data (again, like the OVB or the SBM).

However, xNormal allows you to use a external-defined cages in any supported mesh format. This is very nice if you don't like the xNormal built-in cage editor. For example, is you see the “parallax wall” example that comes with the application you will see that we are using the “External cage file” option in the lowpoly options tab:



This file can be created using your favourite 3D modelling program:



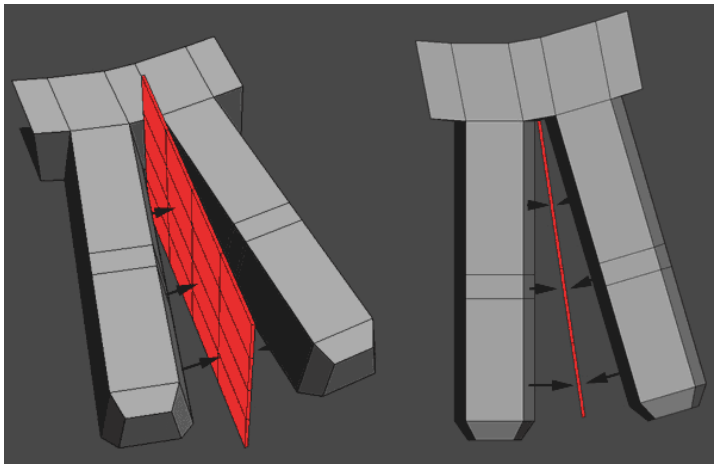
But notice a thing... If you export as, for example, a OBJ the mesh cage, YOU MUST BE ABSOLUTELY SURE THAT THE LOWPOLY AND CAGE TOPOLOGY ARE THE SAME, because xNormal requires a 100% vertex index match between the lowpoly mesh and the cage mesh topology!!! So caution!!! If you edit a vertex in the lowpoly mesh, YOU WILL NEED TO RE-DO/RE-EXPORT THE CAGE COMPLETELY!!!!!

xNormal includes some exporters for very popular modelling programs too!

Usually, the uniform-constant max ray distance is more than enough... but if you have to treat with very complex geometry then use cages. Use the xNormal built-in editor, a 3rd-party application to create your cages or the included SBM mesh exporter.

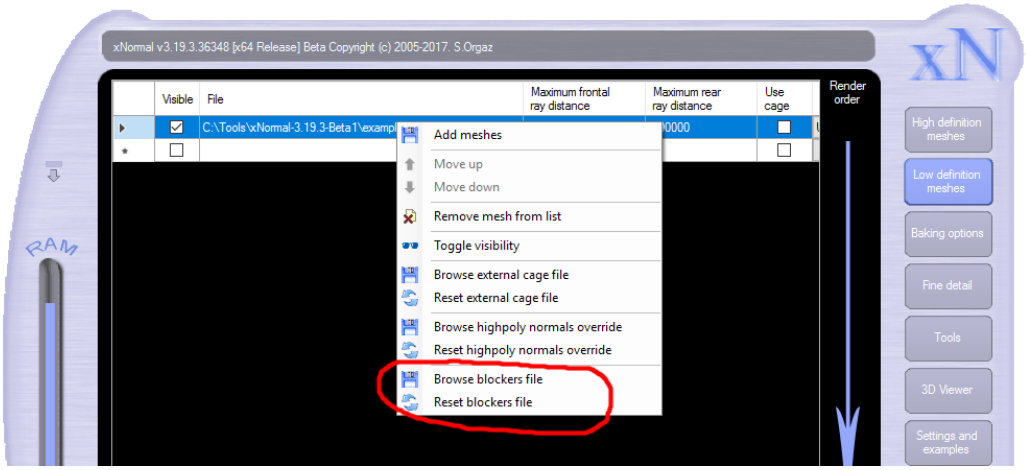
2.4.1.6 Using ray blockers

This is other mechanism to limit ray distances. It was introcuced in xNormal 3.16.2. You can place manually some polygons to block the rays in the same way a scene designer places anti-portals:



Rays will be fired from the lowpoly mesh following its normal. If a blocker is found in the trajectory then the ray will be cut. In that way you can avoid the rays to hit other parts of the model.

To use this feature you just need to export separately these blockers (for example, to a .OBJ or .SBM file) and then assign the file to the corresponding lowpoly slot using the “Browse ray blocker file” from the context menu (in a very similar way that you do for the external cages):

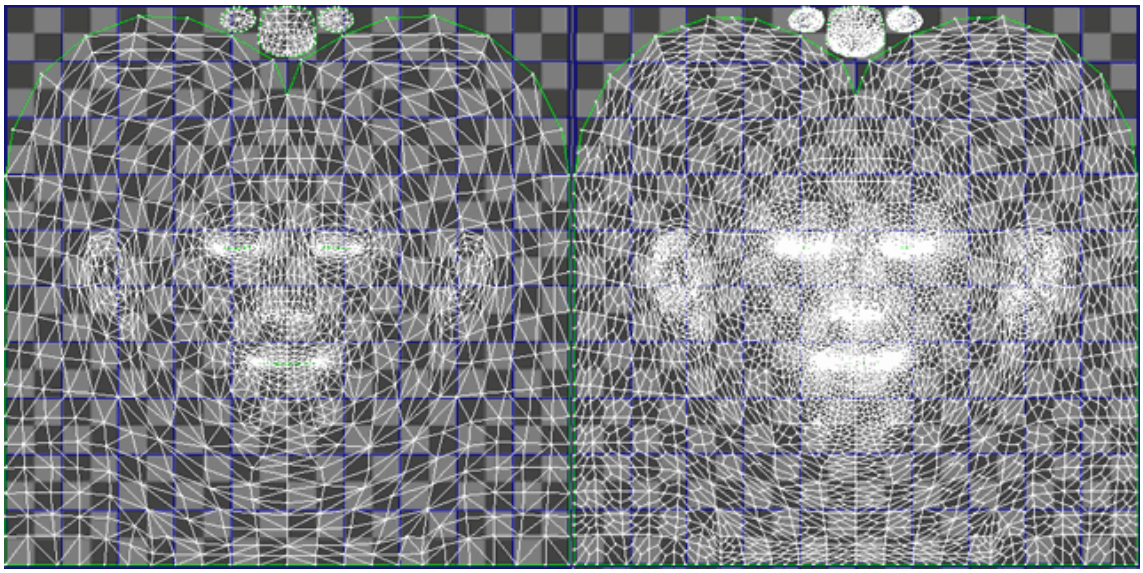


However, cages are preferred because the blocker mechanism requires more ray tracing work, so they can be slower than the uniform ray distances or the cages method. I recommend you to keep the number of blockes as low as you can (like 10 or 12 per object as maximum).... and try always to use simple polygons planes (there's no need to tessellate them).

And a reminder... xNormal uses always a furthest hit policy to allow floating geometry... so keep this in mind when you setup your blockers!

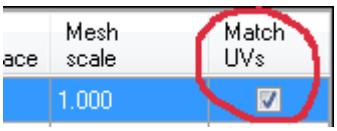
2.4.1.7 Using the MatchUVs feature to render the maps easier

If you made the highpoly mesh just subdividing and sculping the lowpoly mesh, this option may be interesting for you. Subdivided models have an interesting property: their UVs are always contained inside the original control cage mesh's ones. See these images:



The left image corresponds to a mesh without subdivision. Using this mesh as control cage, we subdivided it one time. The image on the right corresponds to a the subdivision level 1. As you can see, the subdivided Uvs are contained always by the lowpoly UV's seam borders. That's good because it's possible to get any highpoly attribute very easily.

You can enable this option in the corresponding lowpoly mesh slot



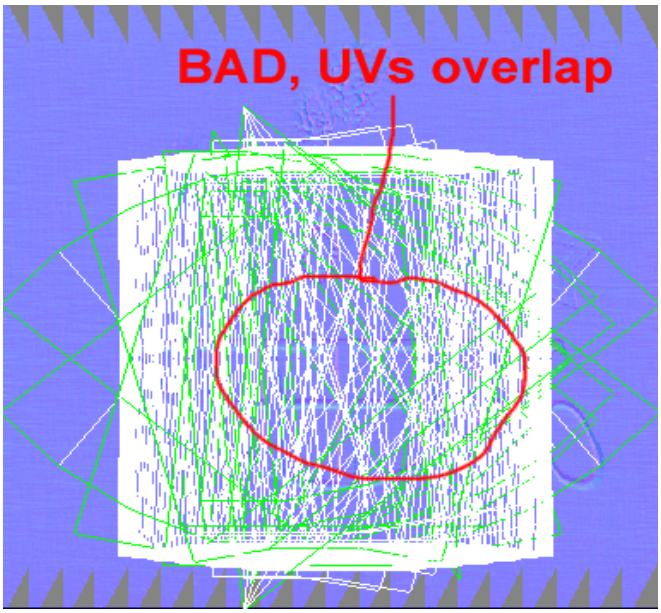
, so xNormal can compute the ray's hits faster and you won't need to setup a cage nor ray distances. For example, the normal maps will be computed automatically without needing to setup any distance! However, the occlusion maps still require distances to be set.

2.4.2 Problems with T-junctions and overlapping texture coordinates

As recomendation, you should plan very well how the low polygon model is texture mapped. Use pixel snap , uniform mapping and avoid T-junctions or “overlapped” triangles if possible:

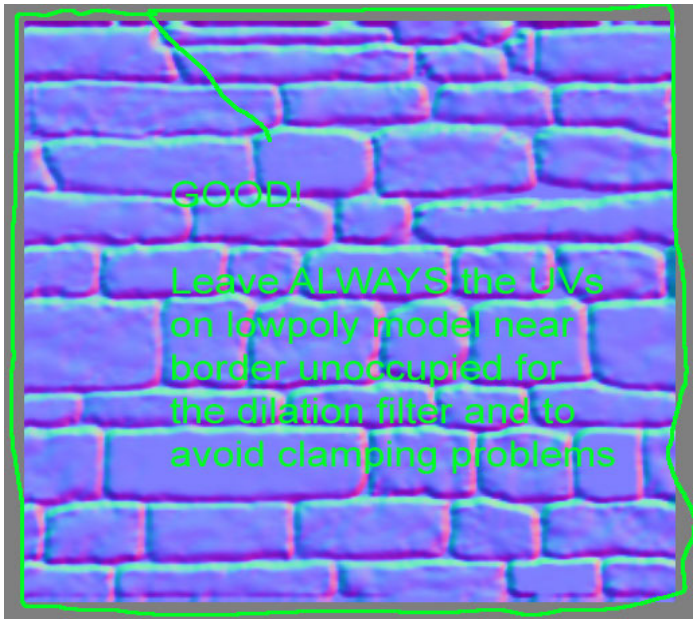
Pixel-snap and UV welding is a good idea too. If two UVs are different but too close can degenerate in visual artifacts in the final normal map.

Also, don't overlap UVs. Check out this image !



All the triangles must have an unique, unocuppied-by-other-faces space in the UVs. In the image, we can see the back part of the face invades a region of the front chest. In this case, an undesired effect will appear when the programmer does the DOT3 bump map lighting. These situations should be avoided at any cost.

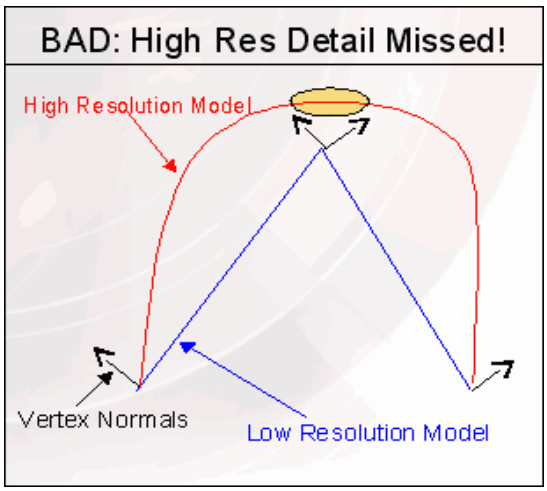
Another good practice is to leave the texels near the borders VOID and let the xNormal image dilation filter to propagate some pixels there to avoid seams. See this image:



So try always to leave the lowpoly UVs near 0.0f or 1.0f border unocuppied.

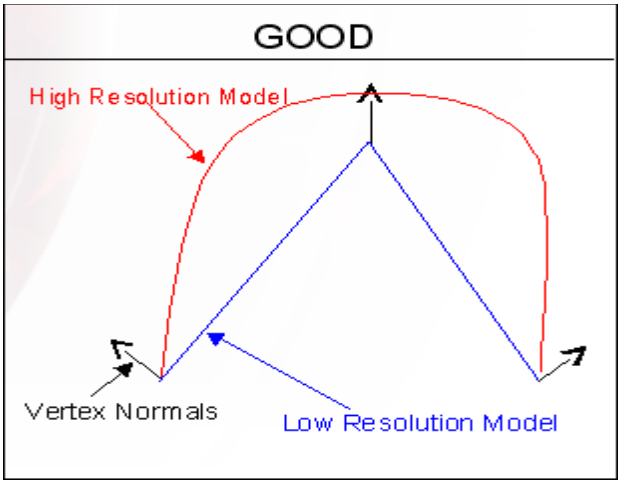
2.4.3 Problems with vertex normals

Other thing to take into consideration are the vertex normals. See this image:



In the image you can see having more than one normal per vertex can cause HARD edges seams and miss highpoly detail. On a normal continuous-mapped model this should be avoided... Under some circustances is ok to specify more than one normal per vertex, for example, for a cube which has hard edges.

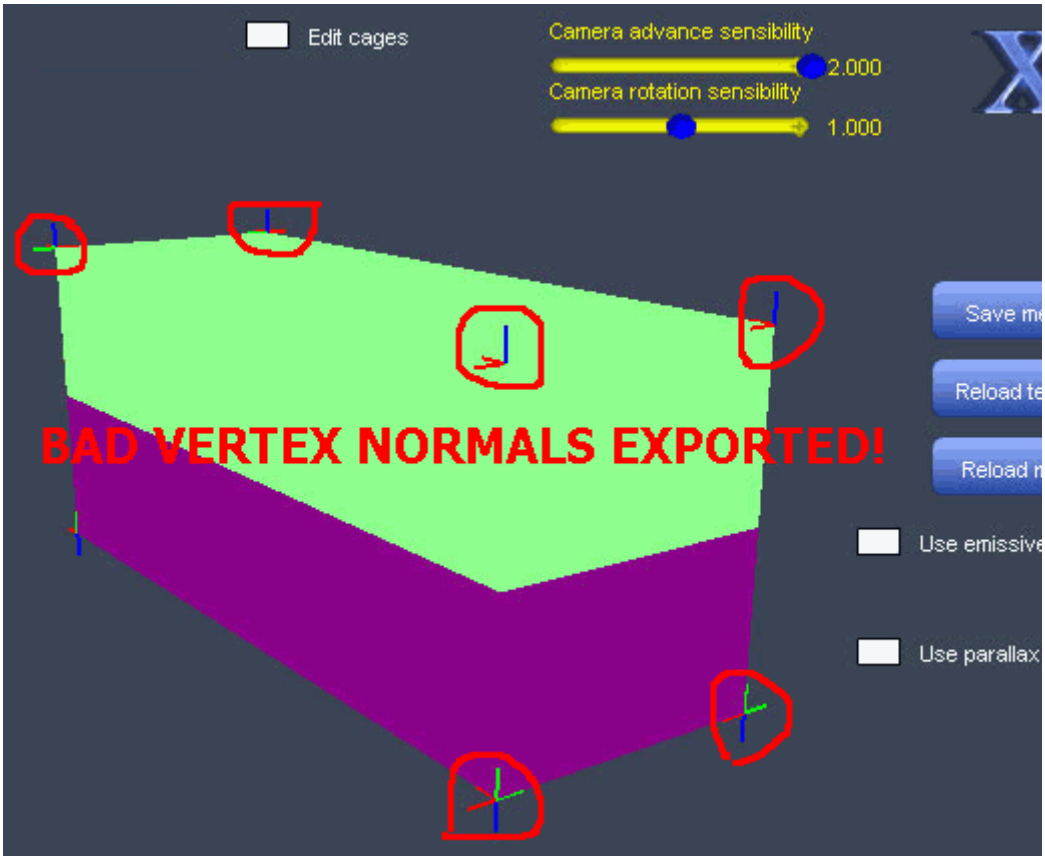
If you aren't working with cubes and you are have a continuous-mapped objects, then you should use only one normal by vertex as:



so you won't skip details in the highpoly model and you will get a smooth and good normal map.

As you can see, setting correctly the vertex normals in the lowpoly model is critical. However, you can let xNormal to calculate smooth normals to avoid all these problems (see the Smoothing Normals mode above).

Another problem can appear using immature and bad mesh exporters. For example, this is what I got exporting from an old 3D modelling program to an .OBJ:



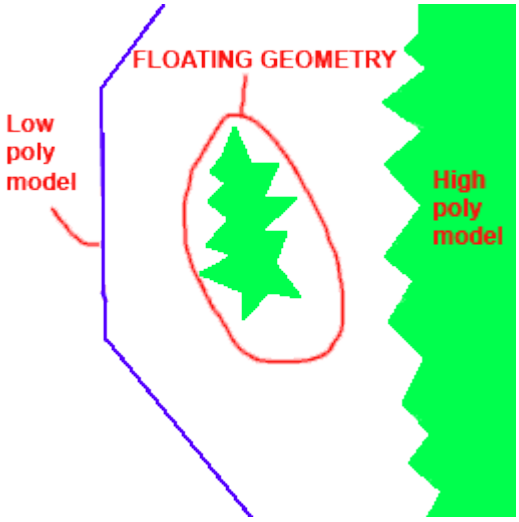
As you can see, the vertex normal are completely bad exported for that simple cube. Editing the .OBJ manually with the Notepad exposes the problem:

```
v  -2.500 0.000 2.500
v  2.500 0.000 2.500
v  -2.500 0.000 -2.500
v  2.500 0.000 -2.500
v  -2.500 5.000 2.500
v  2.500 5.000 2.500
v  -2.500 5.000 -2.500
v  2.500 5.000 -2.500
# 8 vertices

vt  0.000 0.000 0.000
vt  1.000 0.000 0.000
vt  0.000 1.000 0.000
vt  1.000 1.000 0.000
vt  0.000 0.000 0.000
vt  1.000 0.000 0.000
vt  0.000 1.000 0.000
vt  1.000 1.000 0.000
vt  0.000 0.000 0.000
vt  1.000 0.000 0.000
vt  0.000 1.000 0.000
vt  1.000 1.000 0.000
# 12 texture vertices

vn  0.000 -1.571 0.000
vn  0.000 -1.571 0.000
vn  0.000 -1.571 0.000
vn  0.000 -1.571 0.000
vn  0.000 1.571 0.000
vn  0.000 1.571 0.000
vn  0.000 1.571 0.000
vn  0.000 1.571 0.000
# 8 vertex normals

g Box01
f 4/11/4 2/9/2 1/10/1 3/12/3
f 8/12/8 7/11/7 5/9/5 6/10/6
f 6/8/6 5/7/5 1/5/1 2/6/2
f 8/4/8 6/3/6 2/1/2 4/2/4
f 7/8/7 8/7/8 4/5/4 3/6/3
f 5/4/5 7/3/7 3/1/3 1/2/1
# 6 faces
```

As you can see the vertex normals are incorrect. There are duplicated, unnormalized and and we can find only to-bottom and to-up ones. This is VERY common... don't trust in exporters, they can be evil sometimes...

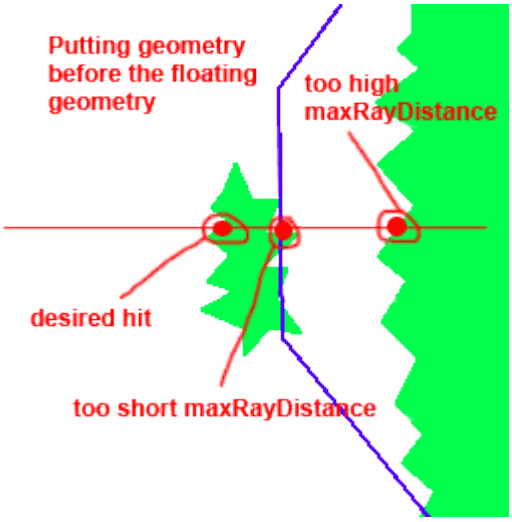
2.4.4 Problems with floating geometry

Another thing to take into consideration is the “floating geometry”. This kind of geometry is, for example, the cloth and armor a model can use. See this picture:



As you can see, there is a small part in the high polygon model that “floats” into the air.

xNormal ALWAYS uses a ray-hit-farthest-double sided approach. So, can appear certain problems with this kind of geometry. You must be cautious setting the “Maximum Ray Distance” in the “Normal map” tab:

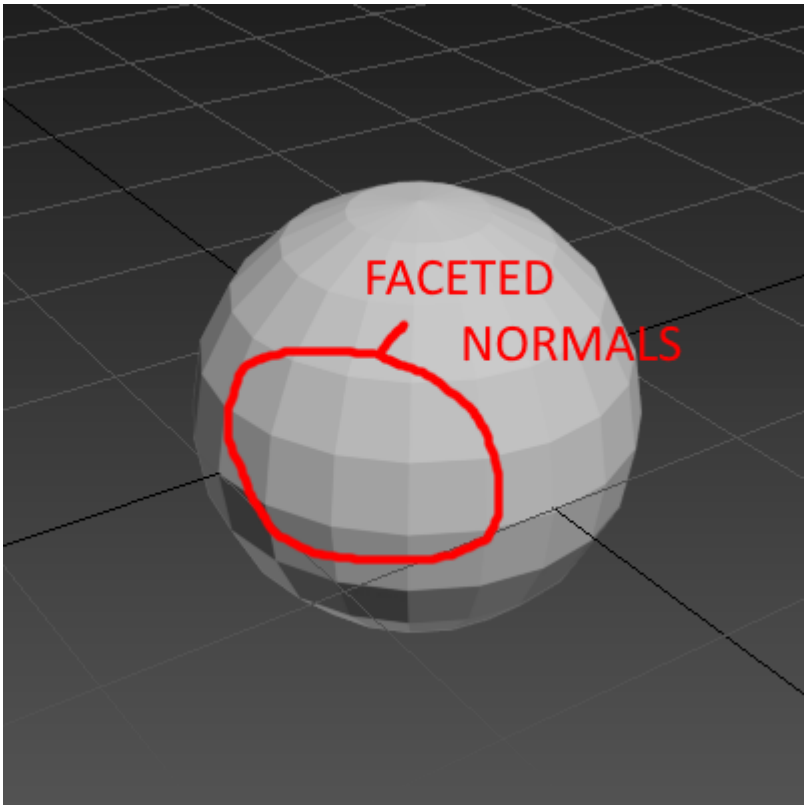


So, as you can see, setting well the “Maximum Ray Distance” is CRITICAL for “floating geometry”.
If you need non-uniform ray distance then see the “cages method” in the sections above.

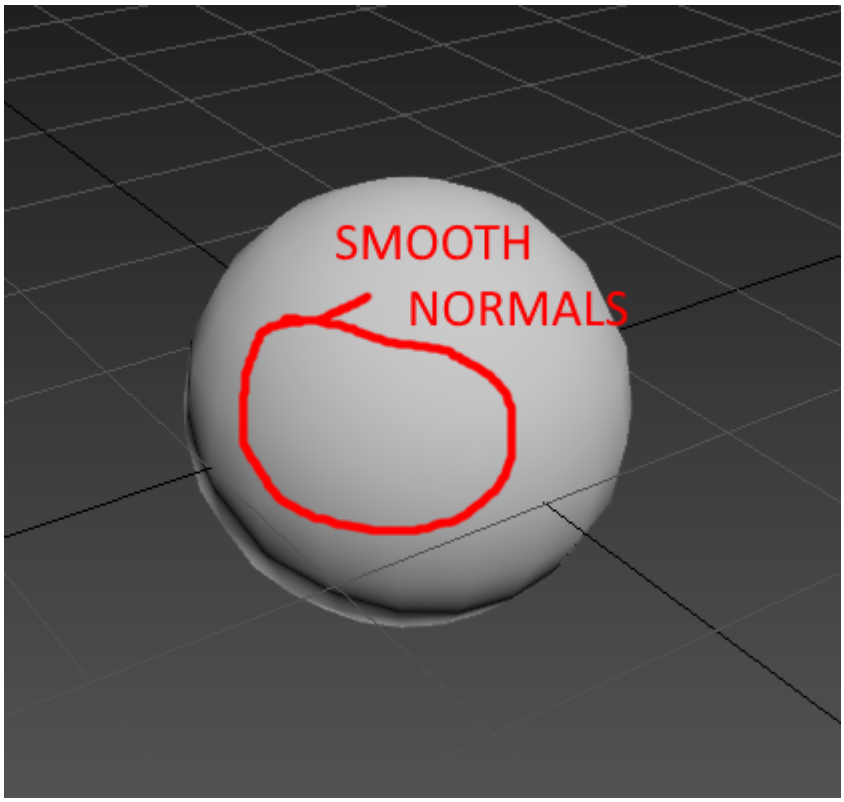
2.5 Smooth normals

You should assign very carefully the smoothing groups for your meshes.

For example.... Imagine you exported the following faceted model :



These normals could cause severe seams and lighting artifacts in the resulting normal map. For complex & organic meshes, we suggest you to set the normals as this:

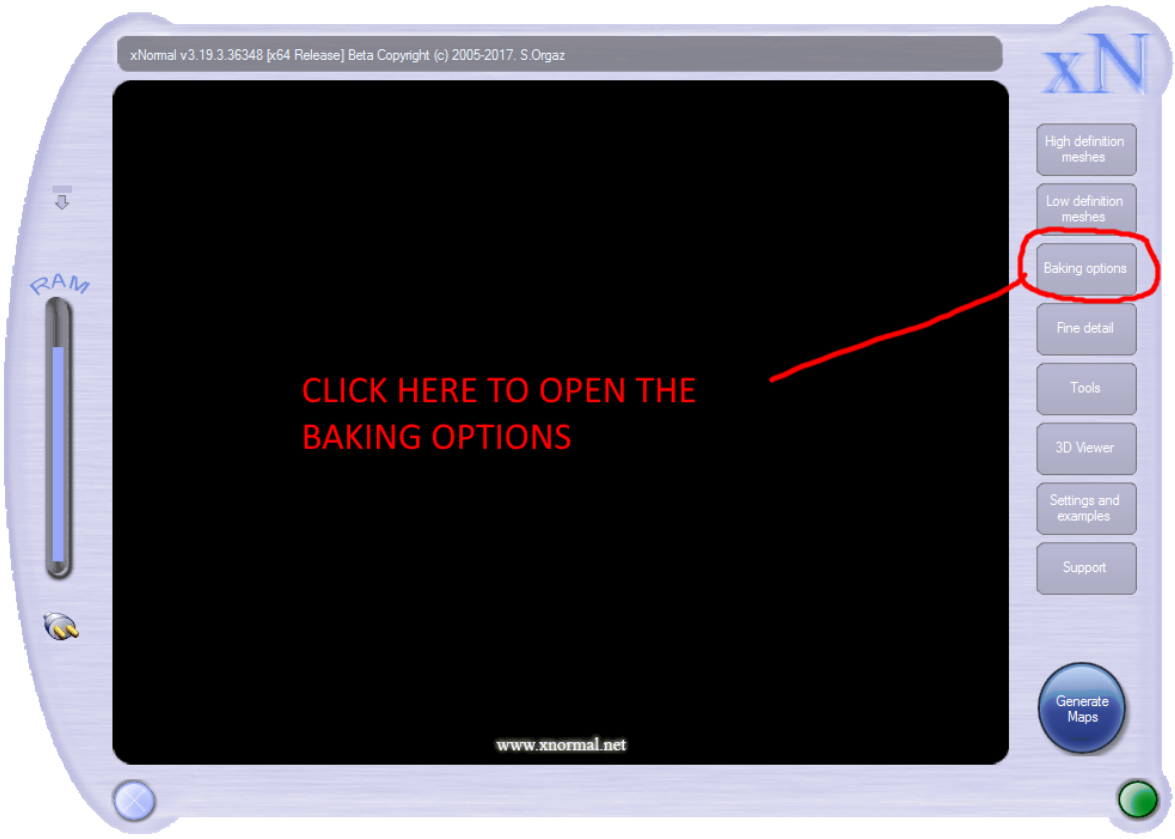


xNormal can automatically average the normals, so you won't need to export the normals for the high polygon model and neither in the low-poly one.

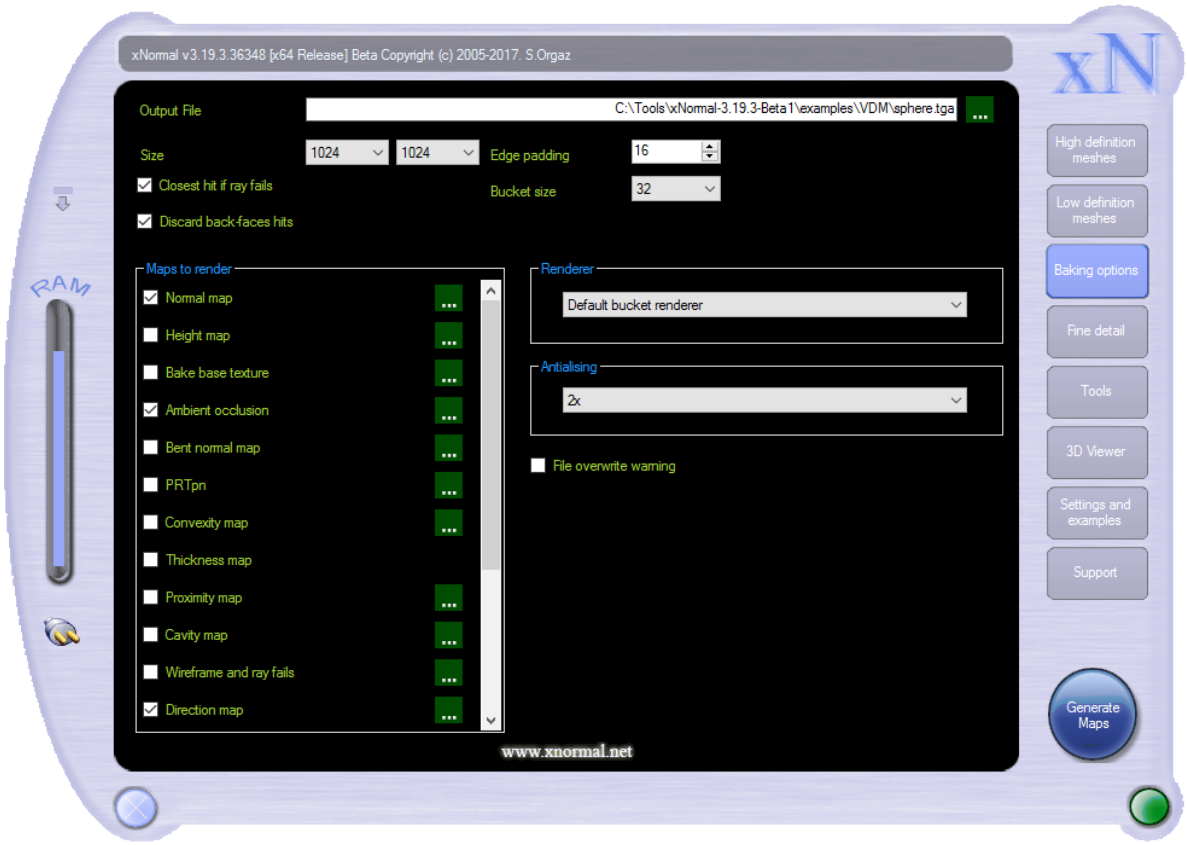
If you set the option “Use exported normals” in xNormal, then the program will load your custom-defined normals from the mesh file.

2.6 Baking options

Go and click on the “Baking options” tab:



This will show the output map generation options:



Clicking in the top-right  button you can select the output filename where to save the result normal map.

See the section **1.6** about the supported normal map image file formats.

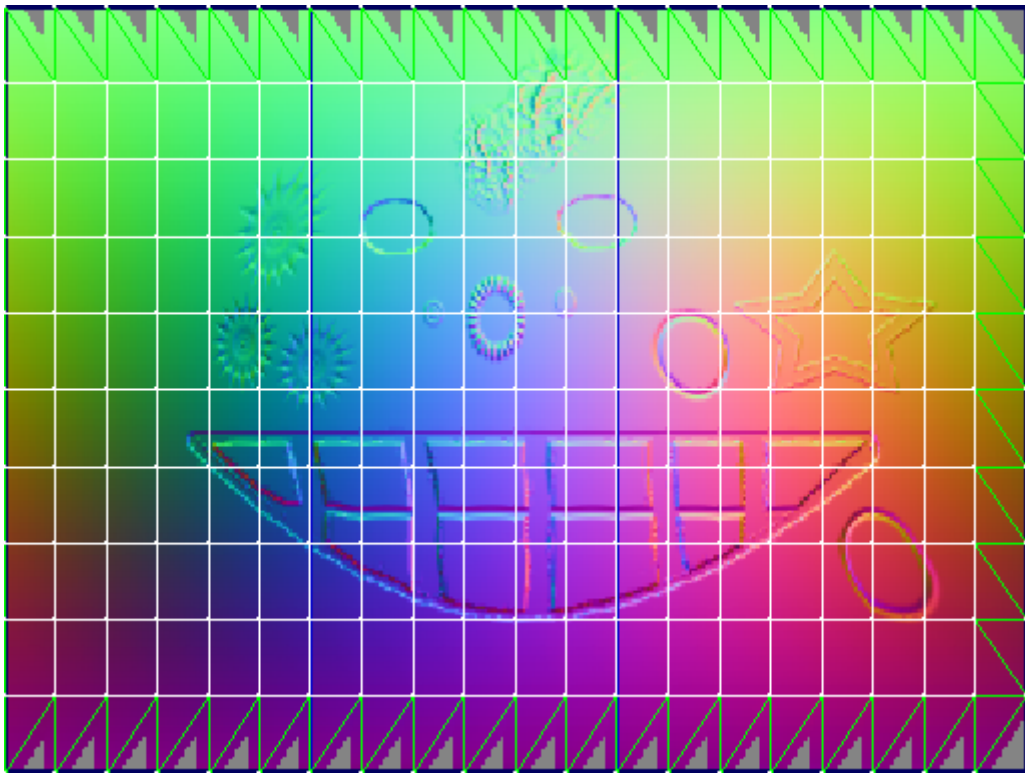
Take into consideration that some formats perform lossy compression (like the JPG), so won't be good if you need image quality results.

Notice there are multiple choices in the “Maps to render”. You must select the kind of map to render and you can control the specific map options pressing over the “...” button.

Now lets see all the other parameters in detail:

2.6.1 Object / Tangent space

Object space[3] normal maps look like:



Object-space normal map with the UV mesh as overlay. Notice you can usually see almost all the RGB spectre in these kind of normal maps.

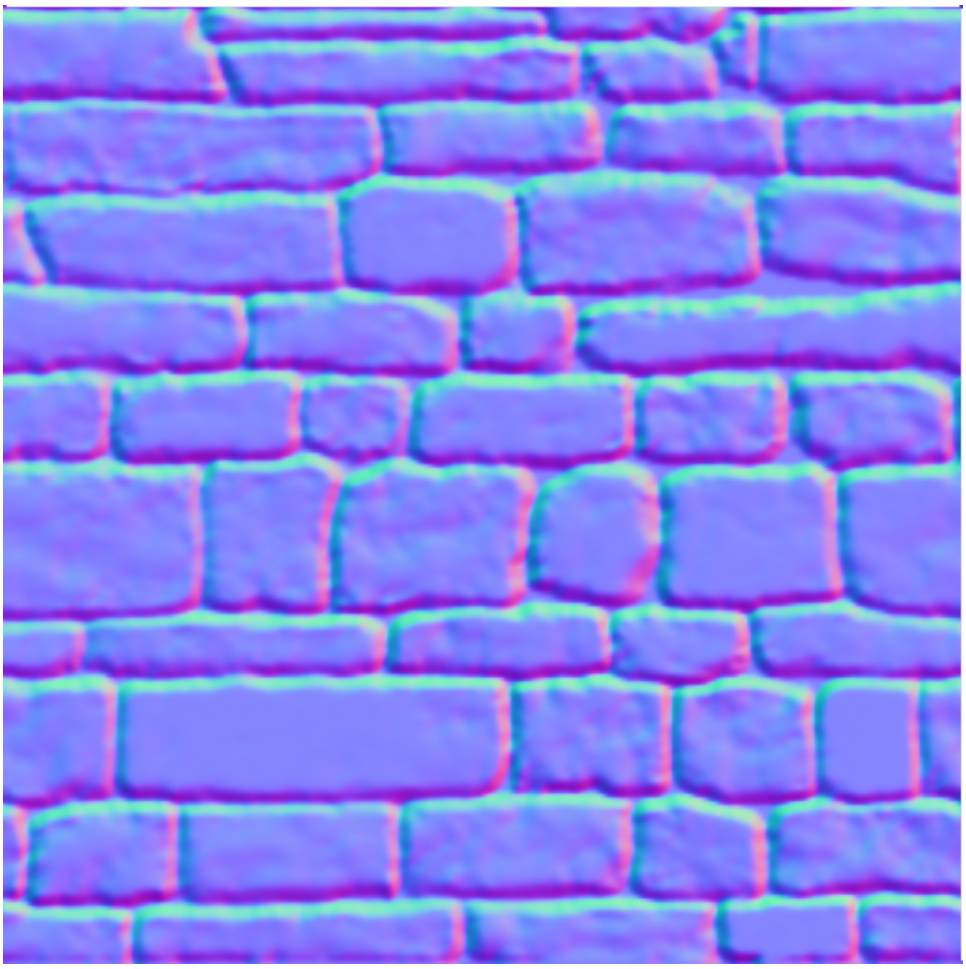
As you can see, is good because the model uses uniform UVs, is pixel snapped (even you can't notice this in the image) and has no T-junctions neither overlapped UVs.

Usually you can see red, green and blue components by equal quantity. Red is the X component, green the Y component and blue the Z component.

Object space normal maps are usually very good for non-deformable objects.

3 **Object space normal map**. The normals are stored in the local space of the mesh using the (X,Y,Z) euclidean coordinates. Red color indicates the X component in the normal is strong, green color indicates the Y component in the normal is strong, blue component indicates the Z component in the normal is strong.

On the other hand, we have the tangent space[4] normal maps:



Tangent-space normal map. Notice these kind of normal maps are usually “blue”

Tangent space normal maps are good for meshes those deform themselves, to achieve parallax bump mapping[5] or if you need to mirror/tile the normalmap.

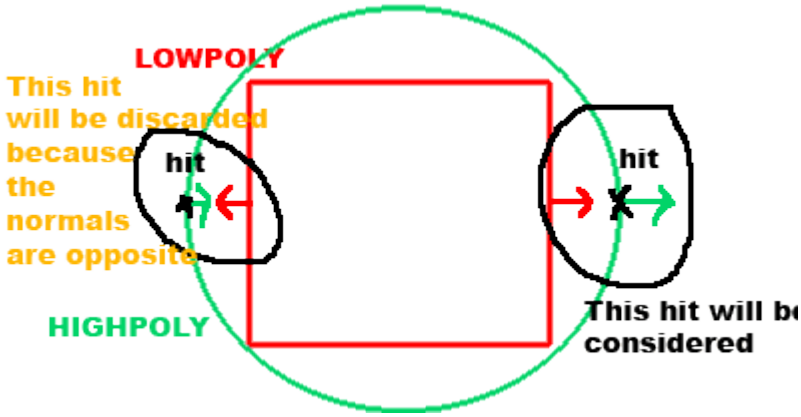
Technically, the tangent space defines a coordinate system for each face (and using averaging, can be extended to each vertex), putting the Z axis as the normal. After we calculate this, we can pass the light to that space and DOT3 the light with the texture normal map value.

4 **Tangent space normal map**. When you need to use deformable meshes (like skinned ones), you can't store the normals in object space because they vary. You need to stablish a local space for each vertex, using the Z component up, X increasing with the U (or S if you use openGL) component and Y increasing with the V (or T if you use openGL) component. As result, this kind of normal maps are typically “blue”.

5 **Parallax bump mapping**. Also known as “offset bump mapping”. Traditional bump mapping looks bad if you are not perpendicular to the surface... The “parallax” one solves this using the “height or displacement map” that xNormal can generate in the alpha channel of the generated normal map. Ask your 3D programmer about it!

2.6.2 Discard back face hits

This option will make the raycaster to discard any hit if the lowpoly normal is opposite to the highpoly surface hit normal. To understand better this look at the following image:



So, as you can see, if the lowpoly normal and the hit highpoly surface normal are opposite the ray hit will be ignored and xNormal will continue searching for valid highpoly hits.

Checking this option you can avoid some “reverse” normals problems. Our advice is to keep it always enabled.

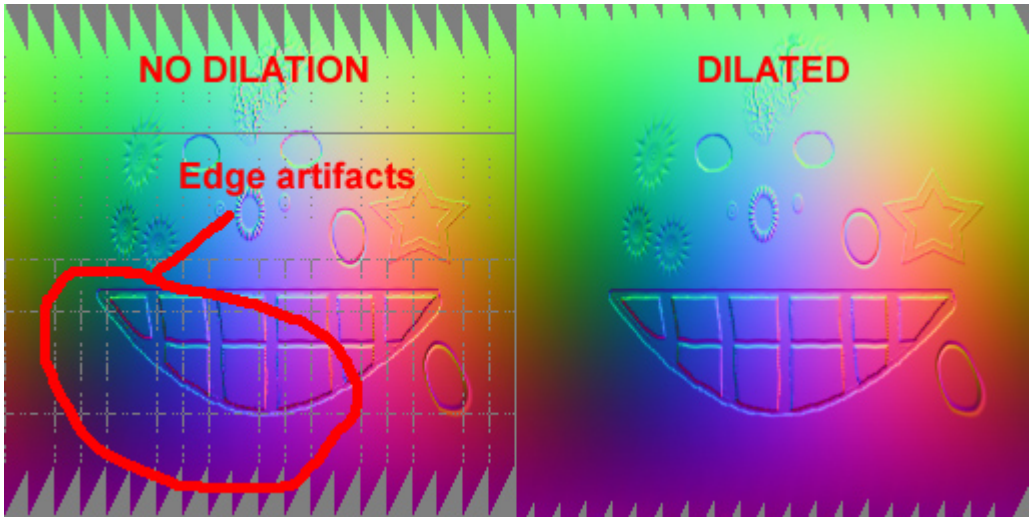
2.6.3 Edge padding

Again, to solve edge artifacts and also mipmapping problems, you can use the “edge padding option”.

The default is six pixels and usually will be fine. For greater than 2048 maps perhaps you should increase it to eight.

To disable edge padding just set it to ZERO.

See how affects this option to the image rendered:

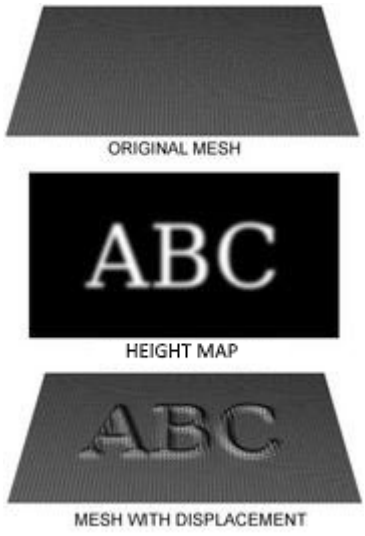


As you can see, “sand” unwritten pixels across edges are a massive artifact ... You should never disable this option. Keep it enabled. The default is six pixels and is an appropiate value for almost all cases.

2.6.4 Generate height map

xNormal can save the highpoly mesh distance to the lowpoly mesh in a per-pixel basis.This kind of map can be useful too to perform parallax bump mapping51.

This is the aspect of the map:



Notice “white” means the distance between the highpolygon model and the lowpolygon model was high. “Black” means the distance between the highpolygon model and the low polygon model was only a few.

For more info about displacement maps see this link:

http://en.wikipedia.org/wiki/Displacement_Mapping

xNormal outputs in the output heightmap into a [0.0f,1.0f] range (where 0.5f means flat, 0.0f is emboss and 1.0f is maximum elevation). You need to bias it in the shader if required.

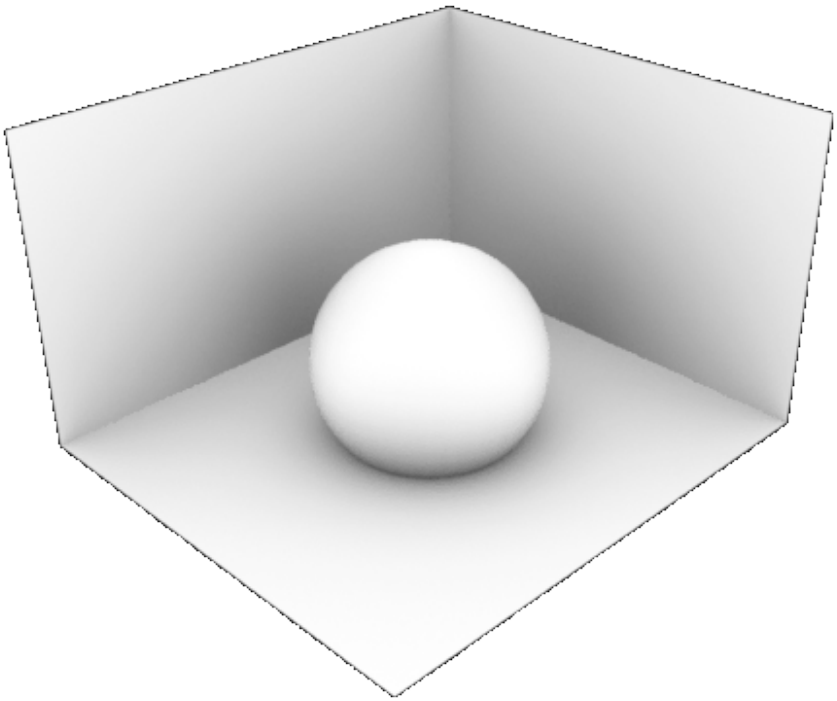
xNormal calculates this as the distance of each lowpoly surface with the highpoly model.

The height map can be used for :

- **Parallax bump mapping**.
- **Displacement mapping**. In this case, you need to pass to xNormal the lowpoly model already subdivided because xNormal does NOT apply subdivision internally. It's recommended to enable the “MatchUVs” feature in this case. Notice that when you enable the “MatchUVs” you can set an option to generate the height map unsigned/unbiased... that's to mix it with a vector displacement map.

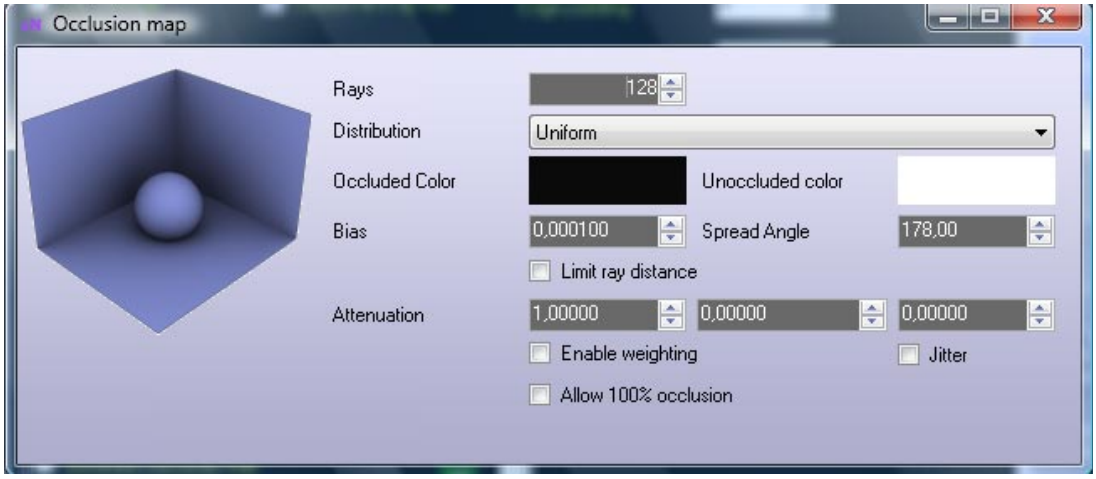
2.6.5 Generate ambient occlusion

The ambient occusion is a global illumination technique to “bake” into the object the sky lighting of its environment. Is called “ambient” because it doesn't take any directional, point or stop lighting, it's just computed assuming a light sphere surrounds the object.



As you can see it emulates very well the environment soft shadows and object inter-relations for outdoor lighting.

Lets see a bit its parameters. See this image:



You can specify the number of rays per pixel to apply. Use 64 rays per pixel to perform fast tests. 128-256 for medium quality. 256+ or more for quality. Take into consideration the time required to render this map almost scales LINEARLY with this value, so if you put there a big number you will have to wait a lot!.

The distribution combo box selects how the AO samples are distributed over the hemisphere:

- **Uniform**. Points are sampled evenly. Usually you will get very pronounced results with this option.
- **Cosine**. Points are distributed using a cosine probability function (so we get more samples near the normal). You will get smoother results with this option.

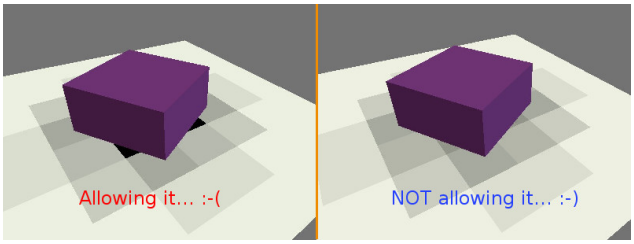
The "jittering" parameter wil add extra dirty-noise to the final map. If you want cleaner AO maps don't check this option. On the other hand, if you want more dirt applied, check it.

The "enable weighting" will modulate the AO value using the cosine of the pixel normal and the AO rays direction (so you will get a smoother transition between pixels). This will make the AO less pronounced but can alleviate some bias problems.

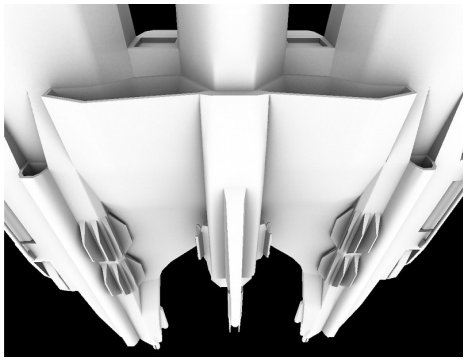
You can also specify the occluded and unoccluded colors. Usually you'll use black and white respectively... but you can use other colors to tint the occlusion shadows.

Now let's see what does the “allow 100% occluded values” option. Here's an explanation from Dan Walter (Vega Strike game):

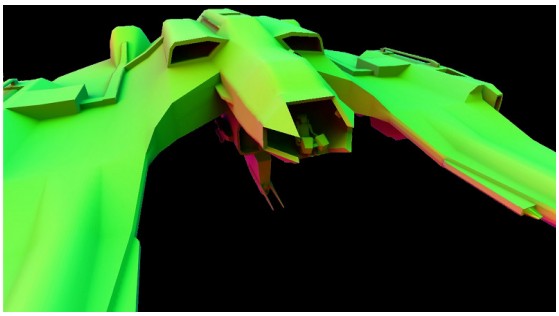
Usually, when you render ambient occlusion maps,you've probably seen jagged lines of dark texels along crossings of planes, or around the edges where a closed piece of geometry penetrates a surface.
The reason for those artifacts is that where a texel is more than 50% occluded, usually none of the rays escapes, resulting in a black texel. However, if a corner of the texel is visible, full blackness may not be representative of the occlusion level of the visible part, and looks wrong, and *is* wrong.
Setting ***Allow 100% occlusion*** to **off** (unchecked, the default) prevents those black pixel corners.



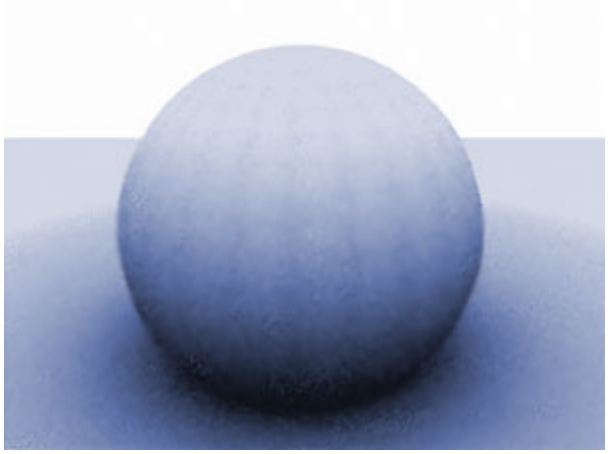
Well, if you think about it, if a texel is 100% occluded, in theory it should not be visible from anywhere; so it shouldn't matter what color it is; right?
And before you ask, no: antialiasing doesn't solve this particular problem; it was, in fact, affected by it (not anymore). Anyways, what matters is, it works; and from now on, AO bakings will look cleaner.



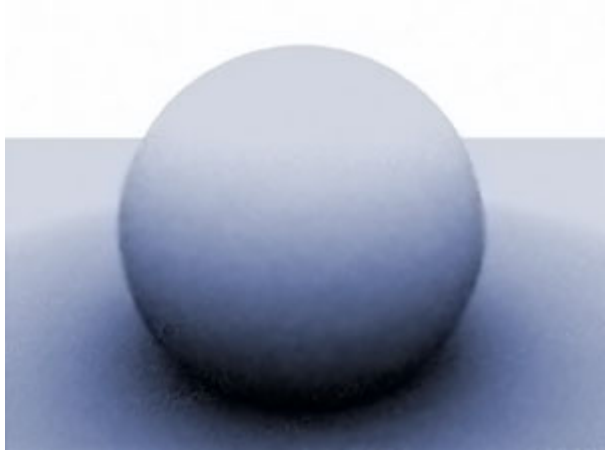
This also applies to the PRTpn:



Now let's see the AO bias parameter. It is used to avoid self-occlusion problems. See these images:



Notice the vertical edge artifacts due to incorrect self-occlusion caused by too small AO bias(0.0)

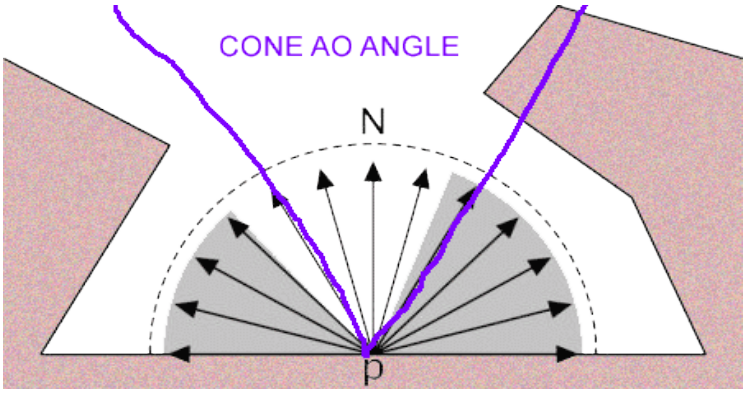


Notice the edge artifacts dissapeared when we incremented sightly the AO bias value(0.001)

Usually set the bias to ZERO. If you have some self-occlusion problems(specially at triangle edges) then increment it a few (for example, 0.001f)

Notice the quantity put there is constant, so you can scale up a bit the mesh set if you can't get a good value. Remember you can scale up/down the mesh set without re-exporting just setting the "Mesh scale" option in the corresponding lowpoly slot(in the "lowpoly" tab section).

The “AO spread cone angle” parameter allows you to specify the cone angle that you want. Minimum are 0.5 degrees and maximum is 179,5. See this image to understand this value:



Usually an angle in range [120,160] gives you better results because is large enough to capture all the surrounding details maintaining the normal direction. A greater angle (like 170-180) could incorporate unwanted samples immediatly on left/right. A smaller angle will make the AO computation faster but will skip some left/right surrounding points.

By default the ray distance is limited using the cages or the maximum ray distance (in the lowpoly tab). To use an infinite ray distance you can check the “don't limit ray distance” option. Usually you will leave this option unchecked for performance but under some circunstances and outdoor environments disable it.

The "Attenuation" option controls the distance falloff of the shadows. An object which is very far will cast a very clear shadow. Objects near will cast more "black" shadows.

The first value is used for constant attenuation. The second is to apply linear attenuation. The third value is for cuadratic attenuation. It follows this formulae:

$$Atten = 1/(att0_i + att1_i * d + att2_i * d^2)$$

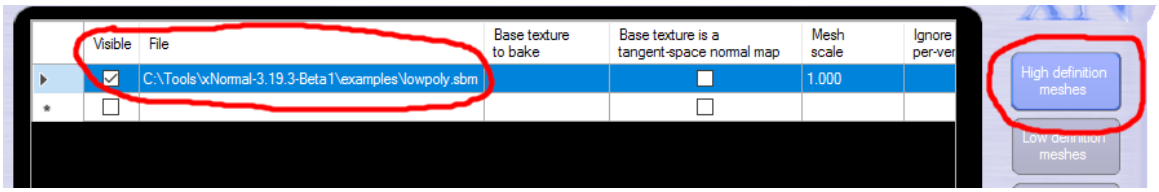
where "d" is the distance between the ray origin and the hit point. att0 is the constant attenuation. att1 is the linear attenuation and att2 is the cuadratic attenuation. The final color value will be multiplied by the "Atten" result.

To disable the attenuation just put 1.00000 into the first value(constant) and leave the others(linear and cuadratic) to ZERO... so the final formulate will be 1/(1+0+0) = 1, so won't attenuate the AO rays.

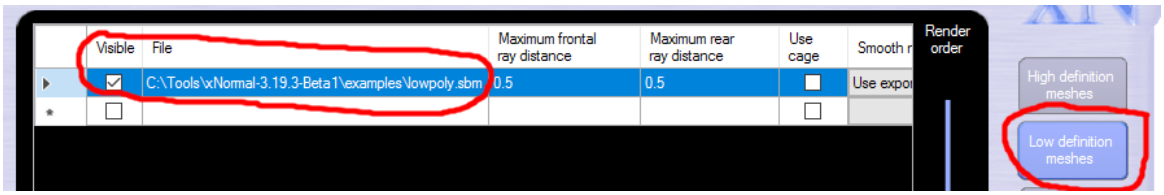
As you can see this is very similar to the Direct3D/OpenGL attenuated lights, but applied to the ambient occlusion.

2.6.5.1 Rendering occlusion without a high-poly mesh

It is possible to render the ambient occlusion without a high poly mesh projection: just assign the high-poly file the same as your low-poly file:



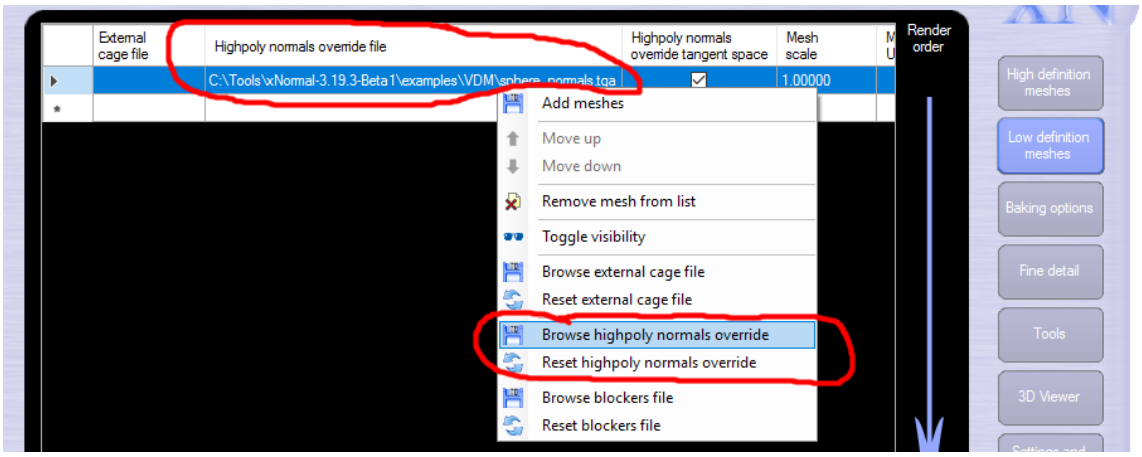
Now assign the same file to the low-poly:



In that way xNormal will calculate the AO for the low-poly model without having to project/raycast a high-poly model.

There is also other possibility to get better detail. If you already have a normal map, the "Highpoly normals override" option can ignore the hit surface normal and use the normal present in a normal map you specify.

See how we assigned the override normal map file here:



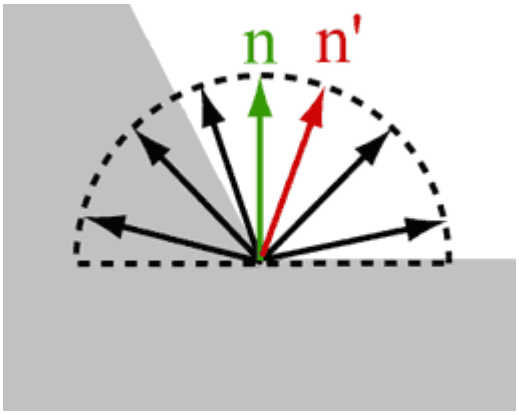
You can assign/browse the normal map file and also to reset it.

If the specified normal map is in object space remember to uncheck the " Highpoly normals override tangent space" box. In that way, specifying a normal map, the lowpoly ambient occlusion (without a real highpoly model) quality will be better.

I recommend always to calculate real ambient occlusion using a highpoly mesh in the traditional way, but if you want to bake the AO without having to project a highpoly model you can specify a normal map to get more quality than if you don't have it.

2.6.6 Generate bent normals

A “bent normal” is the aproximate direction of the rays that didn't hit any surface (so that zone will be an open area). It can be used for some global illumination algorithms:



The surface normal is “n”. The bent normal is n' and represents an “escape direction” where the rays don't hit any other surfaces.

You can also “swizzle”(apply a coordinate transformation) the bent normals or to generate them in tangent space.

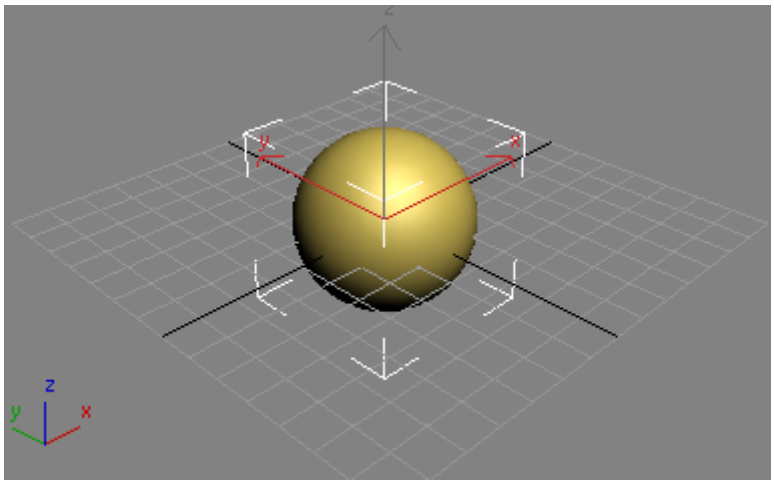
The bent normals can be used to perform more accurate environment mapping. This is the aspect of the bent normals's map:

As you can see it looks like a normal map... but it really represents the “unoccluded rays's escape direction”.

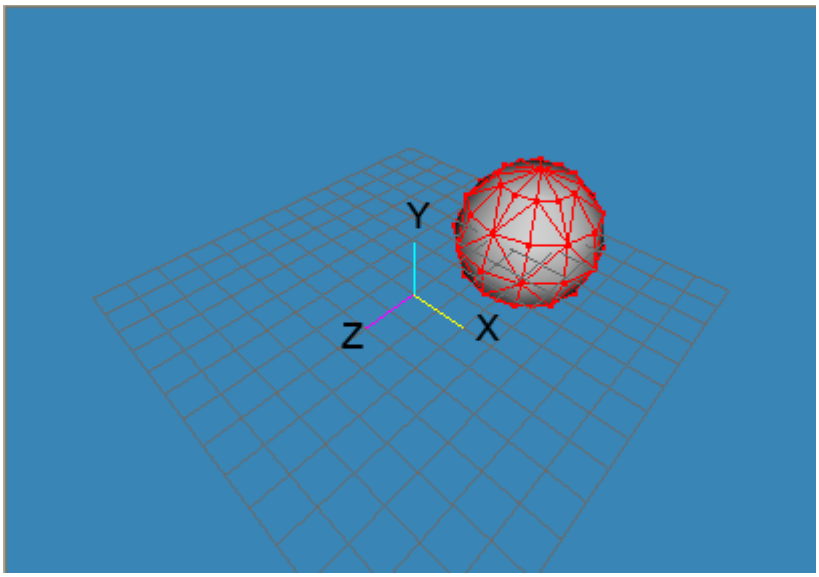
2.6.7 Swizzle map coordinates

You can change the (X, Y, Z) coordinates of the normal map on the fly.

This is very important because the programmers of your game can use a coordinate system than your favourite 3D application. For instance, some apps use Z up as:



but your programmer needs a typical OpenGL coordinate system with Y up as:



so a coordinate system change is required!

Ok, so how “swapping” works? Well, we invented a poweful method. By default no swizzle is applied if you specify “X+Y+Z+”. If you want to pass the above-Z-up model to the Y-up , just use a swap of “X+Z+Y-”. You can swap the XYZ components, negate or replicate.

Usually the swizzle option appears in xNormal as:

notice there are three combos with different color... In the image you can see, for this case, we assigned possitive X to the normal map red component, Y+ to the green and Z+ to the blue. Remember xNormal uses a coordinate system with X+ to the right, Y+ to up and Z coming from far to the viewer.

2.6.8 Normal map antialiasing

xNormal incorporates an aliasing technique which “smooths” the maps to avoid too much abrupt changes. See this image to understand what it does:

Original image (notice the edge jagging)

If you reduce the image from the original 200x200 pixels to 100x100 pixels, this is the result:

Antialiased image (notice the smooth edges)

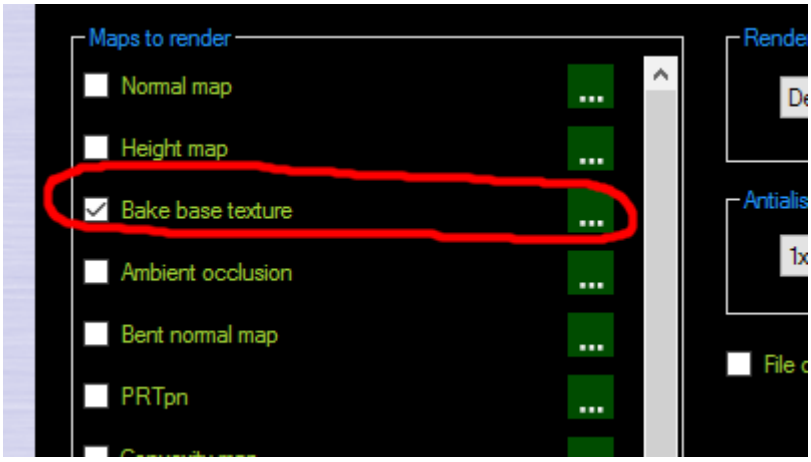
The antialiasing options are:

You can choose to disable the anti-aliasing(1x,superfast), to use 2x (mid-quality, faster) or 4x (full quality, slower).

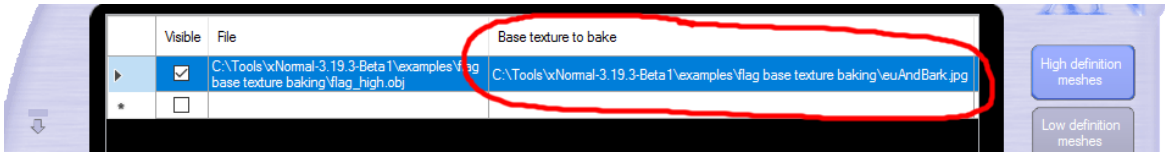
However, the “antialiasing” option takes more render time, so be cautious enabling this...

2.6.9 Baking the highpoly base texture to the lowpoly

If you have a high-poly mesh with a texture painted over it and UVed, xNormal can bake the colormap into a lowpoly mesh. That kind of map is feature is called “Bake base texture”.

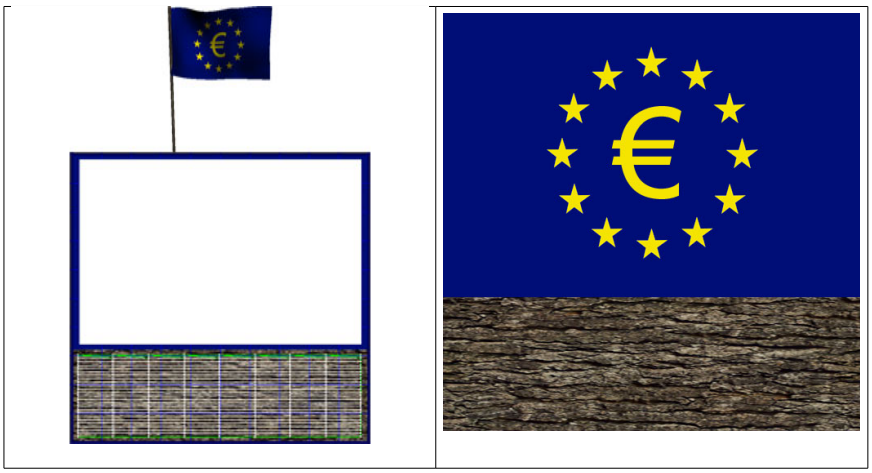


Just be sure you HP mesh has Uvs and you have assigned a texture to the mesh:



Notice xNormal includes the “Flag base texture baking” example to show this technique. You can load it clicking on the examples tab:

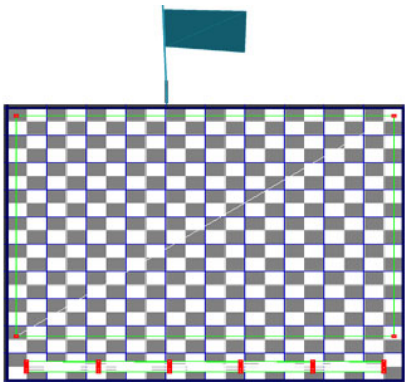
These are the texture coordinates of the highpoly model:



This shows the highpoly wireframe Uvs. Notice it is NOT white, is super-dense wireframe! On the right you have the highpoly texture (occluded in the left due to the superdense wireframe)

You have to texture-map the high-poly model to use this feature, obviously...

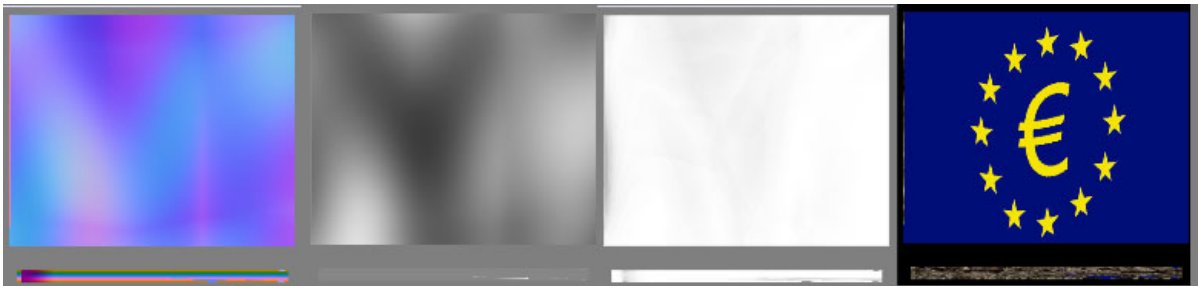
You also have to texture-map the lowpoly model (well, you were doing this already to generate the normal/displace/ambient occlusion map so...). This are the texture coordinates of the lowpoly model:



Notice this UV set is DIFFERENT from the one we used for the highpoly (notice the placement of the “X”s and “N”s and the tile disposition)

The highpoly texture “color” will be baked into this lowmodel texture set in the same way the normals/displacement/ambient occlusion is done.

So, as you can see, the artist can put all his/her effort putting all the detail into the highpoly model (geometry, texture colors, etc...). Then just models the lowpoly one and texture-UV-maps it. The xNormal will bake the normals/displacement/ambient occlusion AND the highpoly texture color to the lowpoly model texture set. This is the complete result generated by xNormal using the ambient occlusion and lowpoly texture bake features:

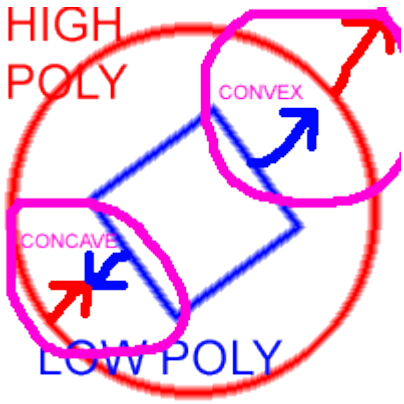


From left to right: normal map, displacement map, ambient occlusion map and base texture baked from the highpoly model

2.6.10 Convexity, concavity, thickness and proximity maps

xNormal can bake some additional advanced maps not very known but amazingly useful sometimes:

- **Convexity and concavity map.** This maps represents how are aligned the highpoly-normalmap pixels respect to the lowpoly ones. See this image:



The convexity map represents how are the lowpoly normals aligned with the highpoly normals.

A typical convexity map looks like this:

The white parts are the “convex”. Dark parts are “concave”. Notice we store the convexity in the RED texture channel and concavity (which is simply the convexity color inverted) in the GREEN channel

This kind of map is good to perform 2-sided lighting or to modulate the normalmap inside cavities like mouth interior or to improve the appereance of wrinkles. You can also use this as a fast and dirty ambient occlusion approximation. Some programs call this “cavity map”.

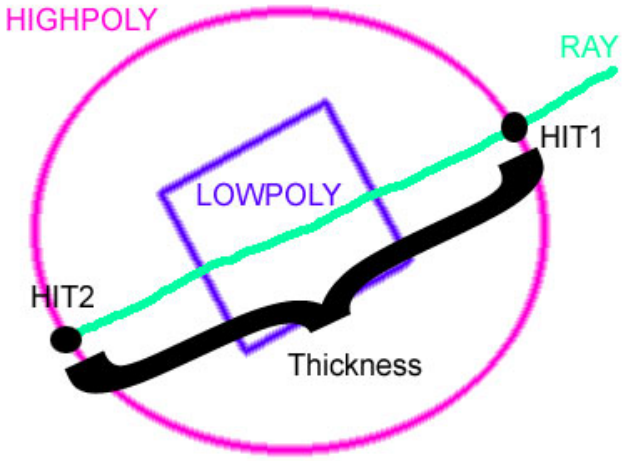
- **Thickness map.** This map represents how much gross is the highpoly model at a point. The pixel value is a length to the next ray collision (aka 2nd depth if you like the shadowmaps). A typical thickness map looks like this:

Evaluating this texture at a point we can know how “gross” is the highpoly model. Notice we project this information into the lowpoly model. Dark parts means thick and white parts mean gross. Notice we store the thickness map in the BLUE channel of the texture.

This kind of map is very useful to perform realtime **SSS**[6] **,** translucency or volumetric fog.

6 **SSS.** Sub-surface scattering

To know better what represents see this image:



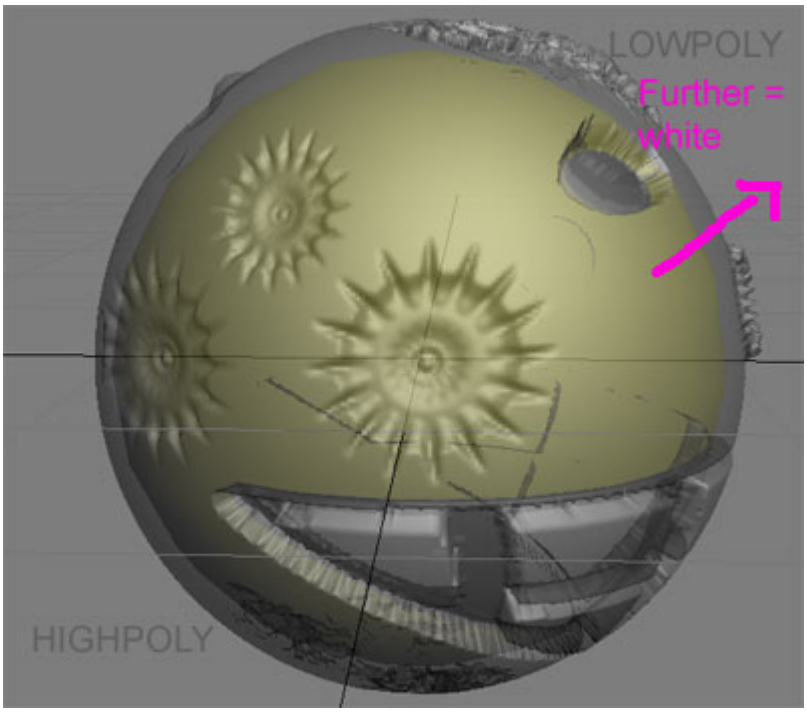
So, as you can see, it represents how much gross is the highpoly at the projected-into-lowpoly-pixel.

So this map represents how gross is the highpoly model at certain point (projected into the lowpoly model). As said, can be used for SSS, volumetric fog or other special effects.

- **Proximity mapping.** This map is a cone-multi-sampled (yes, like the Renderman's gather function) average of the distance between the lowpoly and the highpoly. We fire rays from the lowpoly to the highpoly and measure the distance. A typical proximity map looks like this:

White represents a far distance of the highpoly pixel from the lowpoly one.
Black means a close distance of the highpoly respect to the lowpoly.

This is a screenshot of the highpoly and lowpoly model:



The lowpoly is the yellow one. The grey transparent is the highpoly one. Notice the meaning of the proximity map... When we go further from the lowpoly the pixels will be white. If we go gloser to the lowpoly will be black. See the proximity map up... the acid's nose pixels are white because they go far from the yellow(lowppoly) vertices.

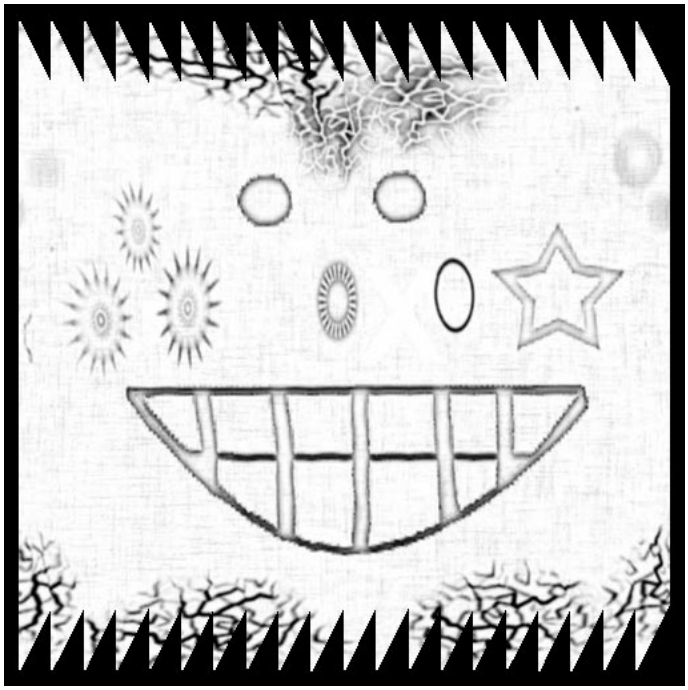
So this map is like a depth-buffer of the highpoly looked from the lowpoly model. To get better results try to use almost 100 rays/sample and a cone ray of 80 degrees.

You can use this to improve the normalmap appeareance compositing it after in a 2D-editing program . Also as an improved heightmap for parallax mapping because like is cone-multisampled so is smoother than the heightmap, but conceptually is the same than this, the only difference is that the proximity one is cone-multisampled (and frontal-only) so it can get more detail and smoother.

Notice your cage setup or the maximum frontal ray distance in the lowpoly “tab” controls how much the rays can extend from inside.

2.6.11 Cavity map

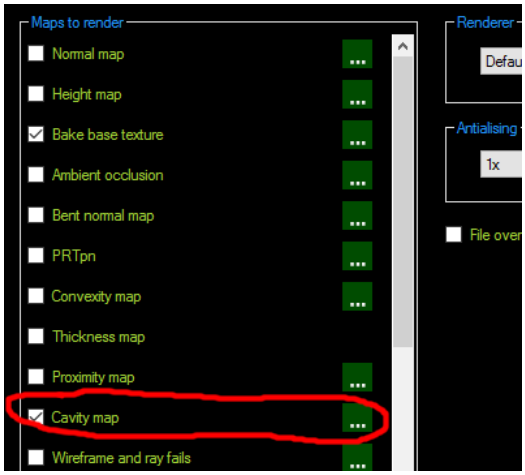
It's basically a very short range ambient occlusion designed to capture surface's microdetail holes/wrinkles/pores. For instance, this is a cavity map renderered from the smiley highpoly mesh:



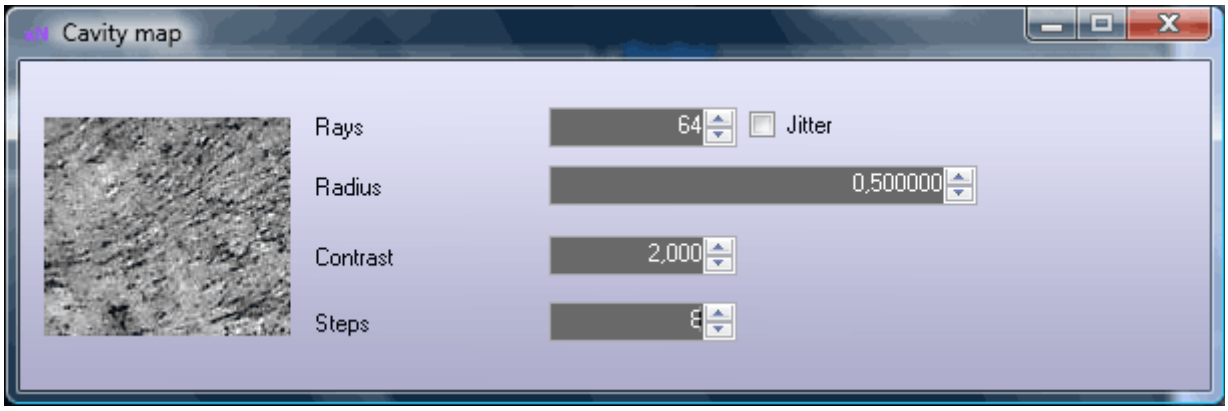
Notice it looks like the ambient occlusion map... but it casts less shadows and reflects better the microdetail.

You can use it to enhace the ambient occlusion appeareance or to multiply the base texture with it to make the wrinkles/skin pores more pronounced.

Click over the "..." button with the "Cavity" type selected to display the rendering options:



Then, this dialog will appear:



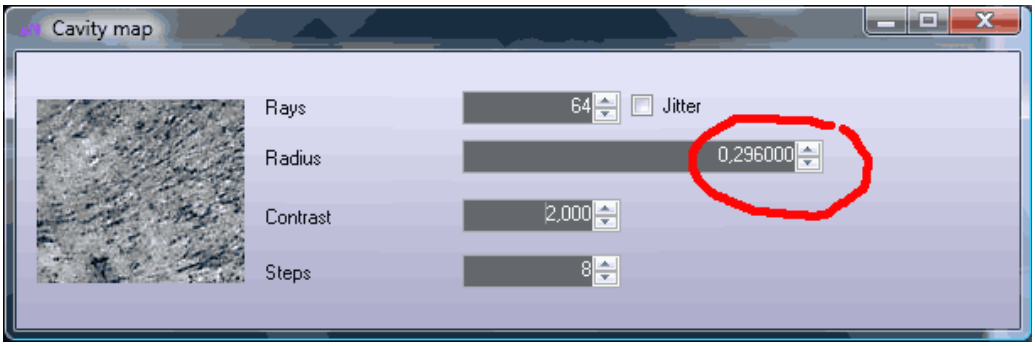
The “Rays” controls how many rays are casted around the highpoly's hit point. Increasing it will produce a more accurate result.

The “Jitter” option will add some random noise to the shadows.

The “Contrast” will enhace the wrinkles/pores appearance just multiplying the cavity value... so dark zones will be darker and white ones even more white. Usually keep this value in range 2.0f – 4.0f.

The “Steps” are the number of ray traces used. More will capture better the microdetail, but will make the rendering much slower. Use 4-8 for fast tests, 8-16 for medium quality, 16-32 for qualilty and 32-128 for film quality.

The “Radius” controls the travel distance for the rays. Making it larger will produce a blur... and making it smaller will make the result sharper. If you set it too small then you will get probably a “too white” undesired result. This option is CRITICAL… you must play a bit with the value until you get a good result !



Following these easy steps will help you a lot to set a good cavity radius distance.

In case you use other modelling programs, they will also include tools to measure the distance between two points... so follow the same procedure... zoom your highpoly model and measure the distance of a small triangle's center to its neighbor.

Notice however that the highpoly can contain a lot triangles... so you need to zoom one that would be representative... and play a bit with the value because perhaps you could want to capture really small detail... so, in that case, you will have to lower it a bit.

And remember.... the radius parameter is critical so if you set it too high then the cavity map will be too blurred... and if you set it too small then will appear white.

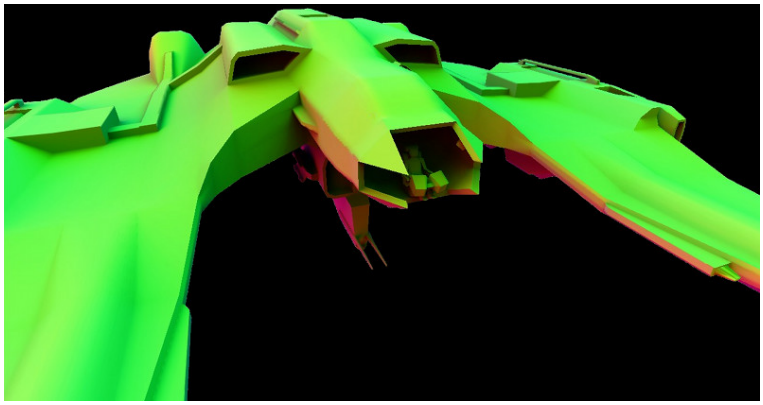
2.6.12 PRT-p, PRT-n self occlusion normal maps

These maps were originally coded for the game Vega Strike (http://vegastrike.sourceforge.net/). They are used to encode a self-occlusion normal map using 6 escape or “bent” directions ... so it's like a special “bent normals” map... with both normals and ambient occlusion coded together.

This is how these maps look:

PRT-p texture

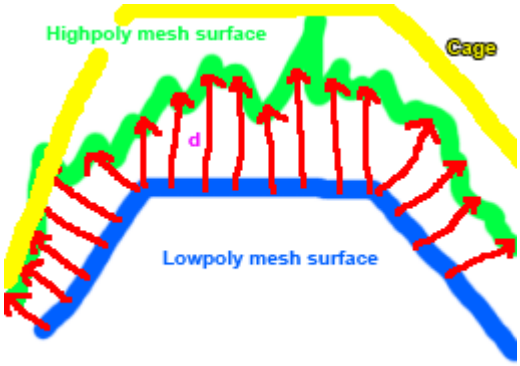
PRT-n texture



If you need more details about it I recommend you to contact Dan Walter or Klauss Freire directly or to visit the Vega Strike's forums at http://vegastrike.sourceforge.net/forums/

2.6.13 Ray direction map

It contains the 3D direction from the lowpoly mesh's vertices to the highpoly's mesh vertices (encoded as an unbiased color). As this:



If you tessellate/subdivide the lowpoly mesh and you have the ray directions + a distance “d” from a height map... you can reconstruct the highpoly mesh just using the lowpoly mesh, the ray direction map and the heightmap's distance!

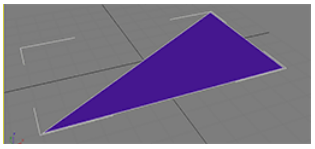
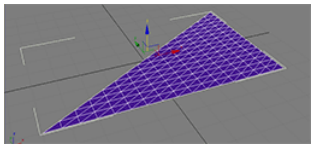
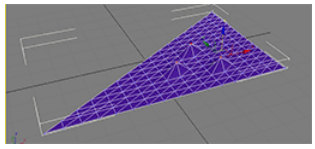
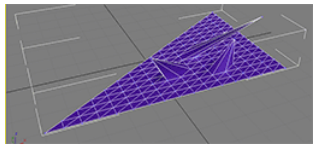
This kind of map can be used to perform vector displacement mapping for subdivided meshes in conjuntion with the “height map” and the matchUVs feature.

The conventional height map can be only used to extrude the vertex along its normals (simple displacement mapping). With the vector displacement map you can extrude the vertices along any direction.

The idea is to tessellate uniformly the polygons in the low definition mesh and then apply the following equation to each vertex:

```
newPos = tessellatedPos + ((directionMap.value*2.0f)-1.0f)*heightMap.value)
```

Understand it better with these images:

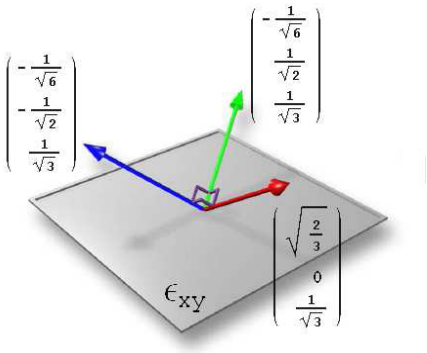
			
Lowpoly mesh's original triangle	Same triangle, tessellated	Vertices extruded along their normal using the height map (simple displacement mapping)	Vertices extruded using the vector displacement map in conjuntion with the height map

Vector displacement mapping can it's superior to the conventional one... because it can be used to model things like ears or holes.

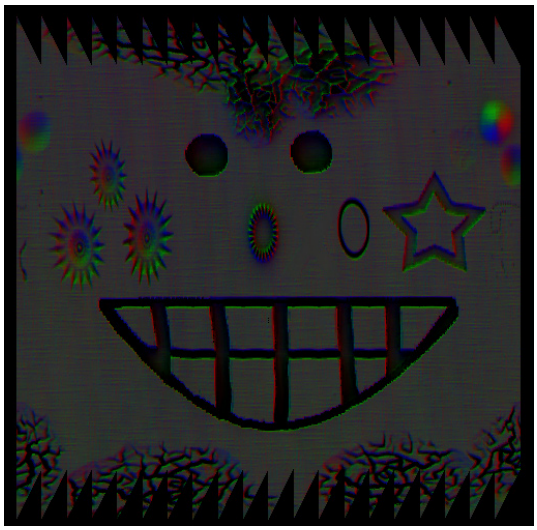
This map also can be used for debugging purposes... to know if there is some glitch/discontinuity/distorsion in the cage or control mesh.

2.6.14 Radiosity self-occlusion directional normal maps (SSBUMP)

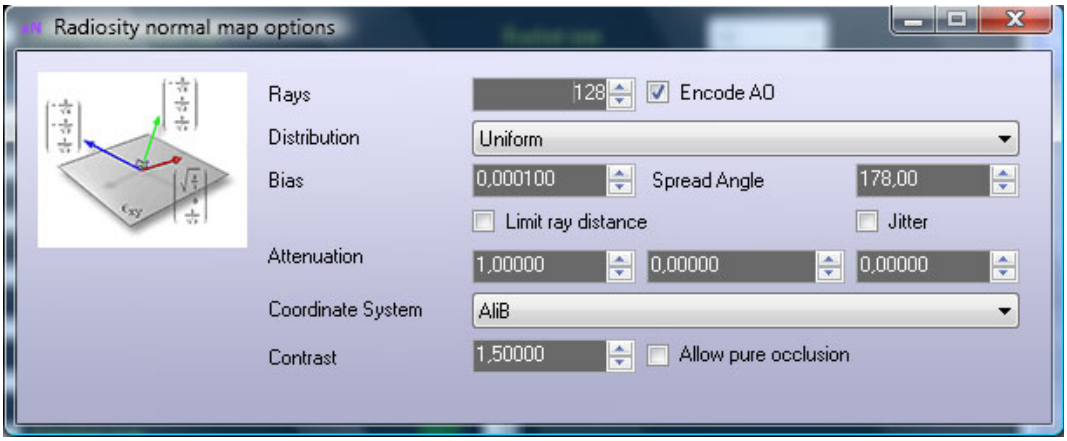
This kind of map is similar to the PRTpn one but is designed to mix the normal map with three light mapping bitmaps. The idea is to use a 3-basis-vectors covering a hemisphere so you can interpolate three light maps using dot products:



These maps look like this:



Now let's see a bit the options:



As you can see, they are very similar to the normal map + ambient occlusion ones. You can set the number of rays per pixel, the occlusion distribution, bias, spread cone angle, attenuation, etc... Pls, refer to the ambient occlusion section for more info.

The “Encode AO” option is used to multiply the radiosity normal map by the occlusion term. If you uncked it then just a tangent-space normal map (relative to the 3 vector basis) will be generated. If you check it then the three ambient occlusion terms will multiply each tangent-space normal map's channel.

The “Coordinate System” will let you specify :

OpenGL style	D3D Style (Val '07)	AliB's style
		

The violet axis indicate the UV/tangent space origin and directions.

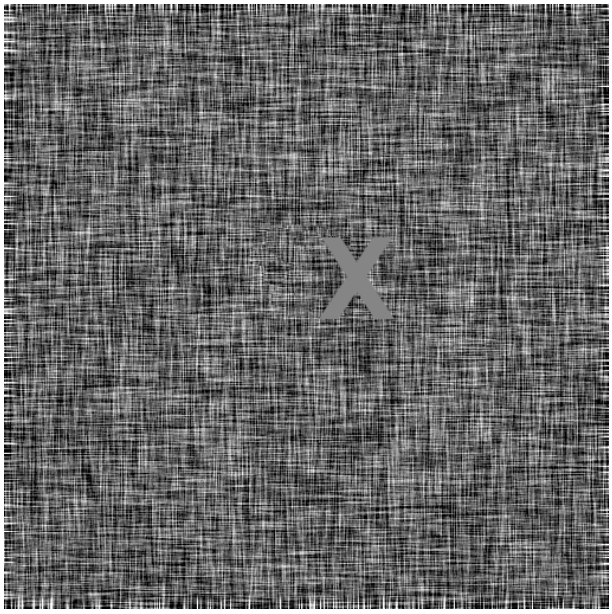
The “Contrast” is used to boost the ambient occlusion term sightly, so you can modulate the occlusion intensity.

2.7 Fine detail

You can add micro-detail to the final object-space or tangent-space normal map. Some kind of details, like cloth or paper patterns, wrinkles, etc... shouldn't be done using the high model because your polygon count will explode! We offer you an alternative: sse a height-map that will be converted to a normal map and added to the generated normal map from the high polygon.

This is a normal map:

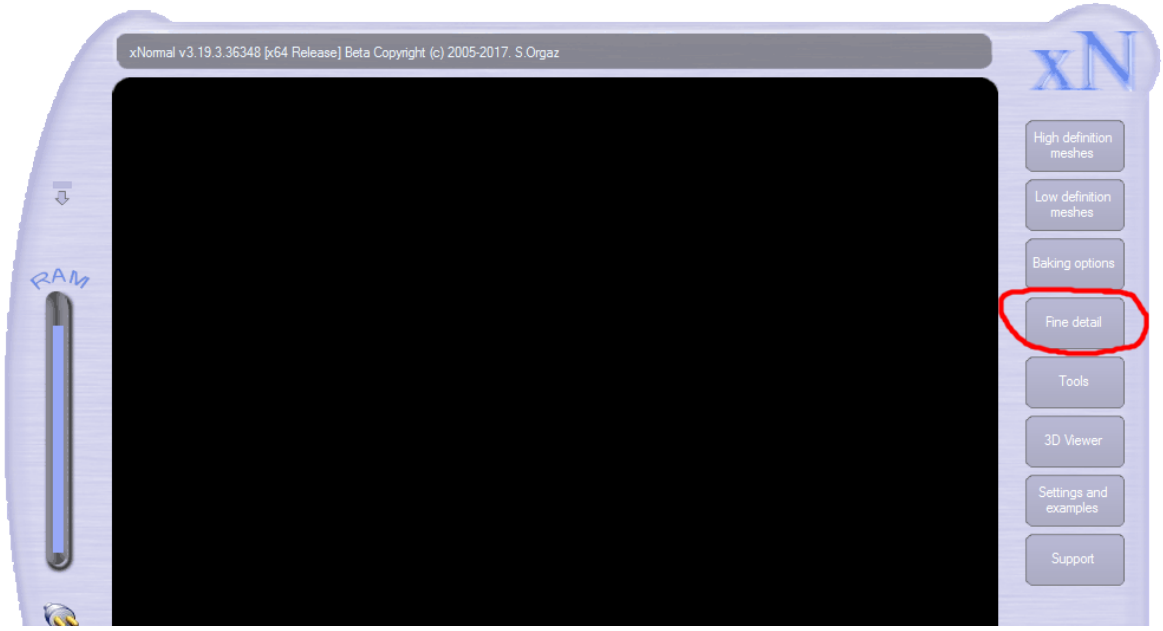
Let’s add this fine cloth detail:



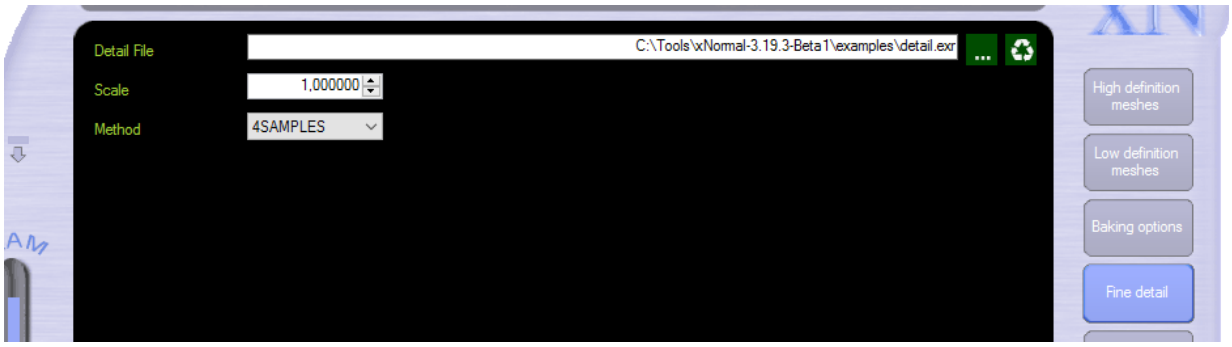
Now, xNormal will add the detail file to your tangent-space normal map:

This is very useful because in this way you can avoid tessellating/sculpting too much the highpoly model, which will increase a lot the memory consuption.

Now see how to use the detail feature. Stating opening the “Detail” tab on the right:



This will open the fine detail options:



As usually, you can press the ... button on the right to browse a bitmap with the detail and the ♻ button to reset it.

xNormal will use the RED component as height. If you use a 8-bits per pixel unsigned file, the values [0 to 127] will mean to EMBOSS, 128 means FLAT (no detail) and [129-255] means EXTRUDE.

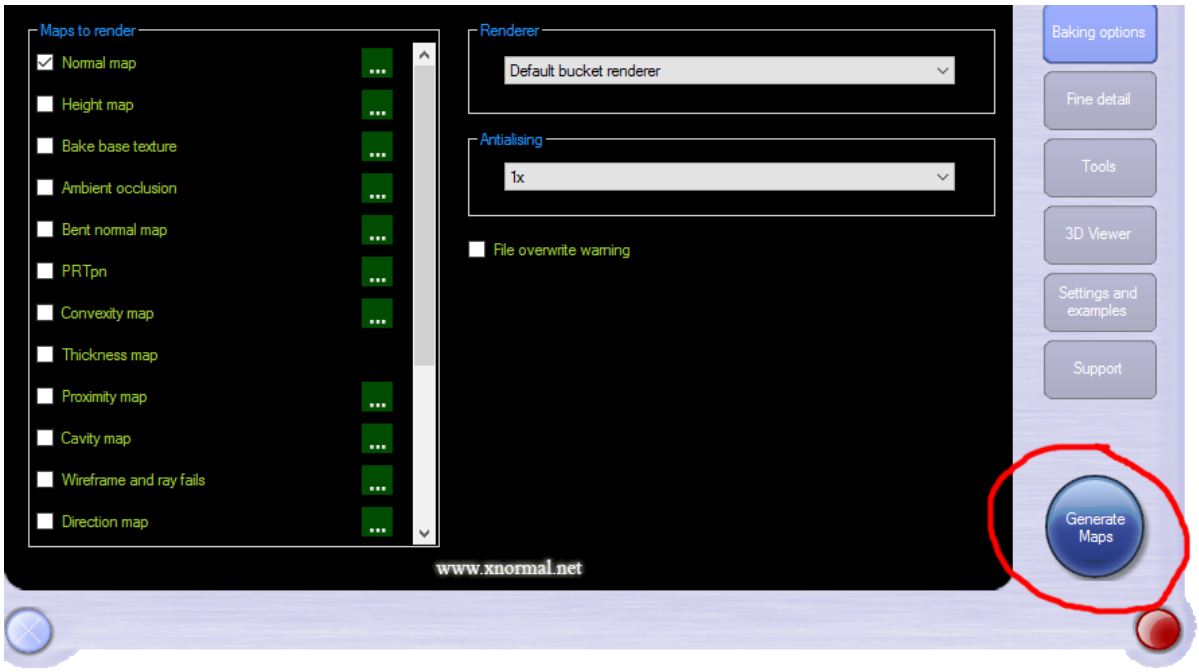
The “scale” parameter controls how abrupt will be the heights in the final mix. Greater values will attenuate the height appereance... and lower walues will make the heights more abrupt. You can se this value from [0,001] to [+ inf].

For normal maps, I recommend to use a scale in range [0.0f,1.0f].
For height maps, the fine detail bitmap's height will be added directly to the floating point map being rendered... so you need to use a scale in range [0.0f,+inf].

Although the detail map can have different size than the normalmap output, I **strongly** recommend you to use a bitmap of the same size than your output normalmap to avoid aliasing/magnifying problems. If you are creating a 512x512 normalmap then use a detail file too of 512x512.

2.8 Generating the final maps

All the operations in xNormal can be aborted. You can abort mesh loading or normal map generation. Once you have selected the high model, low model, normal map and detail options, press the “Generate Maps” button to start baking the maps:

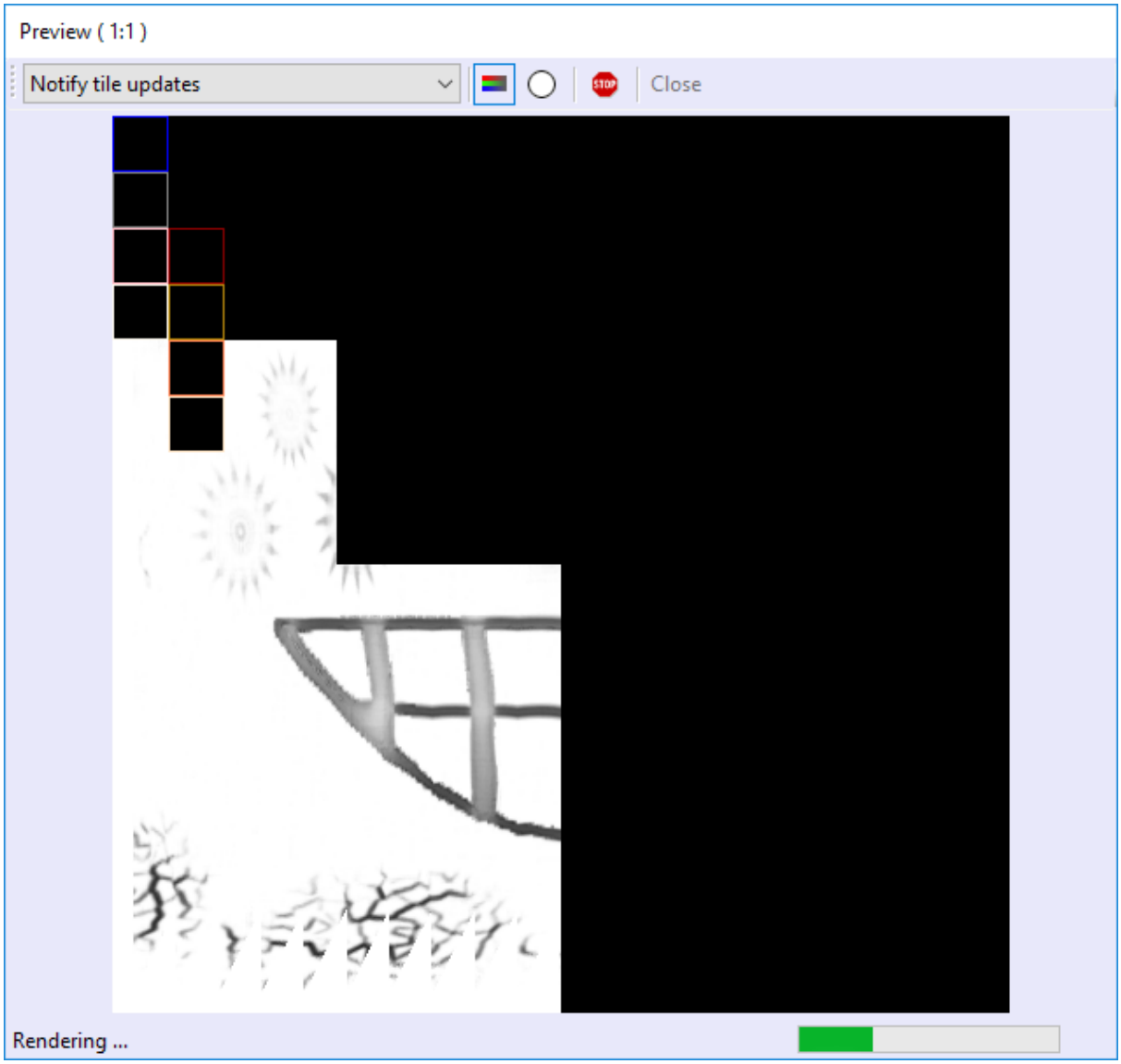


Then, the the circular aqua blue button changes to an abort/stop one:

You can click it to abort the map baking. However, please notice aborting cannot be performed instantly, it takes usually some seconds. Also, if you abort in the middle of a generation/save process, don't expect a valid output file....

Please, notice too you can change and navigate the options while generating the normal map, because the rendering dialog is modal.

While the map in being generated, you will see this dialog:



You can display the RGB channels or alpha one, stop and close the window.
The small colored rectangles indicate the buckets being rendered.

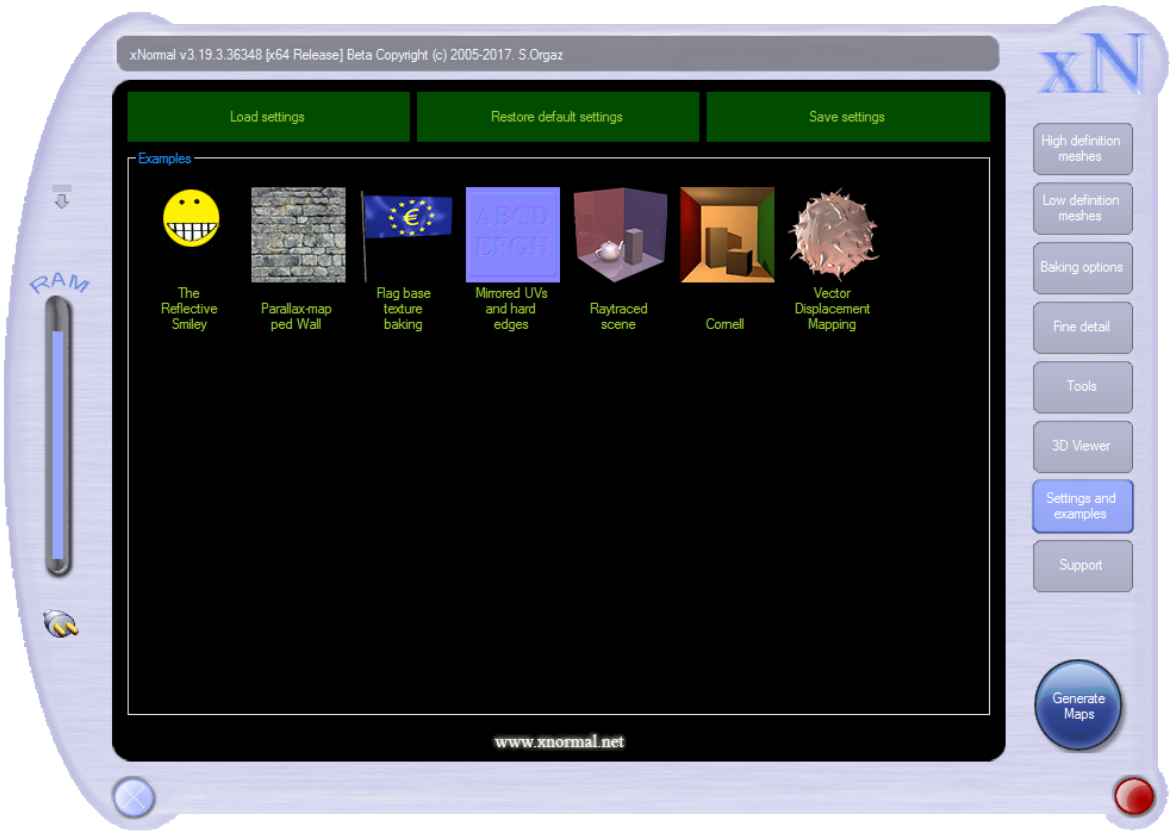
If any error occurs, a dialog will be shown indicating the error description and cause.

2.9 Loading / Saving settings and examples

xNormal saves automatically a file in C:\Documents and settings/[You user name]\Application Data\xNormal\last.xml each time you close it and loads it each time you open the application, so you don't need to put again all the settings each time. This file is saved as UTF-8, so are portable across platforms if you want to copy it.

Also, is possible to load/save a XML file with the settings. To do this, go to the “Settings and examples” on right:

Then the load/save and restore options appears:



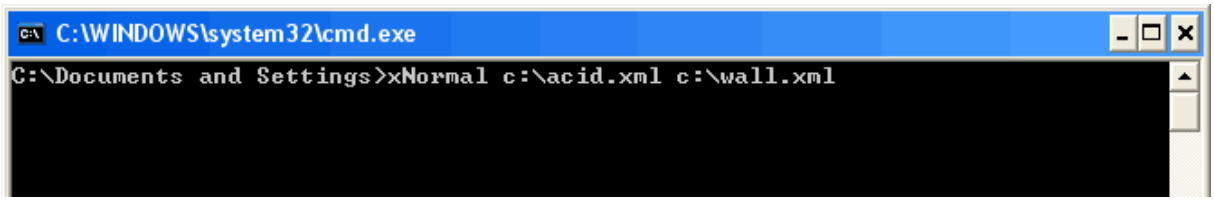
Pressing on the “Load settings” or “Save settings” button a browse file dialog will appear and then you can import or export your current settings to a XML file.

If you click on “Restore default settings”, then the default settings will be loaded.

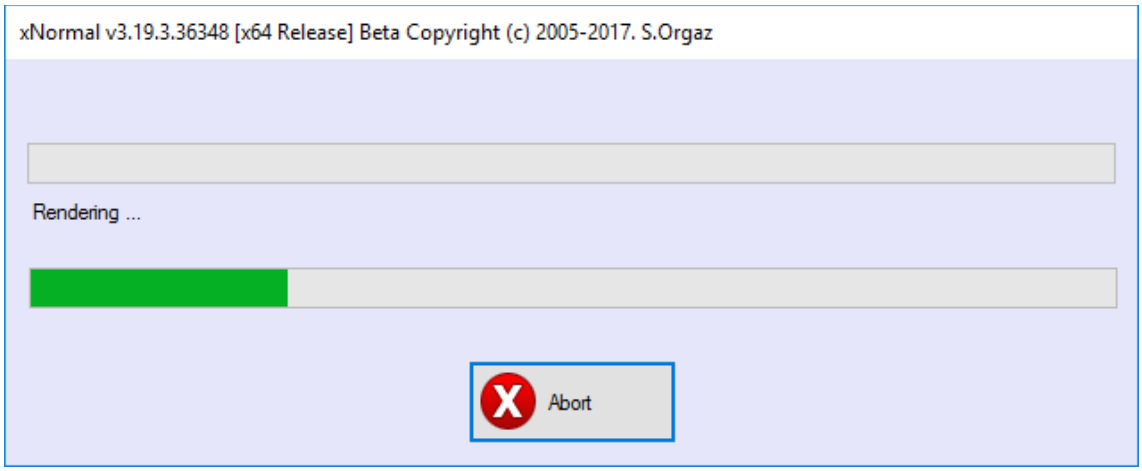
To load one of the examples, just press over the icons.

2.9.1 Rendering maps using the command-line and saved settings

You can call xNormal directly from the command line without loading the UI.



This is very useful to automate some tasks.



Just pass to the command-line as argument the XML settings files (with COMPLETE PATH). For instance, xNormal.exe c:\myModels\chair.xml c:\myModels\roof.xml c:\myGame\walls.xml

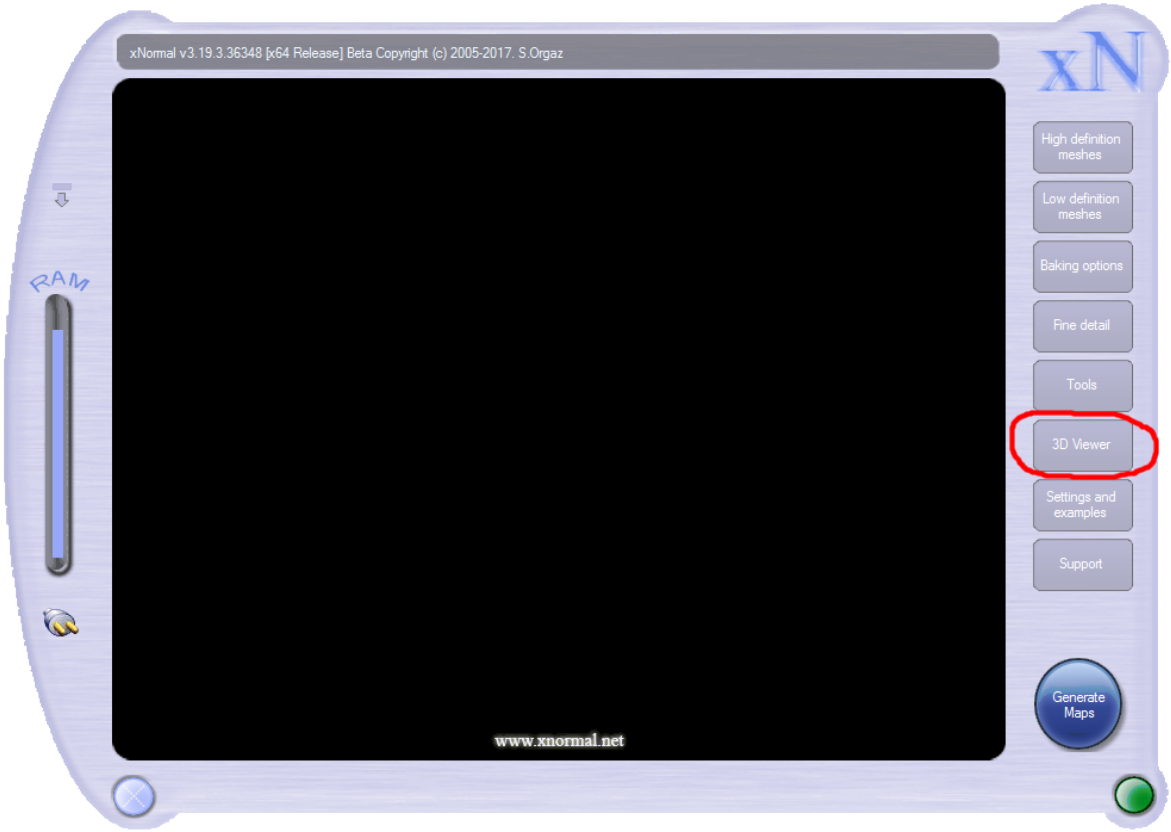
The program will return 0 if successful or 1 in case of error.

For a complete list of the commands, just call “xNormal.exe -h “

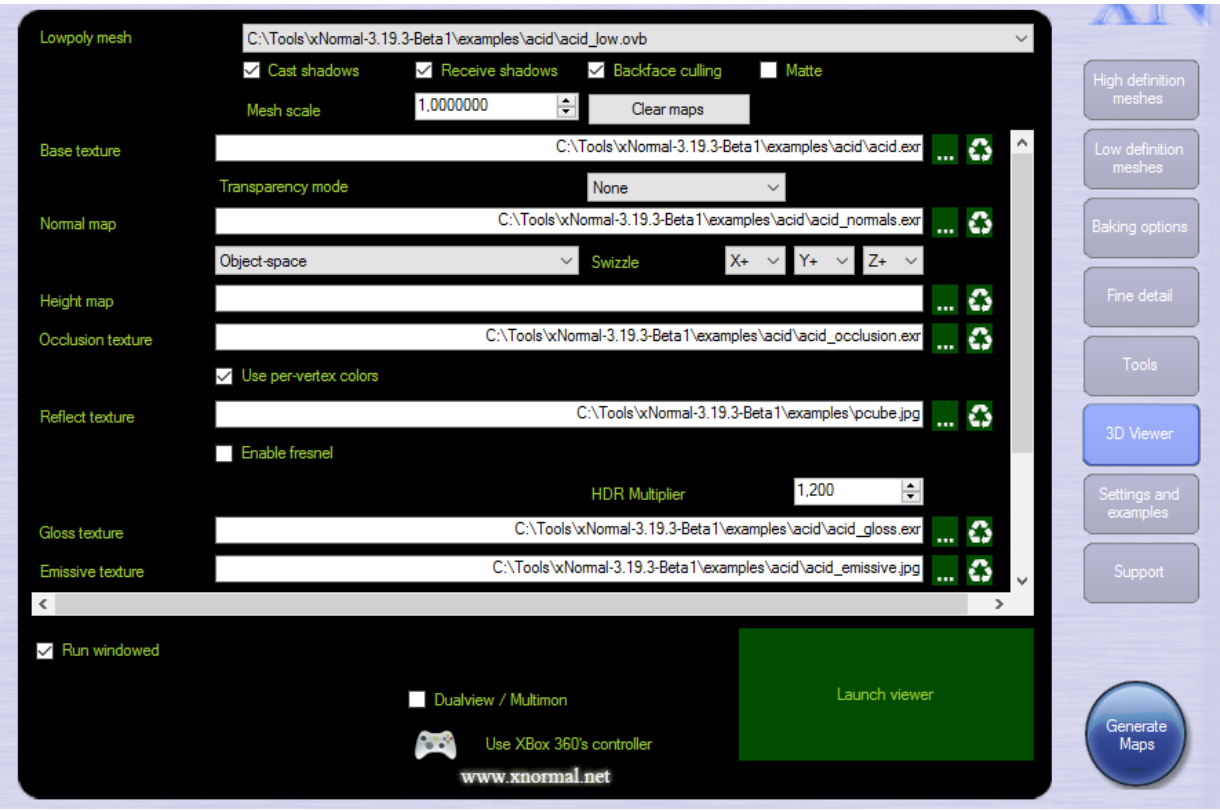
2.10 The interactive 3D viewer

xNormal includes an interactive 3D viewer.

To open it, just press over here:



Then, the viewer options will appear:



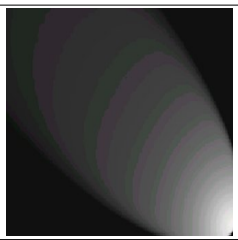
When you change the “lowpoly mesh” combo, the textures panel will appear. Lets see what does each texture:

- **Base texture.** This is the “decal” or “albedo” texture. For example, this wall:

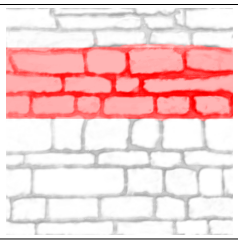
  The alpha channel controls the transparency per pixel. Notice you can set various types of transparency. If you set the 1bit alpha mode then you can specify the alpha test value (values less than that value won't be painted). The “none”, “1bit” and “additive-glow” transparency modes are not “conflictive”. All the others requires transparency sorting but the current HW can't do this effectively (aka ultra-mega-slow) so the xNormal viewer won't sort the triangles. To mitigate the possible problems of these modes you need to play a few with the “backface culling” (try to enable it), to use convex and closed objects (like a sphere) and set well the paint order in the “lowpoly tab” (see the “render order” arrow! But I prevent you... you will get artifacts... So better use the “none”, “1bit” or “additive-glow” only.

- **Ambient occlusion / lightmap.** This texture multiplies (modulates) directly the base texture, so you

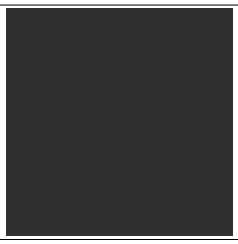
can “dark” the base texture with it. Typically represents the shadows in a scene. This is the effect that you can achieve with it

		
Base texture	Lightmap/AO	Base * Lightmap/AO

- **Specular map.** It controls shininess in the Phong lighting model.xNormal uses the RGB components to modulate the specular color:

	
Specular modulation texture	Specular modulation

But you can control highlight “power” too using the alpha channel of the texture. 0 means a big radius highlight, 255 means a very short highlight radius. xNormal allows you to specify this PER PIXEL (aka variable specular power). See:

	
Specular power (16) in alpha channel	Specular highlight result (big radius)
	
Specular power (255) in alpha channel	Specular highlight result (small radius)

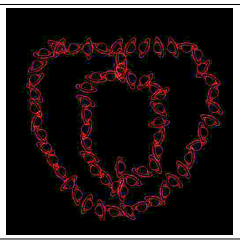
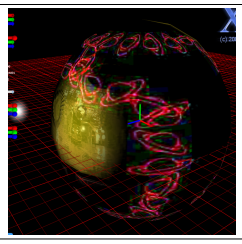
If you use a texture file without alpha channel, xNormal will use a default specular power of 16.

- **Normal map.** It represents the normals of the object for diffuse/specular illumination. Can be in tangent or object space as explained before. It looks like

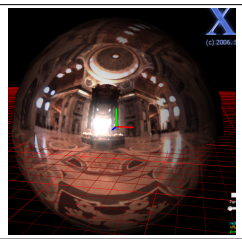
 or 

- **Height map.** Load here the height map for parallax mapping effect. It looks like this one:

- **Emissive texture.** This represents a texture “glow” using additive blending (the texture color will be added to the final result, alpha will be ignored). It is useful to illuminate always a surface, even when there are no lights affecting a surface. For example:

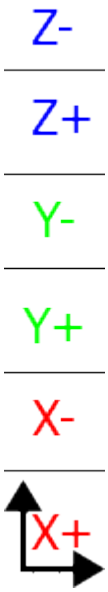
	
Emissive texture	Result (see zones where there is no light)

- **Reflection map.** This represents the environment map for the surrounding objects. Thing this texture is like what you can see if you replace the object surface by a “mirror”.

	
Reflection texture	Result

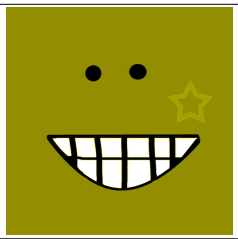
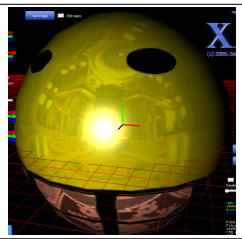
Alpha channel will be ignored for reflection mapping.

A thing you must to know is that the reflection texture must be VERTICALLY defined (for example width=256, height=1576) using 6 faces in this order:



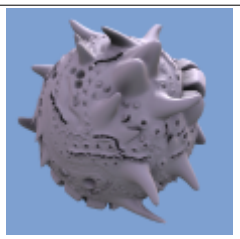
in OpenGL coordinate system (X+ to right, Y+ to up, Z+ from far to near). Notice also you must flip vertically each face bitmap (not the entire texture) to put the face origin at bottom-left. See the referenceCubemap.jpg in the xNormal's examples folder.

- **Gloss map.** This multiplies (modulates) the reflection color. The alpha channel controls the reflection blending level (0= no reflection, 255=full reflection). See these images:

	
RGB gloss texture (alpha is all 100 in this example)	Result (notice the eyes don't reflect) and now reflect is “tinted”

If you specify a reflection texture but you don't specify a gloss texture, a white modulation RGB with alpha = 70 will be used.

- **Diffuse Global Illumination (GI) map**. The mesh with this texture will take the diffuse component (RGB, alpha is ignored) of the cubemap using the pixel normal.

	
RGB diffuse GI cubemap texture	Result (with NO LIGHTING, just diffuse GI using pixels normal)

It is possible, as you can see, to specify per-mesh if you want backface culling enabled and shadows.

You can also define if you want music while the viewer is active and if you want to flip vertically all the textures (the reflection map won't be flipped though, needs manually flip)

You must select the video mode for the 3D viewer. As you can see is in format Width x Height x Color bits per pixel x Vertical Frequency x Full Scene Antialiasing Samples (FSAA).

A note on multimonitor configurations... xNormal ONLY supports monitors plugged INTO THE SAME GRAPHIC CARD. If you have multiple monitors connected to different graphics cards then you will have problems for sure!. To use a multimonitor configuration, just select the video mode of the “horizontal” or “vertical” span you have set. If monitors are each one 1024x768 and are configured in an “horizontal span”, a video mode of 2048x768 should appear to use multimonitor. However, you CAN ignore the multimonitor configuration just selecting the 1024x768 video mode, so although you have two monitors plugged, only one will be used.

The viewer will use the low polygon model options, so sure you put well its options. Notice too the mesh must use only ONE material. If some faces in the low polygon model have assigned one material and other faces have other material perhaps the model won't be well rendered... So use meshes with only one material and texture applied for ALL its faces.

Once you set all the options, press in the

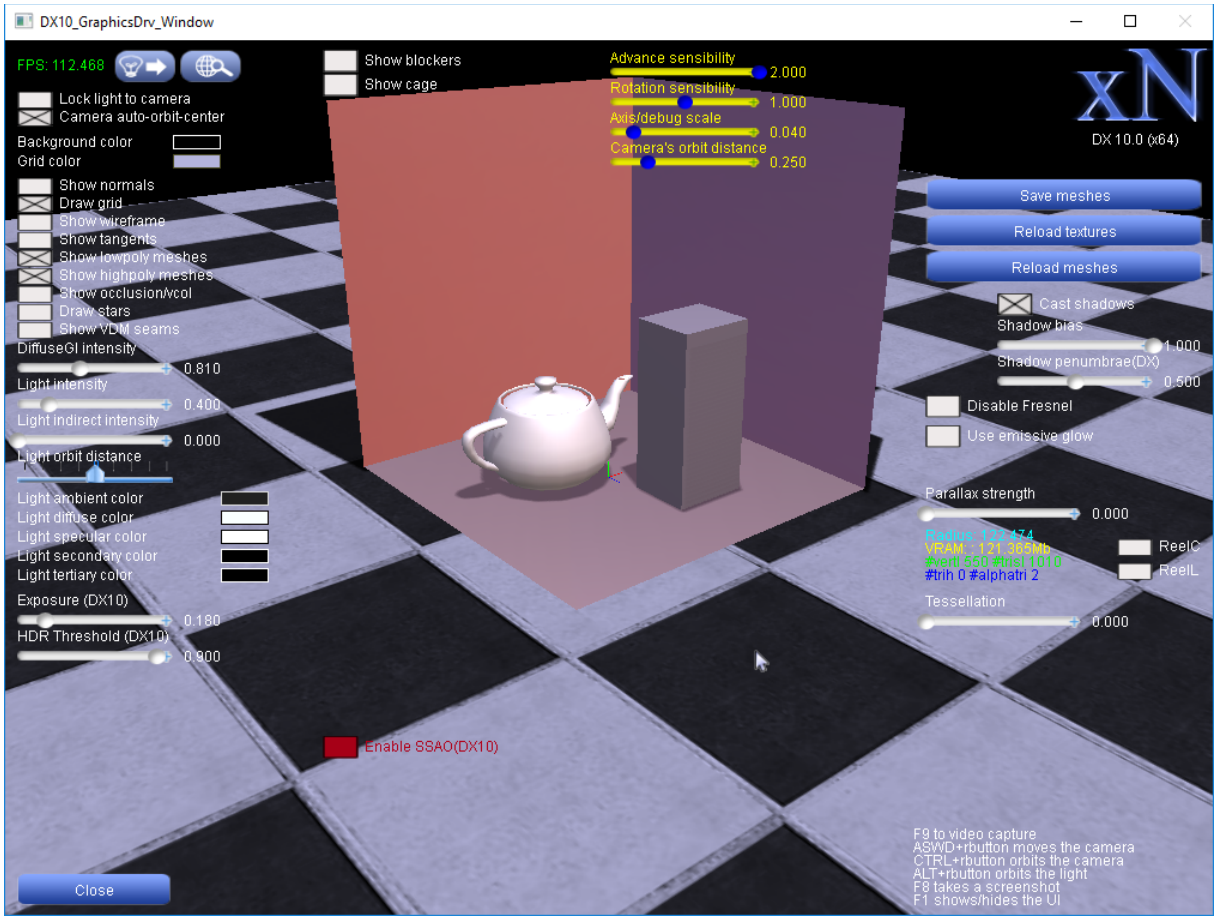
button to launch the viewer. Notice if this button is not enabled is because your graphics card lack any required feature to run it or because you don't have mesh models visible. Also can be disabled if there are no meshes selected.

Once you press it, the textures and mesh will be loaded into VRAM. A “loading ....” screen will appear while.

If any error occurs in the process, a message will appear (check out the xNormal_debugLog.txt file for details).

2.10.1 Inside the 3D viewer

Once you enter the 3D viewer, you will see something like this:



At this point, if you notice it is REALLY slow painting, is because you your graphics card doesn't support a needed feature perhaps the renderer passes to software mode which can degrade the performace a lot (check the xNormal_debugLog.txt file for more info then).

As you can see, you can show normals (to see if model is well aligned and you set correctly the swapping coordinates), show / hide the grid (which is always in height=0.0), show the model wireframe and tangent basis of each vertex.

Also, you can control the light intensity (middle is 1.0, left is 0.0 and right is 2.0) and diffuse/specular color. Too if you want to disable the fresnel reflections and emissive glow effect strength.

Clicking over then “Light to camera” button will place the light where the camera is at the moment.

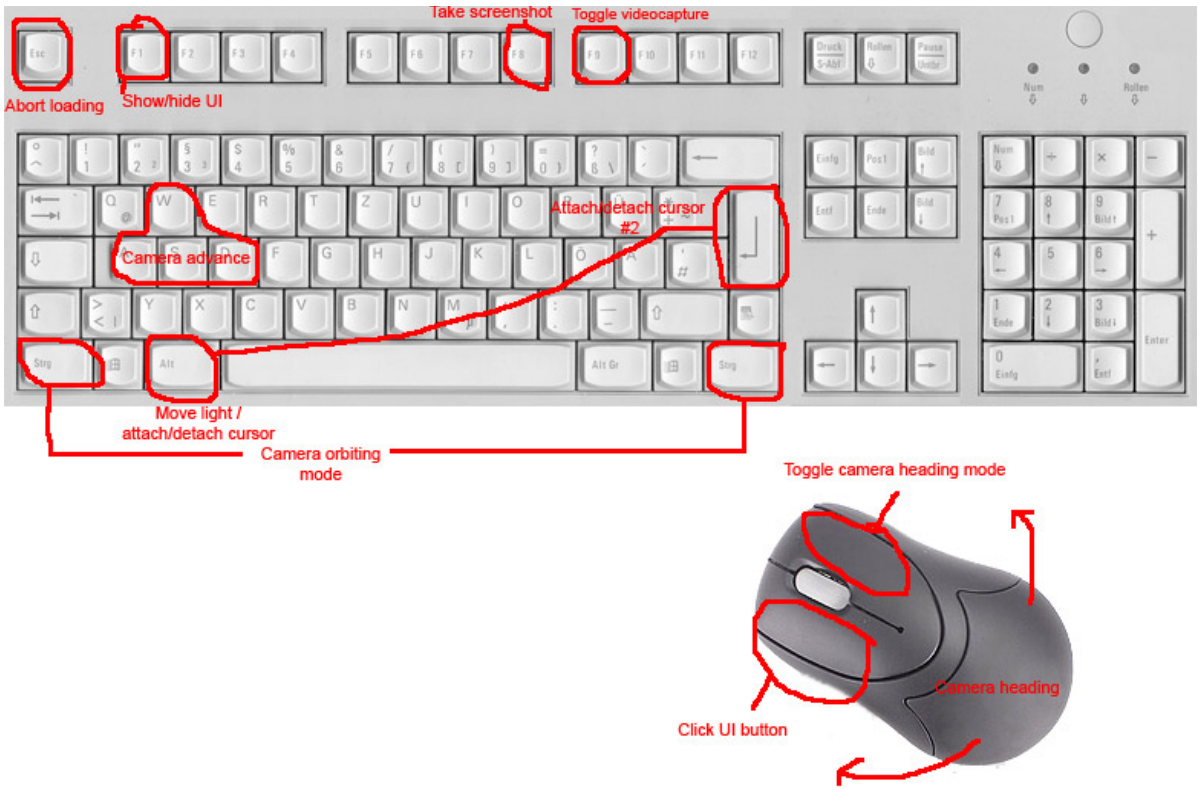
On the top there are two slider to control the camera and keyboard advancement sensibility.

Pressing in “Close Viewer” you return to the xNormal UI interface.

You can control too the “parallax” effect and glow strength with the slider and the checkbox on the right.

Also, you can refresh the models and/or textures pressing over the “Reload models” and “Reload textures” buttons. This is good if you re-exported the textures or the geometry and wanna refresh them in the viewer without closing it.

You can use the keyboard keys as follows:



Use the keys ASWD to straffe and move camera with mouse + right click.

You can use “orbit” mode holding CTRL + mouse right click. Also, you can orbit the light using ALT + mouse right click.

To hide/show the UI press the F1 key

You can take a screenshot with F8, toggle video capture with F9. The screenshots and videos will be saved into My Documents\xNormal\screenshots.

If you run the 3D viewer in “windowed mode” you can attach/release the mouse cursor pressing ALT+ENTER.

If you have a Xbox 360 gamepad controller (aka XInput) and wan't to use it sure you check the

option before running the 3D viewer. The controller will vibrate when you start the 3D viewer to indicate that is ready to be used. Use the gamepad as follows:

Use the “B” red button to toggle the cursor mode (to click the UI elements) / camera mode.
Use the “A” green button to show/hide the UI. Use the “start” button to take an screenshot.

Use the right thumb stick to move the cursor / camera look-at point. Use the left thumb stick to advance the camera (up=advance, down=go backwards, right/left to straffe)

Press the the left-right-up-down pad to click over the UI buttons/sliders.

Use the right trigger+right thumb strick to orbit the camera. Left shoulder+right thumb to orbit the light.

Use the left trigger+ right thumb (with the edit cages option checked) to select cage vertices.

Finally, use the left shoulder + right shoulder (if you set the “run windowed” option) to attach/release cursor.

Press START to take an screenshot, START+BACK to toggle videocapture. BACK to abort loading or exit.

The 3D viewer UI is fully customizable using the LUA scripting language (see http://www.lua.org). Edit the ui.lua file in the [xNormal folder]/ui directory for more details. If you did any World of Warcraft or FarCry games “mod” I am pretty sure you know this scripting language already...

Clicking over the “Save meshes” on the right, all the lowpoly meshes will be re-saved using a mesh exporter.

Re-exporting meshes is VERY useful because you can calculate the tangent basis in xNormal and then export it for own 3D engine. Un-used-by-xNormal data like vertex skin and bone info, vertex colors, etc... will be preserved and passed to the selected mesh exporter !

2.11 The plug-in manager

xNormal uses a plug-in or module architecture. You can make tons of different plugins like mesh importers/exporters, image importers/exporters, image filters, tangent basis calculators or raytracers. Each time xNormal starts, it scans the DLLs inside the [xNormal folder]/plugins directory. When it detects a DLL has been made using the xNormal SDK and it properly exports some functions, the plug-in will be valid and loaded into the application. Notice that a DLL can contain more than one plug-in inside.

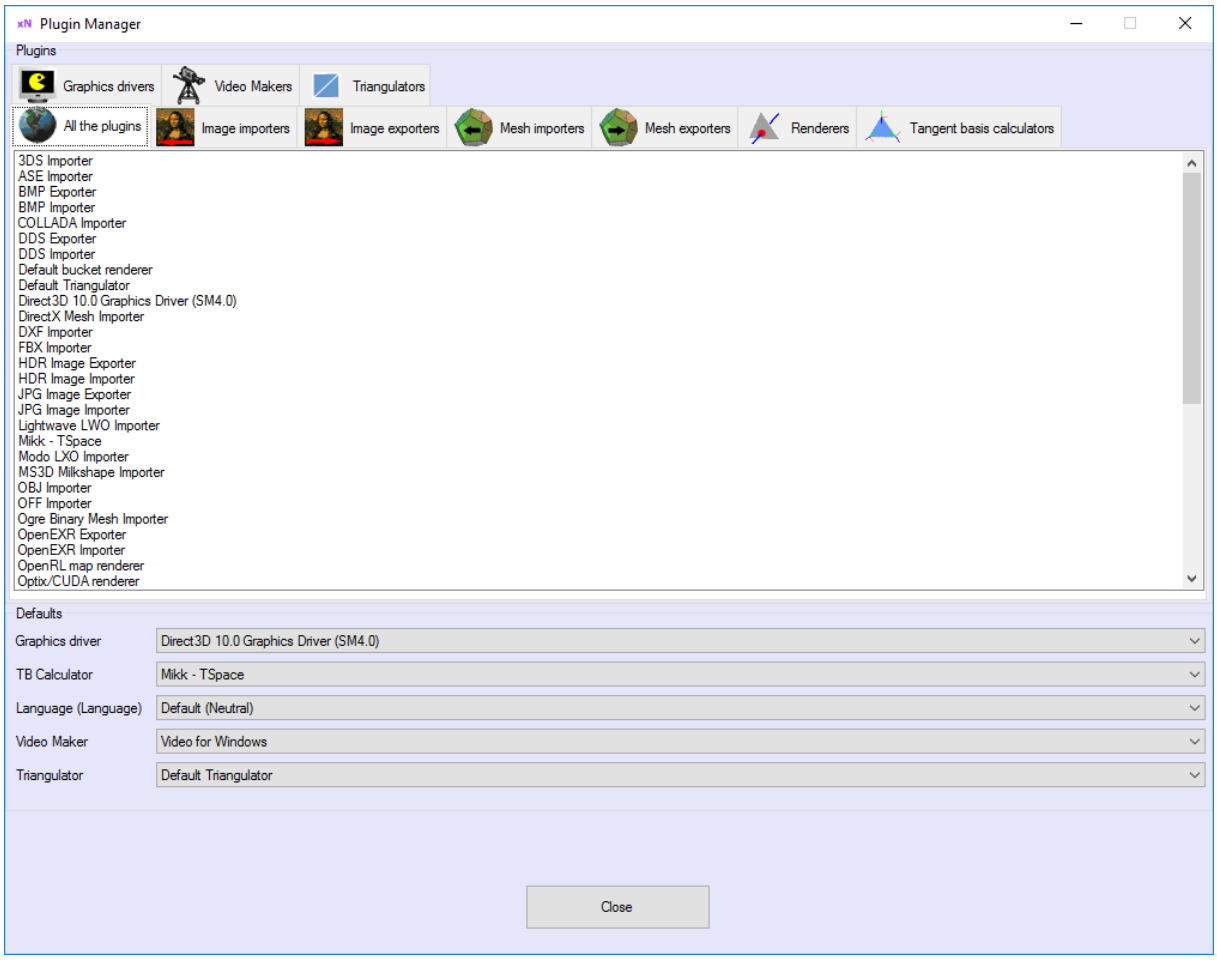
The xNormal SDK allows the programmers both to make plugins for xNormal to extend the program or to use the library as a powerful helper library to calculate mesh tangent space or normals. For more info about this SDK please consult the file [xNormal folder]/sdk/doc/QuickSDKGuide.pdf document.

You can see all the plugins loaded or configure them using the xNormal Plugin Manager. To open it, click in the small plug-in icon on the left as shows the image:

Then, you will see something like this:

You can filter by plug-in type using the tabs and then select a plug-in to see some info about it. Configurable plug-ins will enable the “Configure” button so you can click it to setup some options of the plug-in.

Inside the “Defaults” group box you can set the default graphics driver, tangent basis calculator and default language to use.

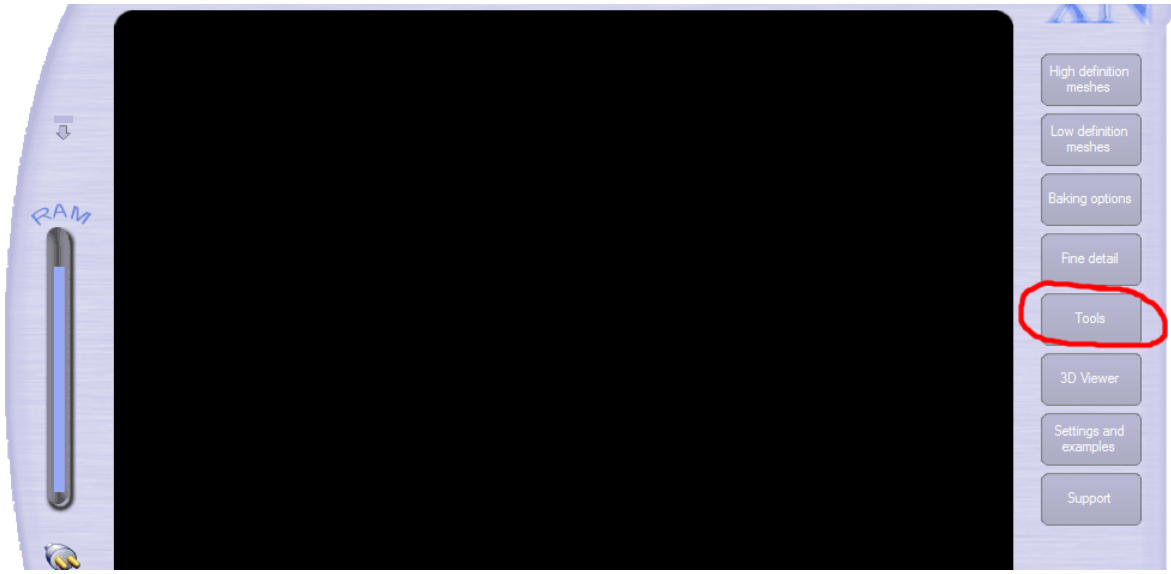


You can select a plug-in and then the “Plug-in information” and “Configure” buttons will appear.

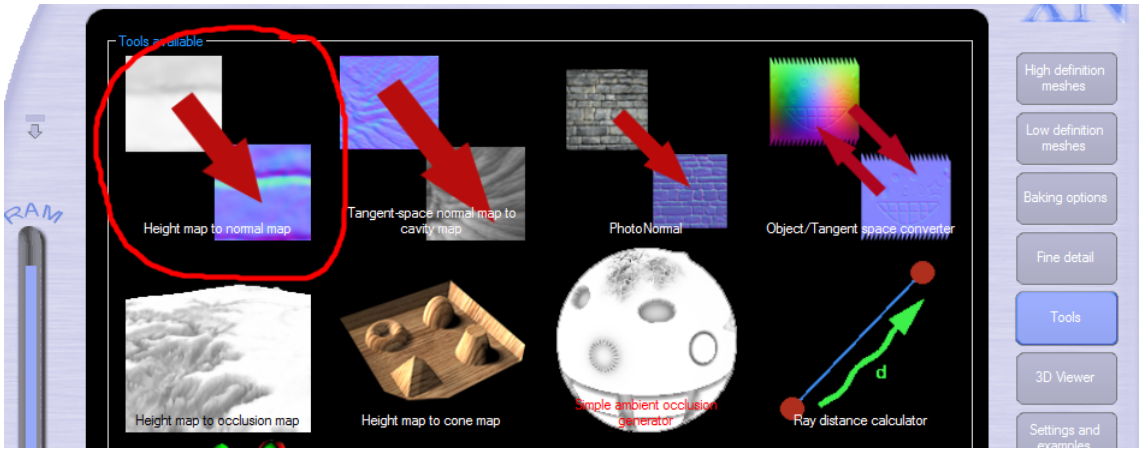
Pressing the “Close” button the Plugin Manager will save the settings and close the dialog. Then, you will return to the xNormal application main form.

2.12 The height map to normal map tool

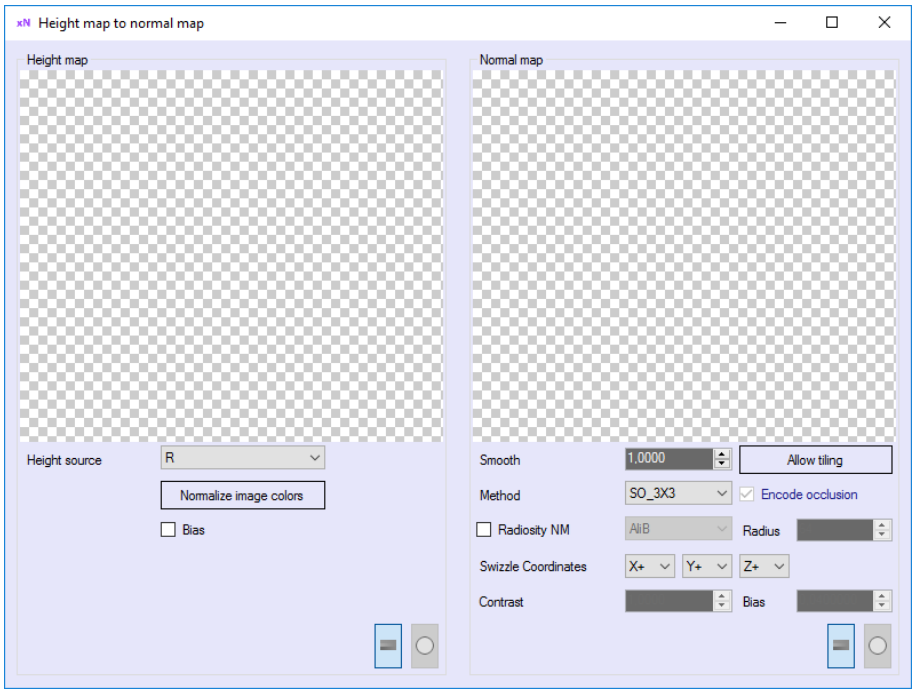
xNormal comes with an extra tool to convert a height map to a tangent-space normal map. To open it press over the “Tools” tab button:



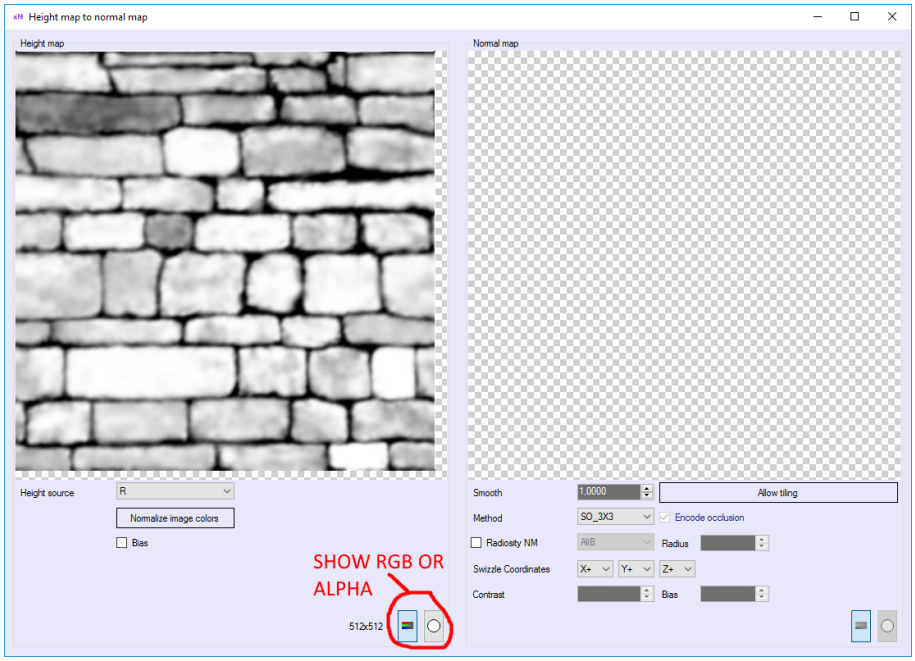
And then press over the “Height map to normal map” button:



Then this window will appear:



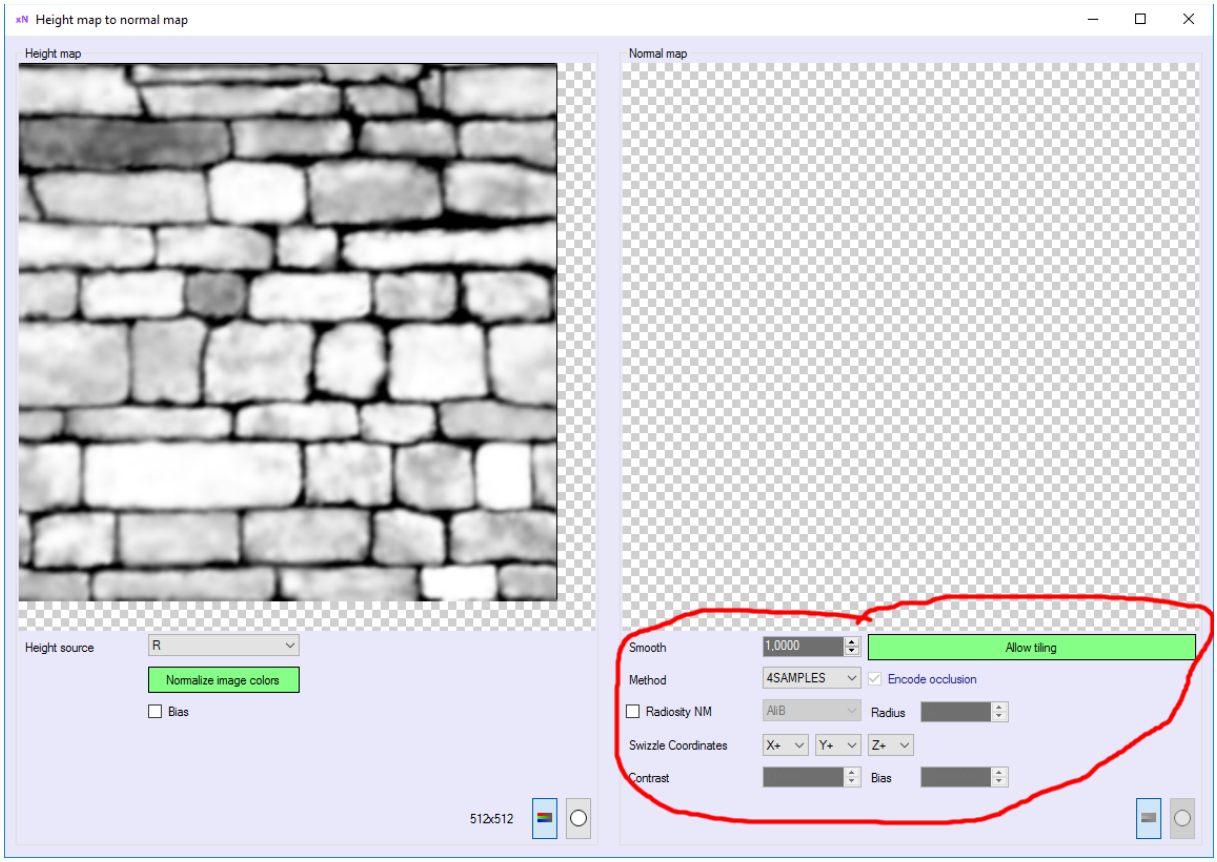
Now you can load a height map clicking the right mouse button over the left panel. Then select “Browse height map” on the context menu and an image will be loaded into the left panel:



Then, you can show the RGB components of the image or to show the alpha channel and also, if the texture you selected has mipmap levels, to view the different mipmap levels (only available in DDS, EXR and SuperHDR-RAW images usually if were exported). You also can choose the source of the height in the image (red channel, blue channel, green or alpha channel) and to normalize the source image colors (this

will auto-adjust the bright and contrast to keep all the colors in range [0.0,1.0f], where 0.0f means black and 1.0f pure white.

Now you must choose some options to generate the normal map from the recently-loaded height map:



The “smooth” parameter will make the normal map less “abrupt” and blurred. 0.5 or 0.6 is a good start value.

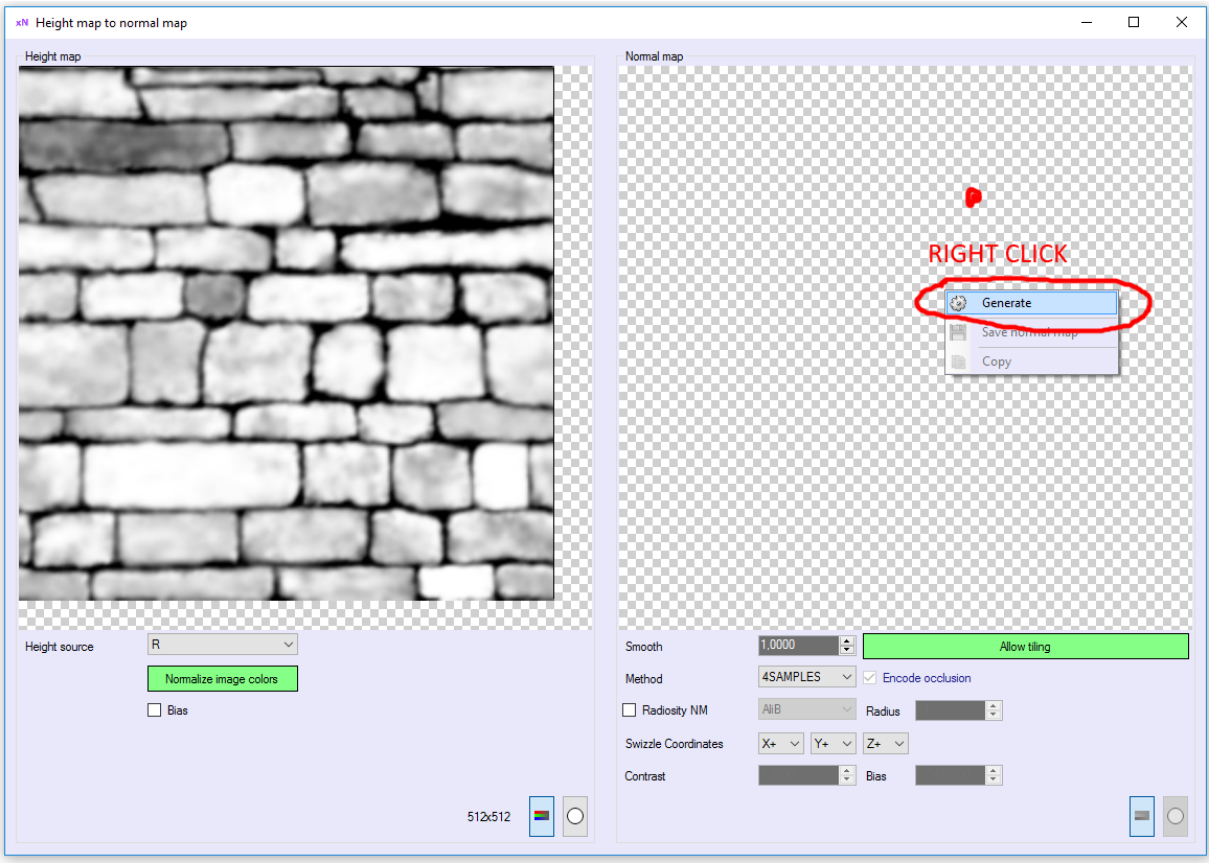
The “method” controls what algorithm will be used to generate the normal map. Usually the “4Samples” is a good starting point.

The “Allow tiling” will make the pixels near the texture borders to wrap to the other side of the texture.

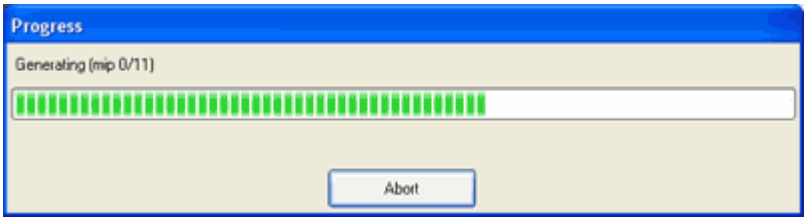
The “Put heights in alpha” will copy the original height map values to the alpha channel of the normal map. If not, will copy the original texture alpha channel.

The “swizzle coordinates” option will transform the resulting normal map to the coordinate system that you specify.

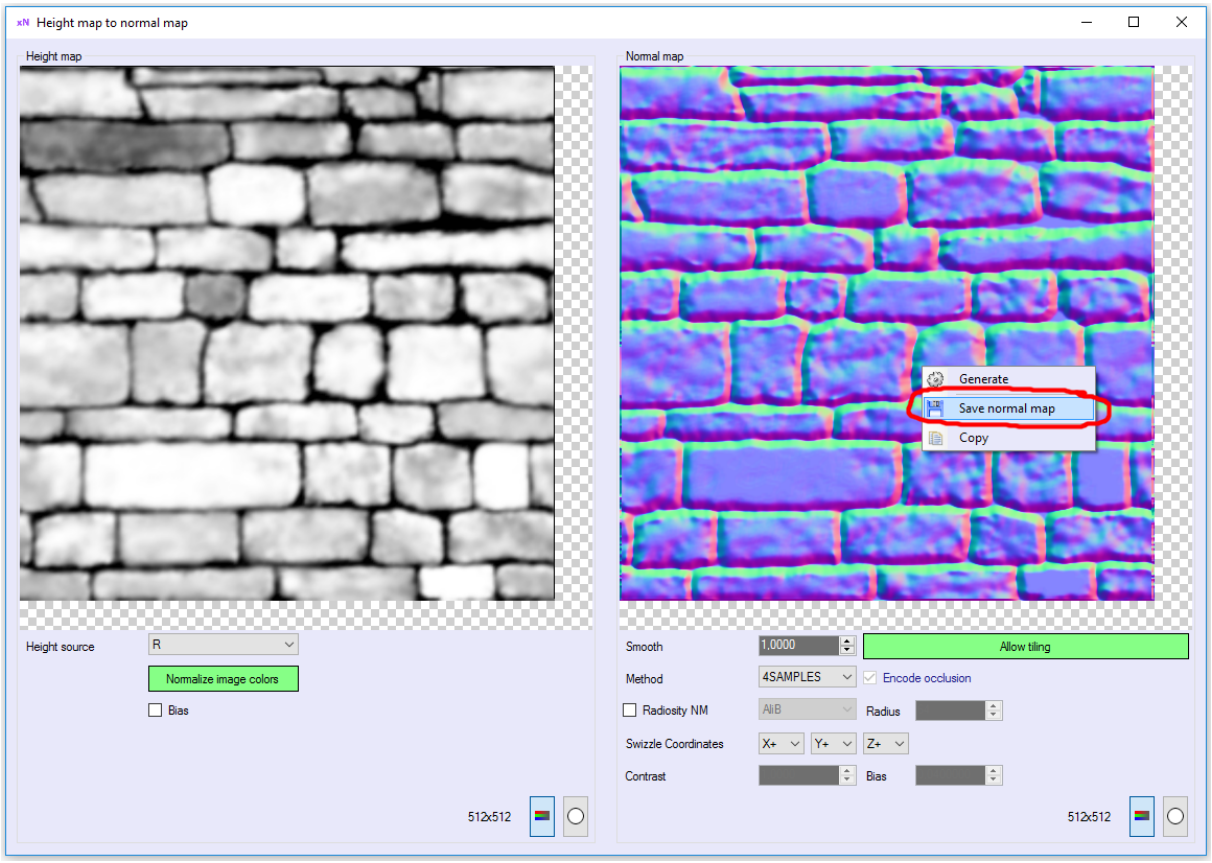
Ok... At this point you can press the mouse right button over the right panel and then another context menu appears:



Then a progress window will appear indicating the total generation progress and you could also abort the process:



Now you could see the tangent-basis normal map generated on the right panel:

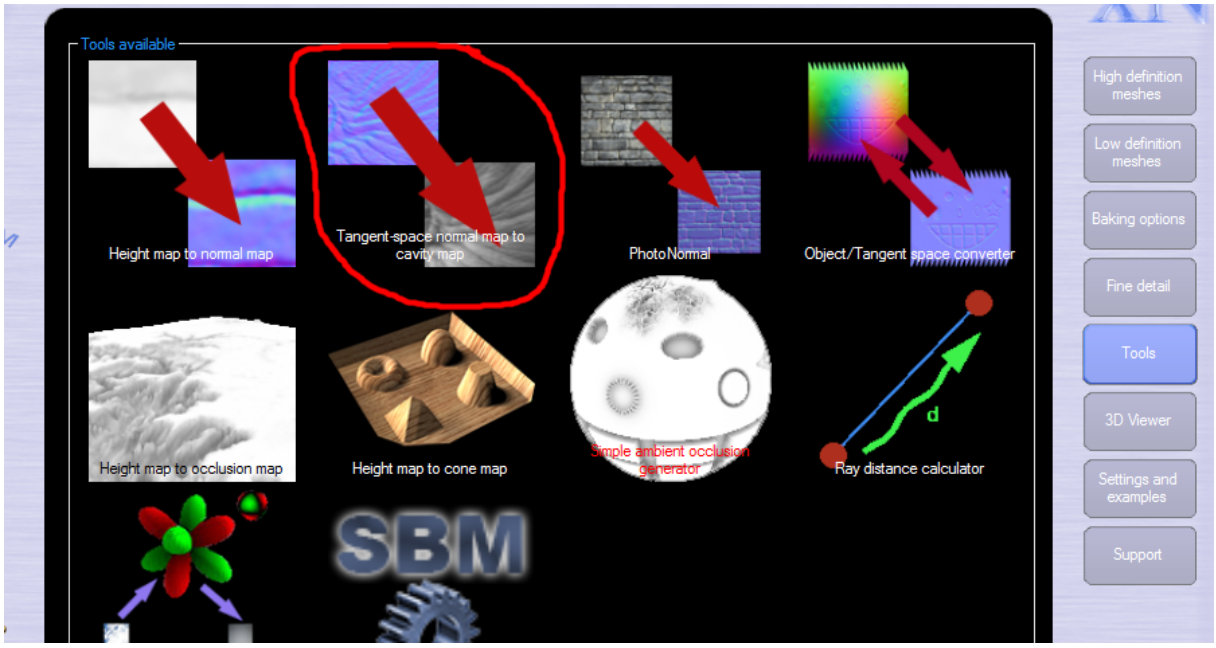


Now, again, pressing the mouse right button a context menu will appear. Notice the “save normal map” item is enabled. It will be enabled only after you generate the normal map from the height map. Ok... Press there and save the resulting normal map in the format you want. If you don't like the results just play a few with the parameters.

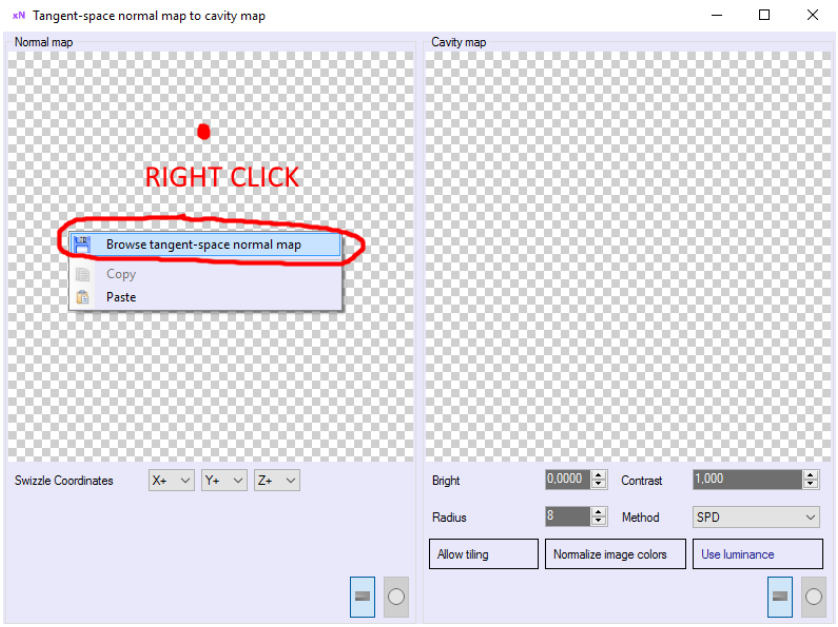
2.13 The tangent-space normal map to cavity map tool

xNormal comes too with other useful too to convert a tangent-space normal map to a “cavity map”. This “cavity map” can be used to enhace the appereance of wrinkles, porus or roughness of a texture. Basically is like a soft ambient occlusion map.

To open it go to the “Tools” section and press over the “Tangent-space normal map to cavity map” button:

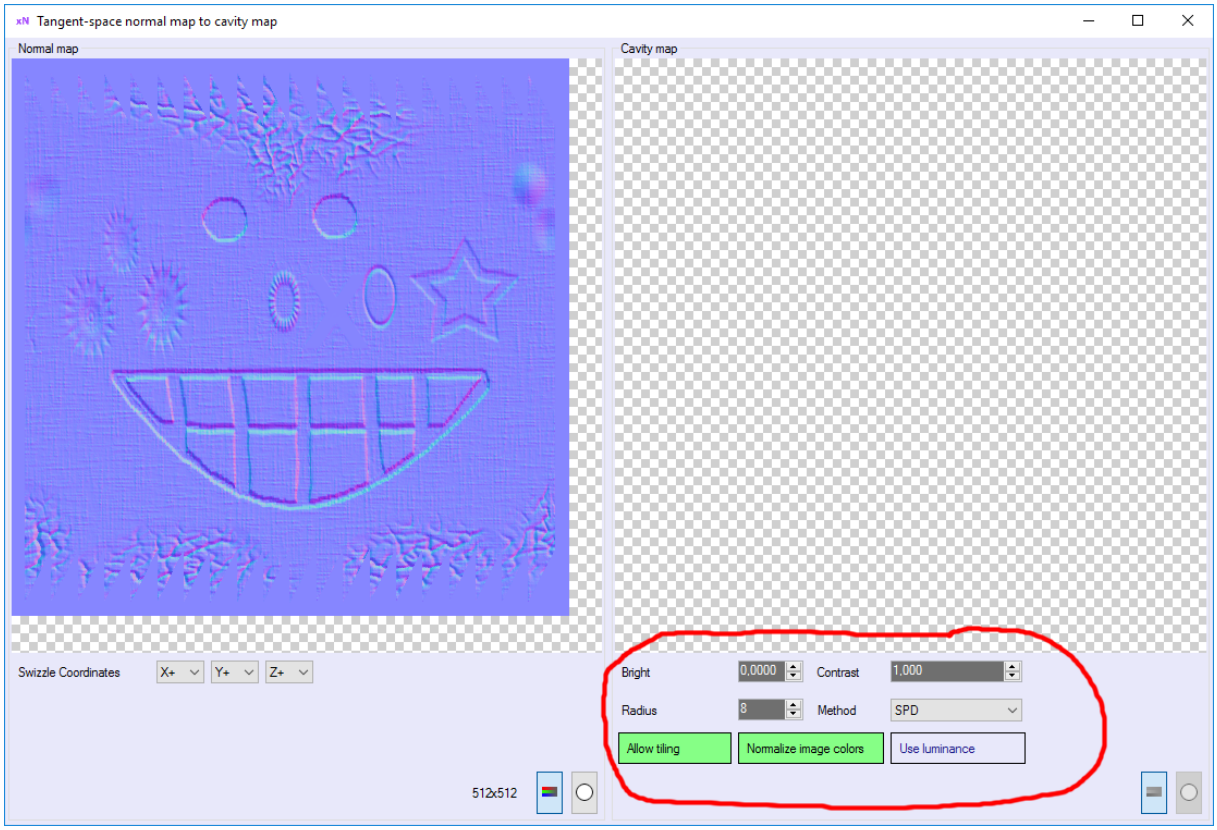


To start, go to the left checked panel and do mouse-right-button-click over it. A contextual menu appears and then you can select a tangent-space normal map texture:



You can swizzle the normal map coordinates to transform into any coordinate system and to show the RGB or Alpha chanels in the normal map texture.... As you can see the mechanism is very similar to the previous tool.... Easy.

Ok, now let's check out the cavity map generation parameters:



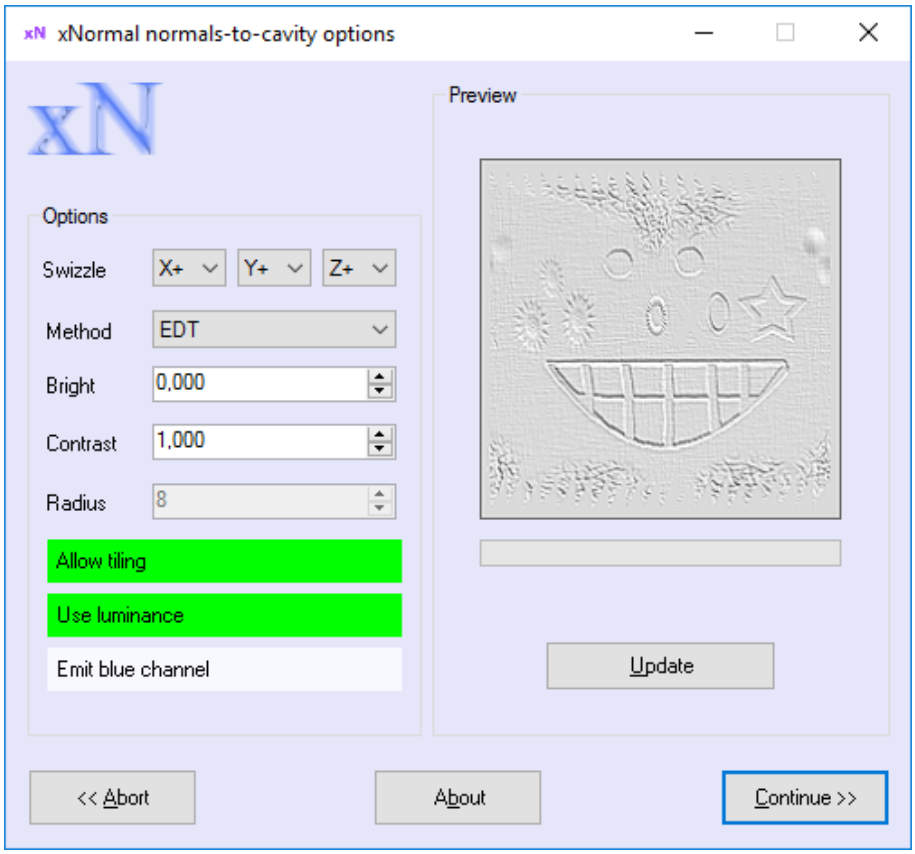
The “Bright” will add a constant offset color to the result, so be very cautious with it our will make your cavity map completely white! (due to color saturation)

The “Contrast” controls how “deep” will be your cavity map. A greater value will make the result much more pronounced and abrupt. Less will make it softer and smoother.

You can control if the algorithm used will consider the opposite texture pixels once it reaches a texture border with the “Allow tiling” option.

The “Use luminance” will make the algorithm used to generate the cavity map to use “color luminance”(intensity) instead of the default “average RGB” values. Usually this option makes the cavity map a bit cleaner.

Finally, the “Normalize image colors” will put all the resulting pixels on the cavity map in range [0.0f,1.0f], so will auto-adjust colors in a way the color spectrum will be better used.



As you can see, the options are similar to the xNormal built-in tool.

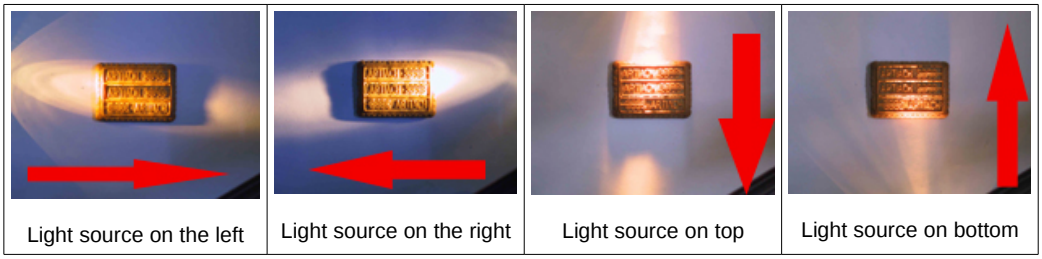
Just press over the “Update” button to see a preview if you change any parameter.

Be cautious with the “Bright” value if you select the “Emit blue channel” or will make the result too white (due to color saturation)... If you use the “Emit blue channel” option I recommend you to set the “Bright” to zero.

2.14 The PhotoNormal tool

This tool can convert four photographs into a tangent-space normal map. You need to take an object, a digital camera and a light source. Then you take a photo of the object illuminated from the left, from the right, from the top and from bottom and xNormal will mix then into a normal map!

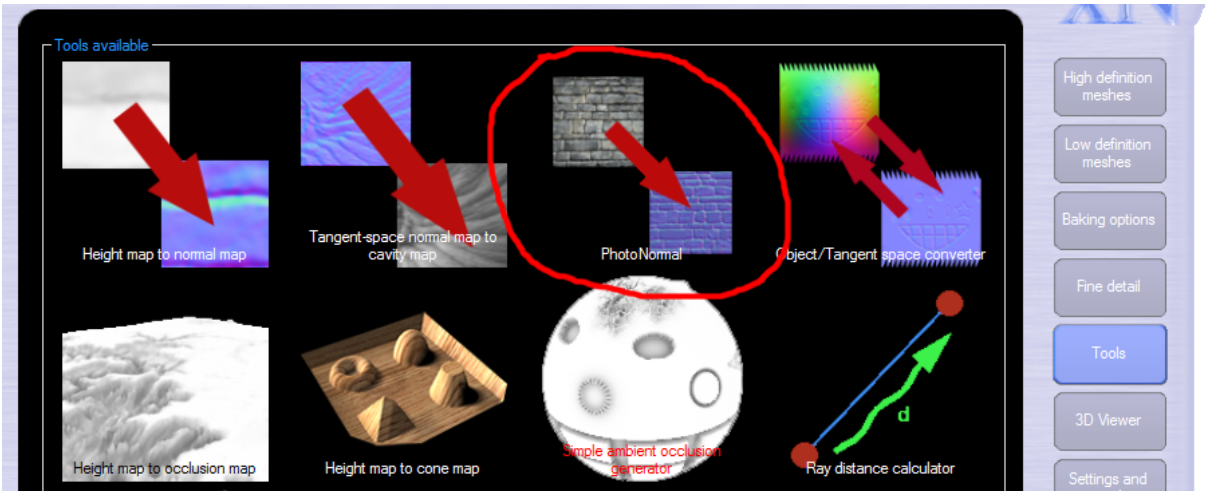
See, this is what we wanted to capture.. a cookie! yum , yum!



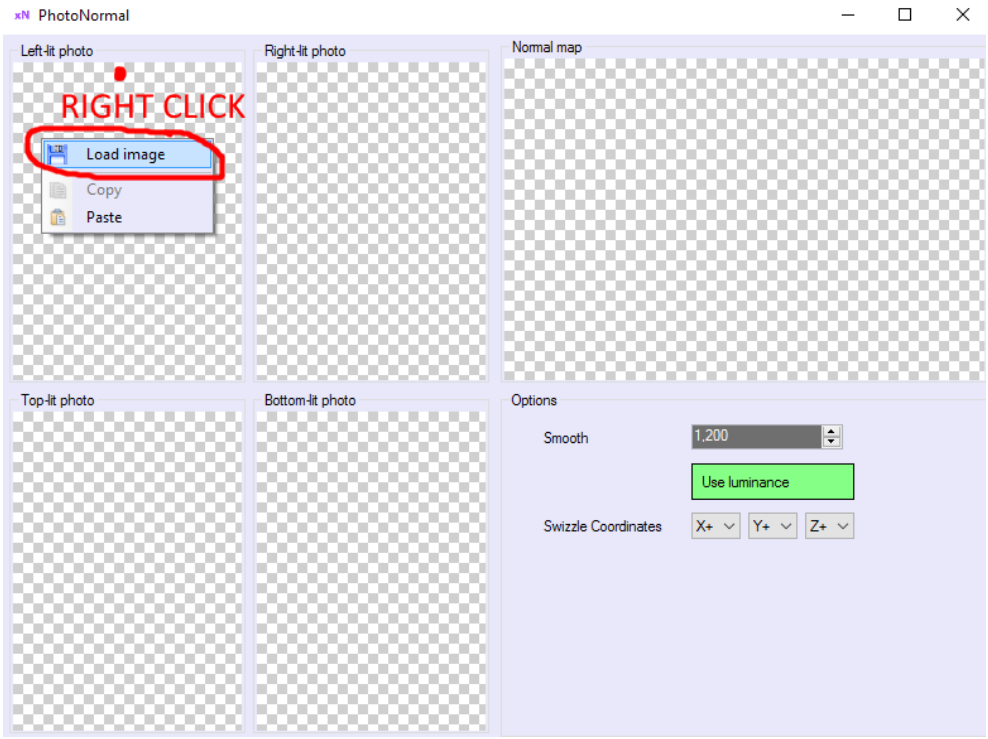
Light source on the left	Light source on the right	Light source on top	Light source on bottom

To get better results you need to use a "box-rectangle" light source, not a circular/ellipsoid one like I use but well... Also I moved a lot the camera due to my rabbit pulse! The ideal situation is to use a long fluorescent light source and use a tripode to avoid camera movement.

Ok, once you grabbed the photos go to the "Tools" tab and press over the "PhotoNormal" button:



Then this window will appear:



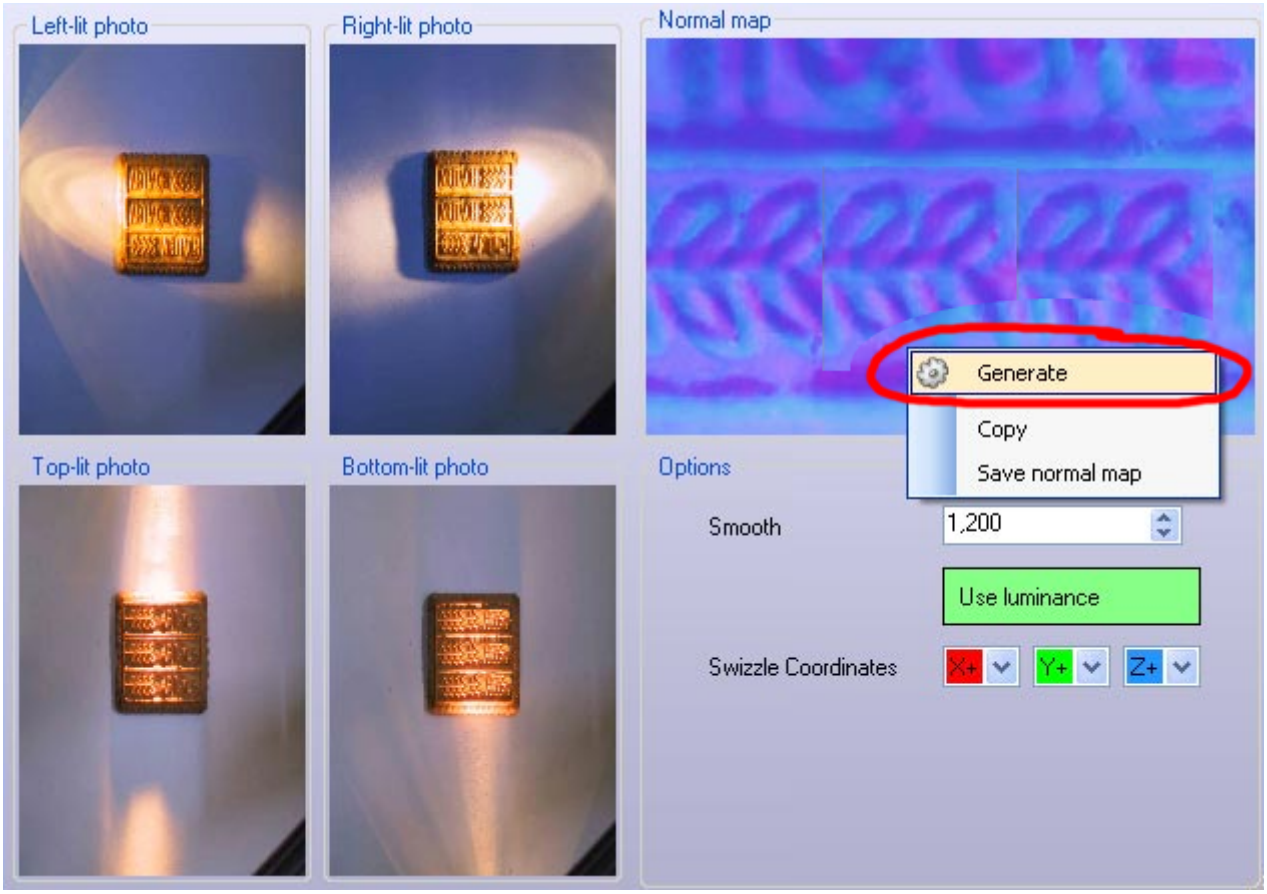
Load/paste the left, right, top and bottom light photographs.

The "Smooth" controls how much "abrupt" will be your final tangent-space normal map. A greater value will make it more "blue".

The "use luminance" option will be used to take the image intensity to compute the normal map. Usually activating this option produces better results.

The "swizzle" option is to transform the final normal map to other coordinate system. For example, if your 3D engine requires to flip the Y axis (== invert green channel) of the normals (use in this case X+Y-Z+).

Ok, once you set all the options and the lit photographs go to the "Normal map" checked panel and press the right mouse click. A context menu like this one will appear. Press over "Generate":



The tangent-space normal map will be generated. Then, you can copy it to the clipboard or to save it in a file.

Very easy!

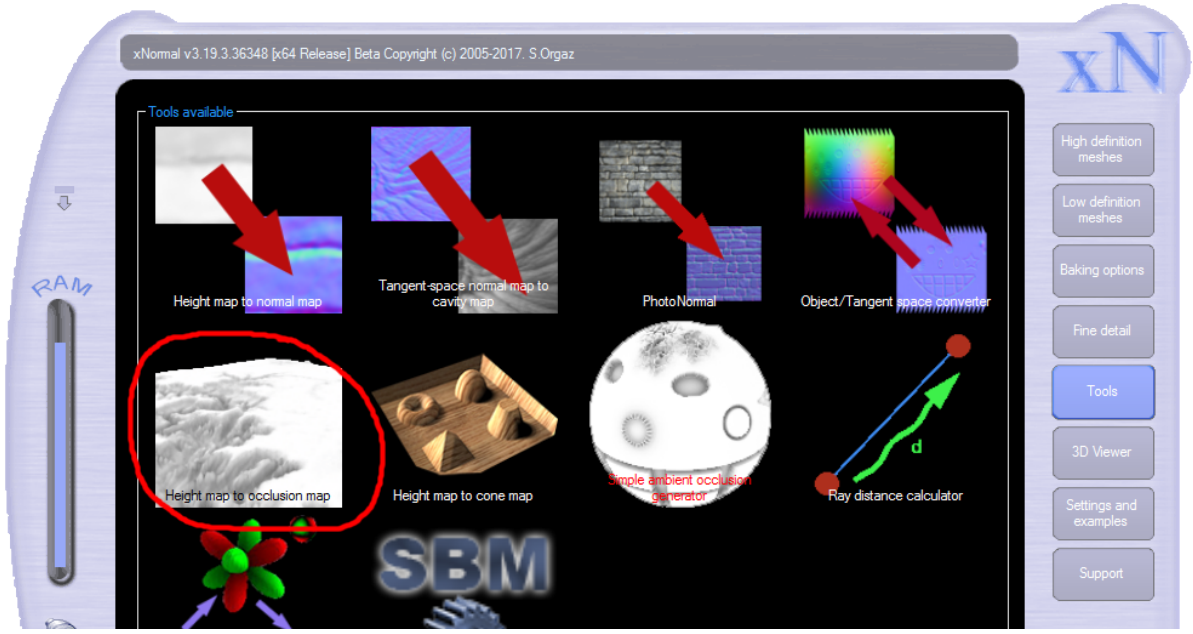
2.15 The height map to occlusion map tool

This tool can calculate a cheap and fast occlusion map from a height map.

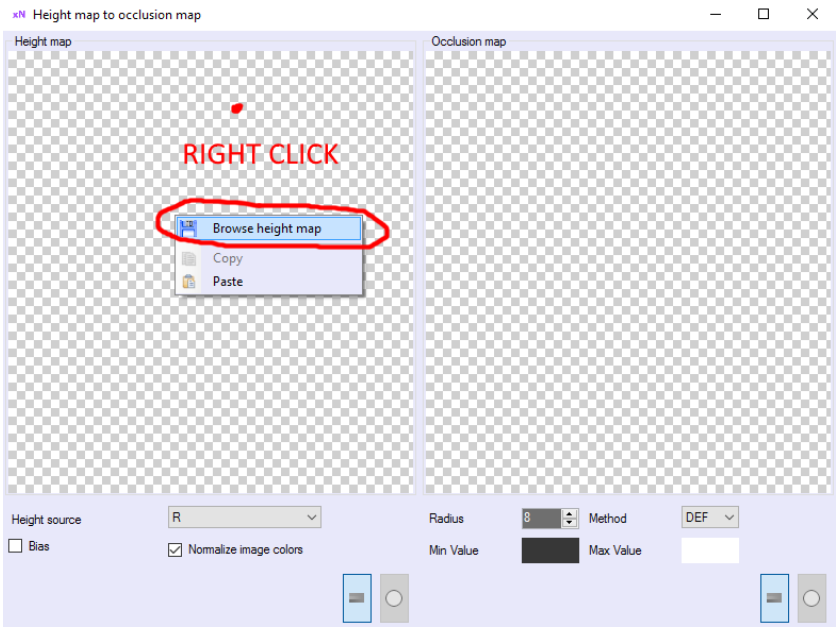
Can be useful to simulate a "cavity mapping" for wrinkles. Is very similar to the normal map - to - cavity map one, but taking the heights map instead the normals.

You can use it too in order to calculate a lightmap (without direction) approximation for a terrain.

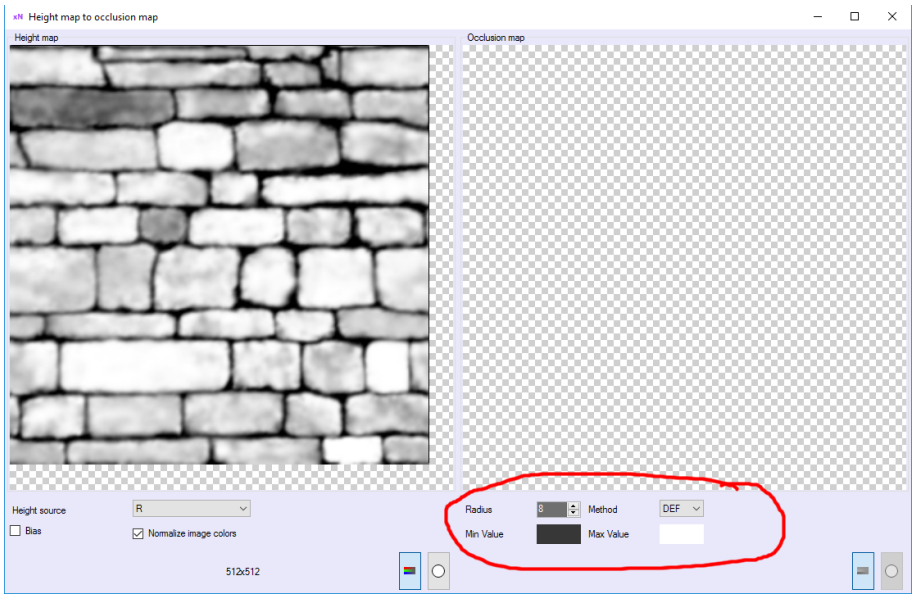
Lets see how to manage it ... Go to the "Tools" tab and press over the "height map to occlusion map" button:



The first step is to make mouse right-click over the "Height map" panel to show a context menu. Then we can load a height map from disk or to copy an image from the clipboard.



After we have set the height map there, we should select the "height source". We can set the R, G, B or Alpha channel or a more advanced combination like "color bias", color average etc....



Checking the “bias” option will scale the image components from range [0.0f,1.0f] to [-1.0f,1.0f]. Use this in case you have a signed height map.

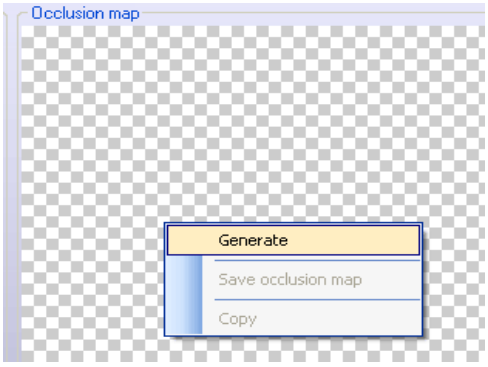
Remember that with the small - three RGB bands and white circle buttons we can see the RGB and Alpha channels of the loaded image. Also, if the image has mipmaps, we can see the mipmap levels with those "V" and "^" buttons.

The "Radius" controls the distance of the occlusion search. A greater value will make the occlusion smoother but can blur it too much. Usually 6-8 is fine.

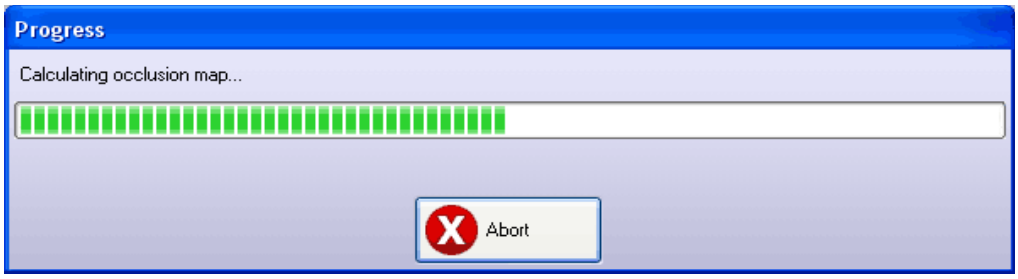
The "Min Value" and "Max Value" controls the "output levels". If you set the min value greater then the final result will be lighter.

The “method” controls the algorithm used to convert. The “DEF”(default) will find neighbor shadow contributions and the “LON”(Lonesock) can calculate the bent normals in the RGB channels + smooth ambient occlusion in the alpha channel.

Ok, now press the mouse right button over the "occlusion map" panel and other context menu will appear:

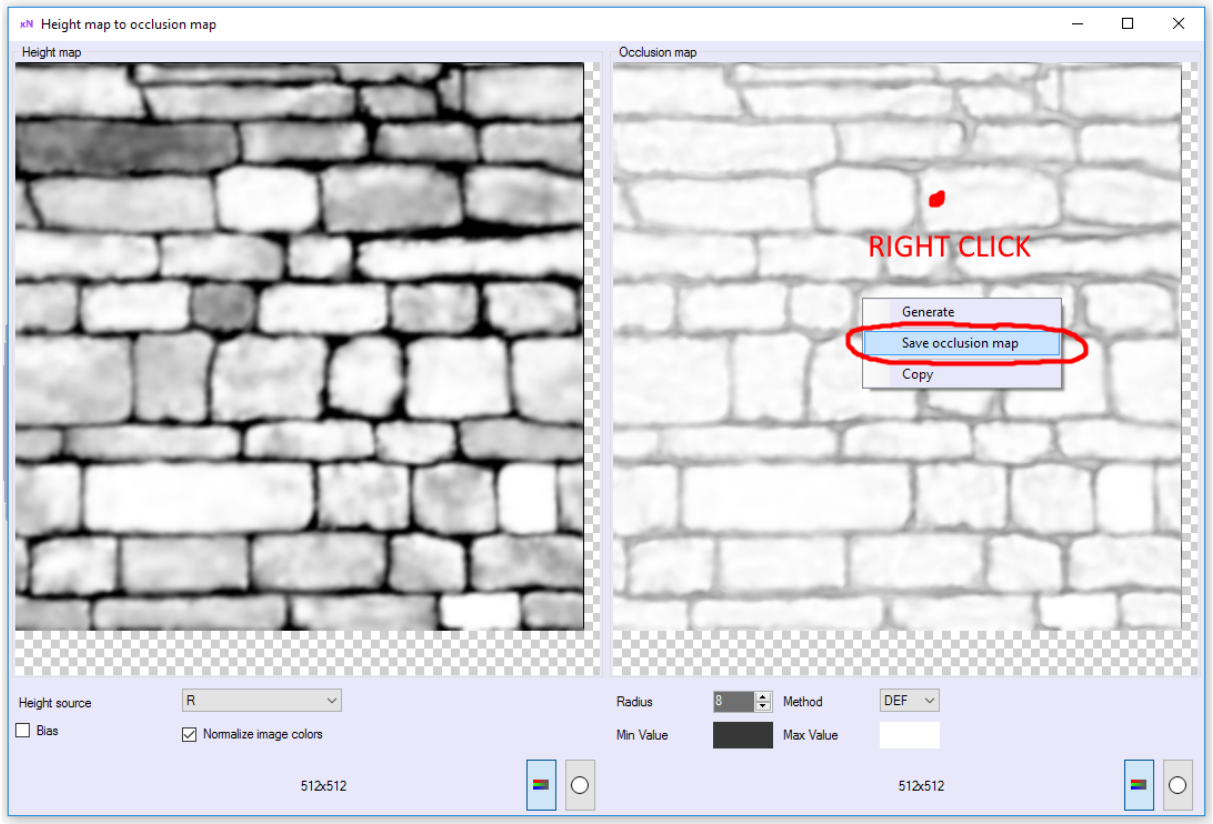


Notice the "save occlusion map" and the "copy" options are disabled because we didn't generate the map yet. Ok, press the "generate" and a progress dialog will appear:



You can abort the generation process if you need or is taken too much time.

Ok, after that we will have our occlusion map ready to be saved or copies to the clipboard:

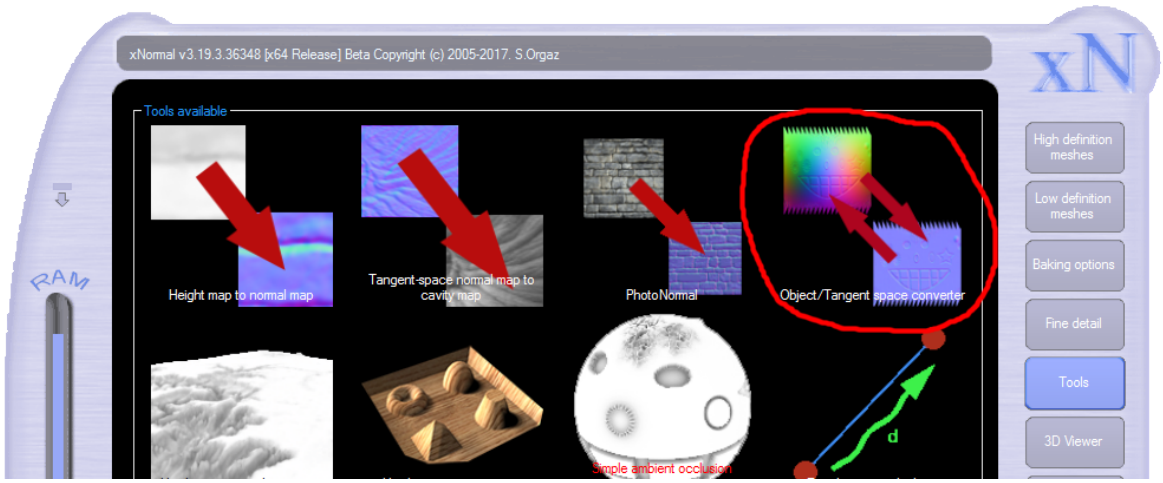


If you press over the "Save occlusion map", a dialog will appear asking you for the file path where to store it.

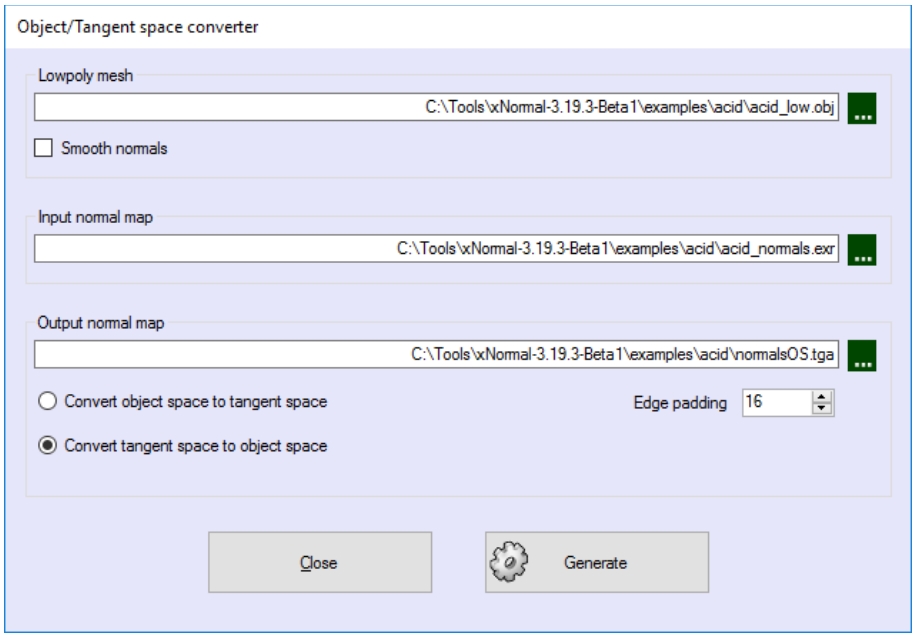
Pressing over the "Copy" the image will be copied to the clipboard and you could paste it after.

2.16 The object/tangent space normal map converter

xNormal includes a tool to convert between object and tangent space normal maps. To open it just go to the "Tools" tab and press over this button:



Then this dialog will appear:

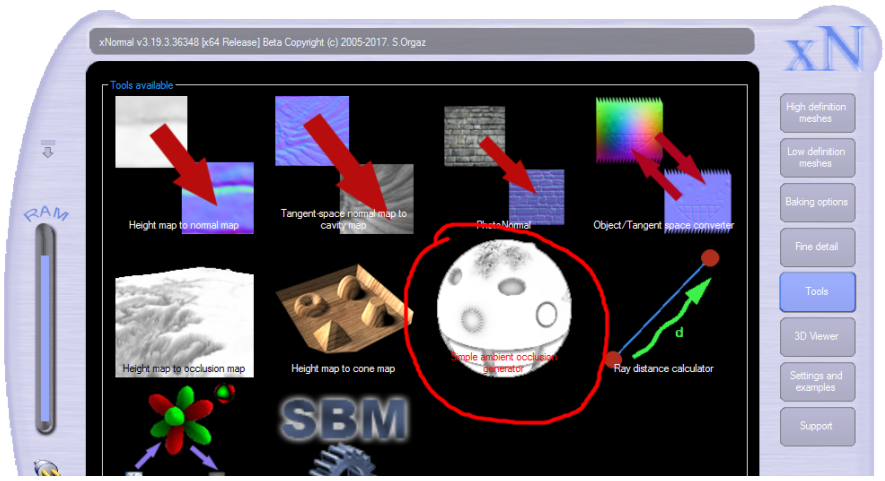


Just select the lowpoly mesh you used to generate the normal map and the input/output normal maps. Select if you want to convert from/to object or tangent space and click on the "Generate" button.

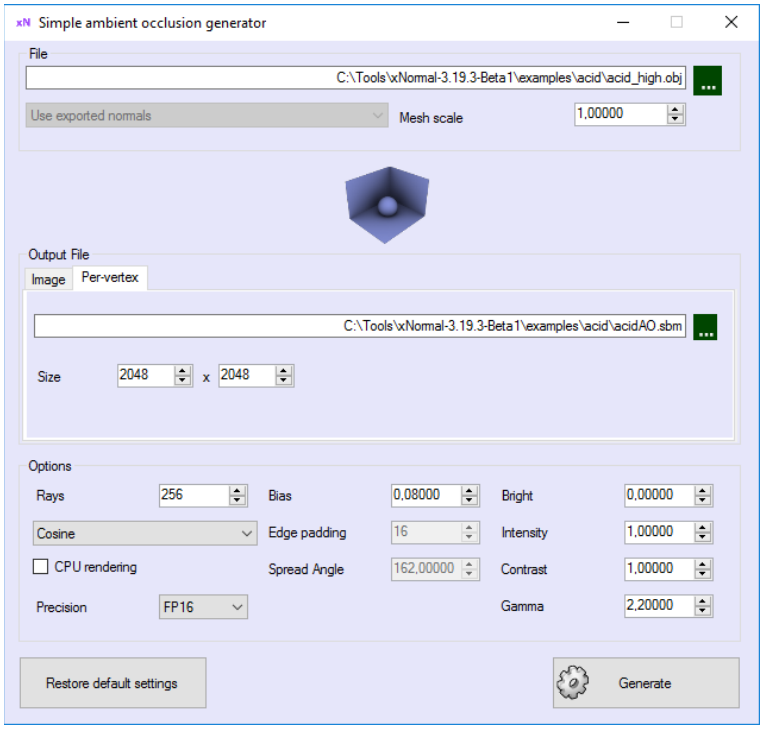
2.17 The simple ambient occlusion generator tool

xNormal usually allows you to use a lowpoly mesh + a highpoly mesh (or lowpoly mesh + a normal map override) to bake the AO, that process can be very slow due to the raytracing projection complexity. With this tool you can bake the AO map or to calculate per-vertex-AO for any mesh **without** having to perform raytracing and neither projecting highpoly mesh over the lowpoly mesh.

To open it, go to the “Tools” tab and press over the “Simple ambient occlusion generator” button:



Then this dialog will appear:



The first thing you must do is to select a mesh file (acid_high.obj) in this case.

Then decide if you want to bake an image AO map (OutputFile - Image) or to render the AO into vertex colors (Output File - Per-vertex). To render AO into an image the input mesh file you selected must be UV-mapped. On the other hand, if you want to render the AO into vertex-colors the input mesh you selected does NOT require to be UV-mapped.

The “Rays” option controls the quality of the AO... more rays are better. To make fast tests use 64. For medium quality use 64-256. High quality 512. Film quality 1024+. Remember the time required to compute the map scales linearly with this parameter.

The “Bias” parameter can be used to eliminate self-shadowing artifacts. Increase a bit this value if the self-occlusion gives you problems. Usually a value of 0.08 is enough. Too much bias can cause the occlusion to loose detail(will be too “white”)... too low bias can produce ugly banding self-occlusion artifacts.

Use the “Bright” parameter to add directly a scalar value to the occlusion map result.
Use the “Intensity” parameter to multiply the occlusion map by a scalar.
Use the “Contrast” parameter to make the shadows darker and the unshadowed parts brighter.
Use the “Gamma” parameter to raise the occlusion map to a scalar value.
The complete equation is result = pow ((1-occ*contrast)*intensity, 1/gamma) + bright

The “Enable weighting” parameter makes the AO map to be modulated with the dot product of the vertex normals and the AO direction. Enabling it makes the AO map smoother and a bit more lit. Disabling it makes it rougher and usually a bit more darker.

The “Padding” controls the edge “dilation” to prevent seam artifacts. Usually keep it at 2 or more. Increase it if you see seams in the rendered bitmap. Notice this option will be enabled only if you are rendering the AO to an image (not to vertex colors).

The “Precision” controls the accuracy of the AO gradients. This tool uses full graphics card **GPU**[7] acceleration using OpenGL. Some graphics cards does not allow to blend floating point maps, so OpenGL will enter into a super-slow software emulation mode...

Notice the CPU mode will work on any graphics card, but can be slower than the FP16 or the FP32 mode. If you set the FP16 or FP32 mode on a graphics card that does not support floating point blending the task will be completed, but will be very slow!

To finish, remember the mesh you select must be UV-mapped. If the mesh you select does not have UV texture coordinates a dialog will appear and the process will stop.

As you can see this tool is very easy to use and it can render the ambient occlusion much faster than the typical AO baking using the “Generate maps” process. However, has some limitations like the impossibility to project the highpoly mesh detail over the lowpoly mesh (you could use the base texture bake to solve this) or to limit ray distance using cages, cannot use distance attenuation, has high-end GPU requisites for full speed and the need for an OpenGL GLSL-compatible card (shader model 2 or above).

7 **GPU**. Graphics processing unit.

3. Known issues

3.1 Known issues table

- Very small objects (radius less than one unit) can cause problems due to floating point arithmetic epsilon error. Please scale up a bit if you get incorrect results.
- Sometimes the “xxx rays failed” indicator in the preview dialog indicates some missed points in your normal / displace or occlusion map. That's due to floating point imprecissions (or because you set bad the maximum ray distance). Use the “antialiasing” and ”blur”/“dilate” filters to mitigate it.
- Percentages for some actions stall or suddenly change or the user can't abort the process in the middle. Well... I know this, but is very difficult to control some actions, specially when are based in external routines or libraries. Also, if you specify a only-2-triangles-lowpoly-mesh and those triangles are very large, the normalmap progress indicator will do only two steps.
- The abort procedure does not stop immediatly. To ensure application stability can't interrupt suddenly some processes, so you usually need to wait one or two seconds until the operation finishes.
- Windows 98/ME/XP/Vista is NOT supported anymore.
- Only the smiley, parallax wall and flag examples include a high polymodel model mesh to generate the normal map. This is because the xNormal package should remain compact to be downloaded well and I can't include the 450Mb highpoly .OBJ model for each example...
- Multiple monitors are only supported if are plugged into the SAME graphic card. If you use N monitors with multiple graphic cards the interactive 3D viewer could not share device contexts and probably will crash. This program is NOT tested on more than TWO monitors... feel free to make a donation so I can buy some extra monitors, please!
- Multiple UV maps inside the same mesh file can cause problems (specially in LWO and XSI files). Please sure you ONLY export ONE before passing the file into xNormal.
- To drag/move the xNormal window by the screen click on the top-right xN blue logo. Keep left mouse button pressed and move the window.
- The highpoly model shader in the 3D viewport does not include specular lighting. The highpoly models are rendered in realtime with smoothed vertex normal(even when the model has hard edges). This is not a bug, is just to draw it faster.
- Using large meshes / textures can make your graphics card to be out of VRAM. See the “VRAM used” inside the viewer to know how much approximate video memory is used at the moment.
- Generating large maps (>2048) can wipe fast your system RAM memory. The normal/AO/height maps can occupy in memory WIDTH x HEIGHT x 20 . See the system RAM indicator on the left (in the .NET UI) to know how much physical RAM is available. If you see the indicator very low then the virtual memory (disk memory) will be used and the system will slow a bit (the hard disk will start to swap memory).
- Some graphics cards with FSAA enabled don't perform well vertex picking. Choose a video mode without FSAA or enable the windowed mode if you can't select well cage vertices with the mouse picking.
- The .DAE mesh importer only supports triangle meshes at the moment. It neither supports multi-material objects.

- The right-to-left languages support are still under development ...
- Some plugin progress information (like the “Loading OBJ mesh XXXX...”) are still not translated. This is because the DLL plugins can use their own display language and the current locale one.
- The screen can flick or refresh the dialog very slowly when you use the heightmap - to - cone map tool or the simple ambient occlusion calculator... that's normal... is using the GPU to compute a lot of things so can be unresponsive sometimes.
- If you render a “wireframe and ray fails” for a very dense lowpoly mesh, it will take AGES.
- xNormal can output raw floating point heights but it expects all the INPUT heightmaps normalized in range [0.0f,1.0f]. So, if you use aheight map, for instance, as fine detail map, please sure it has been previously tone-mapped into a [0.0f,1.0f] range... and remind that xNormal will bias it internally to [-1,1] to use emboss/hills properly.

4. Troubleshooting

4.1 Generate map problems

Q: My normal map is rendered transparent, with no pixels, what can I do?
A: Sure your highpoly model and lowpoly model are aligned and have aproximately the same scale. Also sure your models have almost 0.5 units of radius to prevent floating point problems. Try to check the “Use closest hit if ray fails” in the normal map options. Try too to measure a good ray distance inside your favourite 3D modelling application or use the xNormal cages method.

Q: My normal map is well generated but when I see it in the 3D viewer appears like “Oren-Nayar” shaded like velvet and strange...
A: This can be because you generated the normal map in tangent space and you forgot to check the “Tangent Space” checkbox for the 3D viewer. Sure if you generated the normal map using tangent space then you select the “Tangent Space” checkbox in the 3D viewer tab, near the normal and displacement map texture file name...Notice there are TWO “Tagent space” checkboxes... One in the normalmap tab and other in the 3D viewer options.

Q: I am trying to specify a heightmap to add fine detail to the normal map but it appears very weak, I can barely appreciate it.
A: Try to touch the “scale factor” and “blending factor” in the fine detail tab. If this don't solve it then please notice the heightmap or bumpmap you specify for detail needs to be **signed**[8]. Also, try to use a **normalized**[9] bitmap. If you are using HDR formats sure the alpha-premultiply and the exposure all well set.

Q: I am getting a normalmap like if the highpolymesh doesn't exist... The normal map's normals are exactly the lowpoly model ones...
A: Try to align your lowpoly and highpoly models BEFORE exporting them. Both models must be placed aproximately in the same position. Don't mix formats... If you use FBX files use FBX for lowpoly and highpoly model.

8 **Signed**. Using 8bpp 0 means complete emboss, 127 flat and 128+ elevation.

9 **Normalized.** The bitmap must use 0 as low color and 255 as max color.

Q: I need to manage a very very very high polygon model or a very large normal map. I run out of RAM (vieweing the RAM indicator on the left) and an error appears while generating the maps. What can I do?
A: Some advices:

1. Perform a pre-triangulation in the modelling program you used to create the mesh. xNormal performs triangulation when it loads the models and that needs extra RAM. If you export triangulated models then xNormal does not need to perform the triangulation process so less memory will be used.

2. Set the dilation(edge padding) pixels to zero. The dilation filter needs to copy internally the bitmap to work. If you set it to zero then no dilation will occur so less memory will be used.

3. Export the model without normals. In that way xNormal will average the normals on the fly saving some memory (this is also good to reduce the model's disk size).

4. Try to keep the output map into a reasonable size. That means a normal map between 512 and 2048x2048 is ok. A normal map of 8192, although allowed, will consume an untolerable amount of RAM. Sometimes is better to render a 2048x2048 map with 16AA than a map of 8192 and the reduce it to 2048 using supersampling.

5. Sure your virtual memory file is not limited. Let Windows to adminitrate it automatically. If you set its maximum size then it cannot grow if needed.

6. If you use a x86 operating system, let me remind you that the theorical physical limit is 4Gb... but the practical one is 2Gb. Better use a x64 operating system!

7. Render your model by parts (aka “explode meshes” technique).

8. Try to use BMP or TGA as output file format. The other formats require more memory to save the data to the disk.

9. Buy more RAM.

Q: I want to reconstruct the highpoly model in my fav modelling app using the displacement map that xNormal generates but it doesn't work... looks too rough... Why?
A: You want to use the lowpoly model as subdivision base mesh. Then you subdivide this model. Then you sculpt it using those programs. The displacement map that you need is the distance of the subdivided model and the sculpted one. xNormal displacement map is a different thing... at the moment it just calculates the distance between the lowpoly and highpoly models, so that is NOT the displacement map you need.

I'm working in a subdivision algorithm for the lowpoly model) . Meanwhile, use the subdivided (but not sculpted) highpoly model as lowpoly model. Use as highpoly model the highpoly model scultped in you fav app... But be warned... this only will work if you DON'T need to use cages... because the cage editor inside the 3D viewer is NOT prepared to draw well one million polygons...

Q: I am using external-defined mesh cages. I don't know why but the point-to-point line indicator does not go to the correct vertices and my normal map once generated shows tons of thrash. Why?
A: You edited the lowpoly model. If you touch ONE vertex in the lowpoly its topology changes completely and you will need to re-do/re-export the external cage because the vertex indices won't match anymore. Keep in mind that if you delete/create a vertex or a face in your lowpoly model you will need to re-do/re-export completely the mesh cage!!!

Q: The GPGPU renderer makes the UI very slow to respond. Why?
A: There might be not enough resources to paint the operating system's windows. Don't worry, the UI will become as fast as it should be once the render has finished. As workaround, get a second GPU and use it exclusively to compute the maps.

4.2 Import/Export problems

Q: I see incorrect normals in the lowpoly model. When I launch the 3D viewer and turn on "show lowpoly tangent basis" normals in the tangent basis axes only come up and down. Also then I turn "show lowpoly normals" I see only two tones for the normals (usually purple and green).
A: Some old exporters can be old and unmaintained. Try to re-export in other format (like 3DS or ASE) and re-try. If problem persists, try to open your lowpoly model file with an hexadecimal/text editor and see if the normals are really well exported. See the chapter named "**Problems with lowpoly normals**" in this document.
Also, try check/uncheck the "Use smooth normals" options in the lowpoly model table.

Q: Can you give me any advice about exporting models?
A: Try to weld very close vertex positions and texture coordinates. Use pixel-snap when you texture-map your model and don't overlap coordinates. Try to use only one material for all your model faces. Remove any degenerated (zero area) polygons and duplicated vertices. Avoid T-junctions. Set well the vertex normals. Sure the data is well exported before you pass it to xNormal.

Q: Can you give me any advice to load faster the models?
A: You can import you model and then re-export it in a better format. For example, load you model exported from your favourite 3D modeling program. Then load the 3D viewer and re-export it in a format like the Simple Binary Mesh(SBM) or the OVB format. These formats are optimized to work faster in xNormal.

Other advice is not to export UVs if you don't need them. For example, you don't need UVs in your highpoly model unless you're using the "bake highpoly base texture into lowpoly model" feature.

Exporting normals with your model will reduce the load time because won't need to be calculated on the fly (but you model will occupy much more disk space so it could be worse in certain cases)

4.3 3D Viewer problems

Q: The launch viewer button is always disabled so I can't launch the viewer, why?
A: Will be disabled if your graphics card doesn't support some required feature to run the viewer or no good video modes are available. ALWAYS keep your graphics drivers updated.
Try to check the “Run Windowed” option. If this don't solve your problem try to edit the xNormal_debugLog.txt file located in the Documents and settings\[Your user name]\Application Data\xNormal folder with the notepad. This file contains some logs about errors and warnings. Read it and see what happens.

Notice the button will be disabled while generating the maps. When the process stops then the button will be re-enabled. Notice too the button will be disabled if there are no “visible” meshes in the highpoly/lowpoly list tables.

Q: When I select and use a fullscreen video mode with antialiasing some text appears “blurred” in the 3D viewer...
A: Some graphics cards can't disable FSAA dynamically. Although I specifically try to disable FSAA for writing text, some graphics drivers will ignore this and will antialias all the screen.

Q: The performace inside the 3D viewer is very bad, specially when I try to edit the ray cages.
A: Effects like the parallax or the glow effect can be expensive. Try to disable them. Try to disable fullscreen anti-aliasing too.

Notice the viewer can enter into “software mode”. The graphic card can to pass to this slow mode if your graphics card doesn't support a non-critical feature. See inside the xNormal_debugLog.txt for any comment about “software mode”. If you find it then try to use less textures, specially the reflection one. Try to assign only a base + normal map texture. If problem persists then you need to update your graphic drivers or buy a new and more powerful video card.

Q: With fullscreen antialiasing I can't select any cage vertex. I define a 2D rectangle with the mouse to multi-select vertices inside the rectangle but no vertices are selected once I release the mouse button.
A: Some graphics cards don't allow to “lock” (VRAM-to-system RAM operation is not allowed) multisampled framebuffers, so xNormal can't use the depth buffer values to perform the “Ignore backface selection”. You have two options here... One is to uncheck the ignore backface option (sometimes this is acceptable because you have tons of vertices hidden by the geometry you don't want to select). The other is to select a video model without antialiasing and then problem will be solved.

Q: When I enable the shadows and I try to move the light the computer halts a few and then shadows are updated...
A: I only recalculate shadows when it's needed... Only if you move the light... This takes time because xNormal has to make a lot of things... Shadows are not cheap at all... The are slooooooooooow, so if you notice a big perfomance hit better disable them in the 3D viewer.

Q: When I enable the shadows and I put the light too far from my models no shadows appear or they loose tons of resolution...
A: At the moment I am not using any kind of adaptive shadows. If you put the light too far from the objects no shadows will appear and they loose tons of quality, thats normal.

Q: When I enable the shadows and I put the light close to the objects, a black moire appears. How can I solve it?
A: Touch the “shadow bias” slider inside the 3D viewer. The “bias” parameter is designed to mitigate this effect. Need manual setup, sorry.

Q: I get an error when I launch the 3D viewer. Says something like “Can't call the UI lua file init() function: C:\xNormal\ui\ui.lua:837: attempt to call global 'xn_get_diffuseGI_intensity' (a nil value)”.
A: This happens sometimes when you try to upgrade xNormal from a newer install. Try to uninstall completely the old xNormal version and then re-install the new one.

Q: The 3D viewer shows an error loading the highpoly mesh that says “Unexpected error loading mesh into VRAM”. **How can I solve it?**
A: Some graphic cards don't allow to use meshes with more than 1M vertices/triangles. Try to split your highpoly model into smaller parts or check the highpoly model as invisible in the “highpoly model tab” so won't be loaded into VRAM and neither drawn.

Q: The 3D viewer ignores my vertex normals and specular lighting on the highpoly model, why?
A: To optimize, the graphics driver can ignore the vertex normals in the highpoly mesh. The highpoly model usually will be represented without texture, without specular lighting and with averaged vertex normals. Notice, however, that the lowpoly model should be painted with the vertex normals you specified. This optimization only applyes to the HIGHPOLY model.

Q: How can I resize the 3D viewer window?
A: First at all you need to be in “windowed mode”. If you unchecked this option and you are running in full screen mode you can't. Notice also the Direct3D graphics driver does not allow to resize the window currently. If you need to resize the window then select the OpenGL graphics driver as default graphics driver and re-run the 3D viewer in windowed mode... once your model is loaded and displayed into the window press ALT+ENTER to release the mouse cursor... go to a window's border and resize the window.... once you set the desired size press again ALT+ENTER to attach mouse back.

Q: The 3D viewer video captured .AVI is black, why?
A: Some graphics cards don't allow to capture video if you use full scene antialiasing. If that happens, try to select a video mode without AA. Also can be due to codec imcompatibility with the current video mode... try also to select other video codec or save the file as uncompressed frames.

Q: The 3D viewer video capture goes too slow, what can I do?
A: Select a video mode without full scene antialiasing, try to use a windowed mode and reduce the window size to the minimum (800x600). If you need to use a full screen video mode try to select one with small size or try one with 16bits per pixel instead a 32bits one.
You could also try to change the default graphics driver (use the xNormal plug-in manager)... usually the OpenGL is faster capturing video than the Direct3D one.
Finally, get better hardware (better CPU and GPU, more memory and a good hard disk).

Q: The 3D viewer does not show well the transparencies. I see like some kind of zbuffer issues in the transparent polygons. Why?
A: The viewer does not support transparency sorting and you need to play a few with the backface cull modes. Opaque, 1bit-alpha and additive objects can be drawn without problems but the others not because require really complicated methods to sort well the transparencies.

If you want to display screen-glass or alpha-blended objects sure you set well the desired backface cull mode and sure too you specify the meshes in the correct order in the lowpoly mesh list. Try to use closed-convex objects and enable backface culling so spheres, car crystals and transparent windows will be well painted. If you use non-closed/non-convex 2-sided objects you will have problems.

Q: Sometimes the 3D viewer shows two cursors and I loose the mouse control. What can I do?
A: When that happens running the viewer in “windowed mode”, press ALT+ENTER to release or to attach the cursor to the 3D viewer window. If the problem persist please send me a bug report indicating your system hardware, operating system, graphics card model/driver version and a detailed description about the problem.

Q:I use Vista and the DX10 graphics driver. I see all my model's texture “burnt”. Why?
A: Go to the DX10 graphics driver's options (go to the plug-in manager, graphics drivers and select the DX10 graphics driver. Then click over the “options” button). A dialog will appear with some options. Sure you disable “texture compression” and also the “use gamma correction”.

Also notice the DX10 graphics driver assumes that the input textures are saved in sRGB color space (gamma aprox 2.2).

4.4 Install and start problems

Q: xNormal crashes at start, why?
A: Install the .NET 4.0 framework manually. If you don't have it installed will crash when you try to execute the xNormal.exe file.

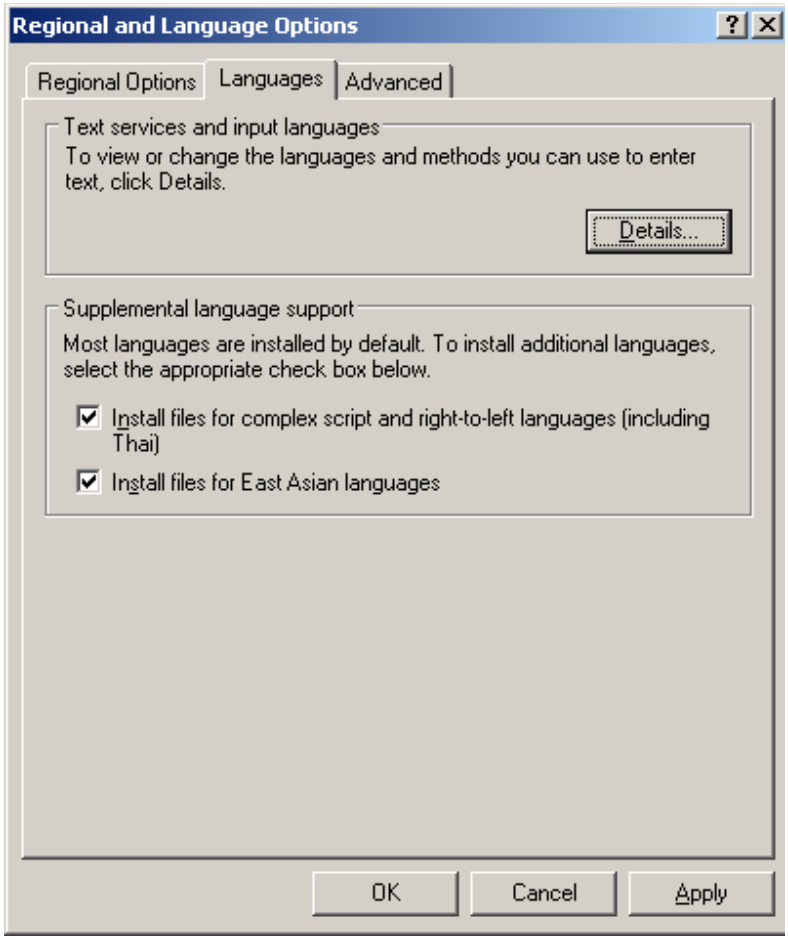
Also, try to go to the C:\Documents and settings\[Your user name]\My documents\xNormal. Then delete all the .XML files these (delete the files NOT the directory). Notice that folder name can change if you don't have an english version of Windows... For example, if you are using a spanish WindowsXP then the folder will be at C:\Documents and Settings\[Your user name]\Mis documentos\xNormal

If the problem persists then find in the registry (use regedit.exe) remove the entre “xNormal3” from **Current User -> Software.**

4.5 User interface problems

Q: I can't see japanese/chinese/arabic or hebrew characters but I selected any of those languages in the xNormal's plugin manager... why?

A: You need to install the appropiate Windows font for these. By default, occidental Windows versions usually doesn't come with asian/middle east fonts. To install them go to the **Startup->Configuration->Control Panel->Regional configuration and language->Languages** and check the “Install complex language scripts and right-to-left writting” and also the “Install asian fonts”.



5. Project history

5.1 Project history table

Here are the changes for all the xNormal's versions:

3.19.3c

- Now the SDK is offered as a separate installer.
- Added full 2018 plug-in support and partial 2019 support.

3.19.3b

- Added the SDK to the installer.

3.19.3a

- Added partial support for 2018 plug-ins.
- Fixed a problem in the installer that was assuming binary-compatibility for old version plug-ins and some error messages were displayed when the plug-ins were loaded.

6. Legal stuff

All the trademarks correspond to their legal holders.

Please, check out the xNormal_legal_unified.rtf in the xNormal's installation directory for more info about copyrights, licenses and other legal stuff.