

# **A Practical Guide On Normal Mapping for Games.**

by [Alexey Oshchepkov](#)  
([Superfranky](#))

## **Introduction**

This Guide is created for those, who want to understand Normal Mapping. It's written for newbies, who want to get into game art and for veterans, who want to freshen up on the subject. In this guide, I will try to address every possible problem, and provide a concise and easy-to-understand answer. I hope that this Practical Guide will teach you everything there is to know about Normal Mapping and its application in game art. Normal mapping problems should not get in the way of your art!

Thanks goes to EarthQuake from Polycount for inspiring me to create this guide.

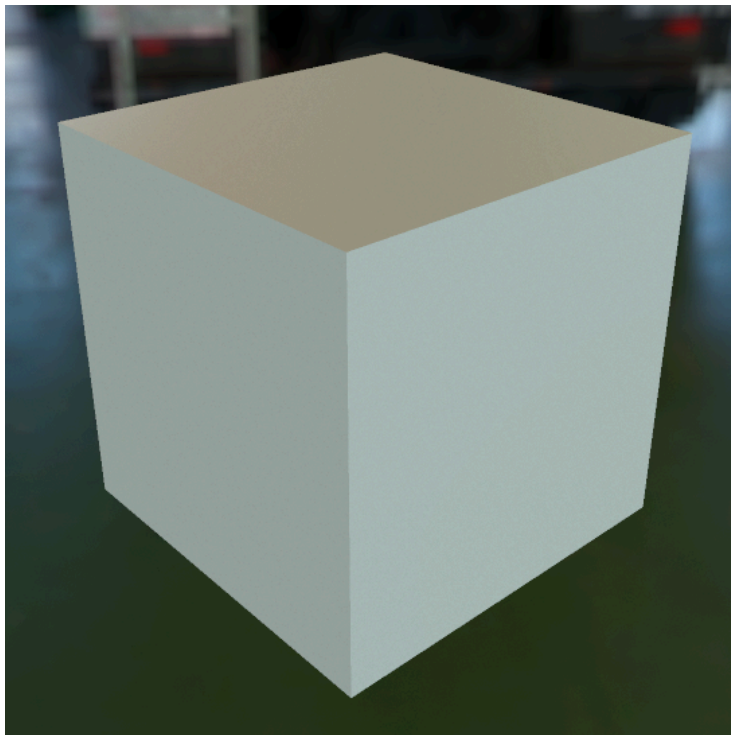
## **Terms of Use**

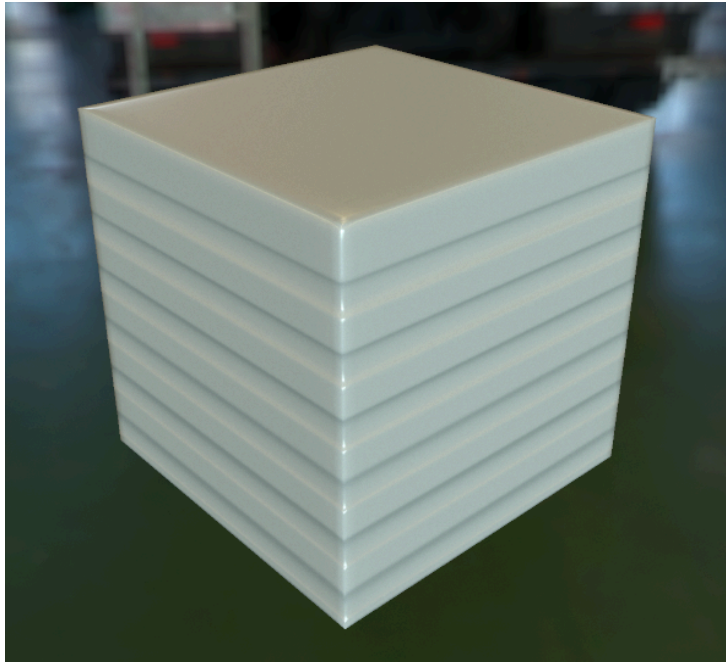
Please respect the work of the author and don't copy and paste it to websites other than Polycount without my([Superfranky](#)) permission. If you see a mistake or want to see something added, please PM me on Polycount or send me a message on [networkcat2@hotmail.com](mailto:networkcat2@hotmail.com).

# Tech babble

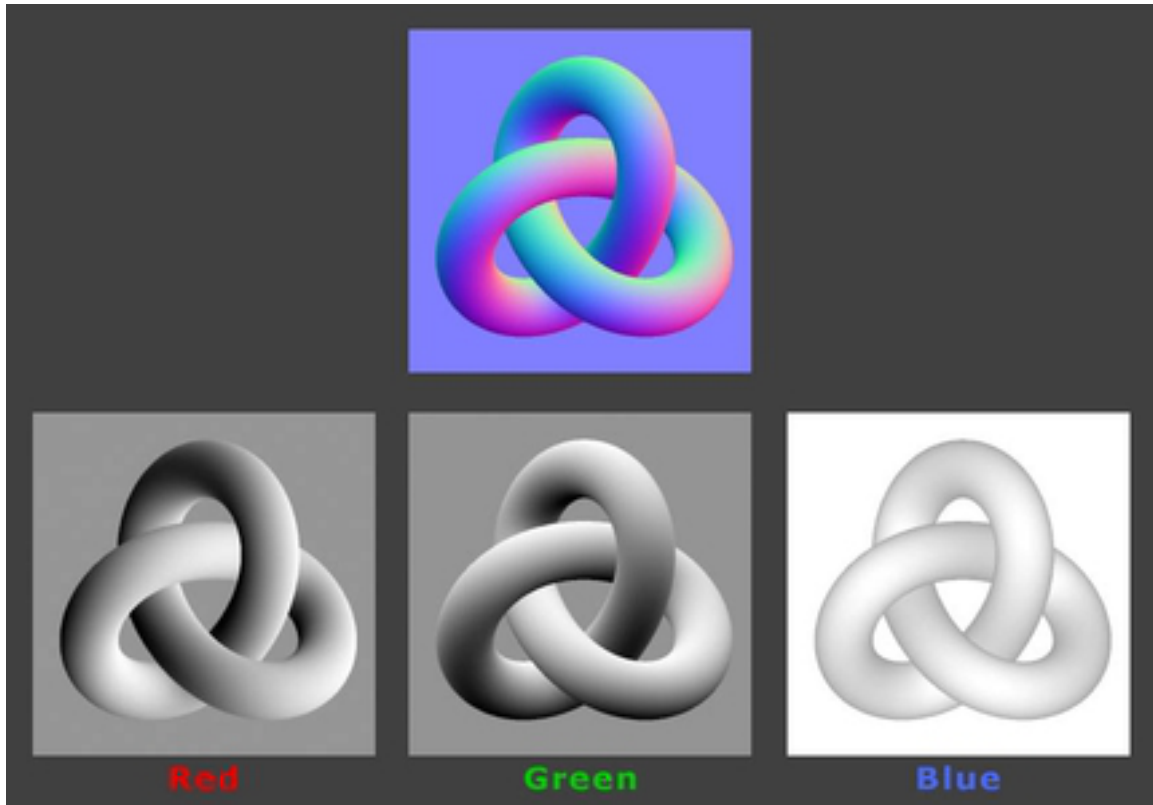
In this chapter, I will try and provide all the technical information related to Normal Mapping. I will keep it as simple as possible and will not provide any information that is not directly related to the process of creating game art.

Normal Mapping, in 3d art, is a technique that is used to fake the lighting of bumps and dents from the high polygon object. It is used to make your game model appear more like your high polygon model. It can be used to add various details that you can't possible model in low poly due to the strict polygon limit of your project or add smoothed edges to make your simple object look a bit rounded to better catch lighting and look more realistic.





Normal maps are usually RGB images where the RGB components (Red, Green, Blue channels) correspond to the X, Y, and Z coordinates, respectively, of the surface normal. Red channel of a tangent-space normal map stores the X axis (pointing the normals predominantly leftwards or rightwards), the green channel stores the Y axis (pointing the normals predominantly upwards or downwards), and the blue channel stores the Z axis (pointing the normals outwards away from the surface)



## Tangent Space

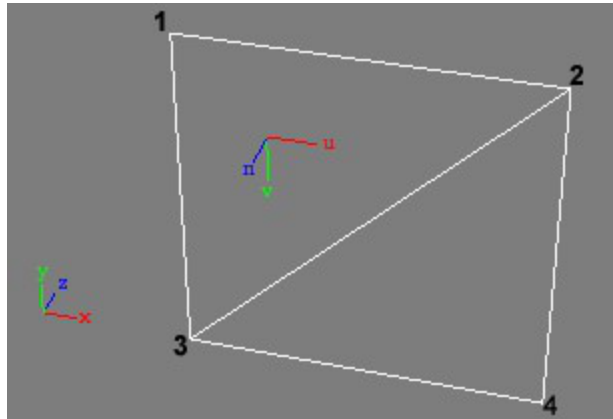
Before we get into any modeling and baking I have to inform you about Tangent Space and what it means.

The usual type of a normal map that you can see everywhere on the internet that looks all blue and pretty is called Tangent Space Normal Map.

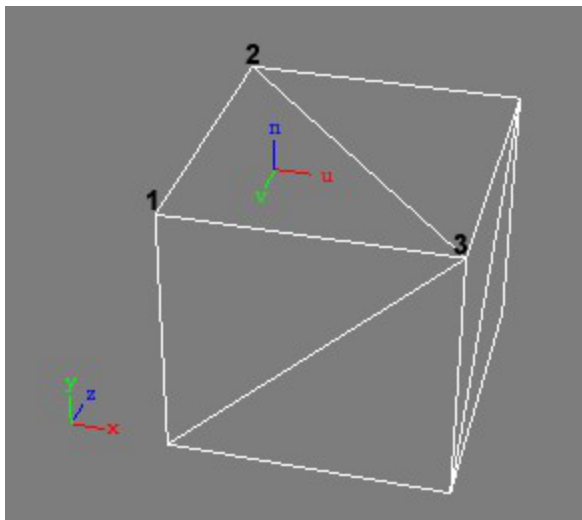
In 3d world there exist many different coordinate systems: world space, object space, camera space etc. Tangent space is just another coordinate system with its own origin, it's used to specify texture coordinates for a polygon face. You are probably familiar with UV coordinates, imagine X axis pointing in V direction and Y axis in U direction. We have a coordinates of 2d space now. But a coordination system needs 3rd axis to operate in a 3d world, and that's where normal of a face comes in. In



tangent space coordinates, a face normal (N) direction is used for Z axis on the coordination system.



On this image you can see world coordinates in the bottom left and tangent space coordinates attached to quad made of two triangles.



This image helps visualize the coordinates on the more common shape - a cube.

The u, v, n axis represent the direction in which u, v, n values increase across the face, just as the x, y, z values represent the direction in which the x, y, z values increase in the world space coordinate system.

What it all means is that we can map RGB channels of the normal map to match tangent space coordinates. Red channel is responsible for U axis, Blue is for N axis and Green is for V axis.

To simplify, Red is Left and Right; Green is Up and Down, Blue is Outwards away from the surface.

If you see lighting coming from the wrong angle when you're looking at your normal-mapped model, and the model is using a tangent-space normal map, the normal map shader might be expecting the red or green channel (or both) to point in the opposite direction. To fix this either change the shader, or simply invert the appropriate color channels in an image editor, so that the black pixels become white and the white pixels become black.

### ***Man, this is hard. Why do I care?***

Tangent Space normal maps use a special kind of vertex data calculation called the *tangent basis*. Light rays are in world space, but the normals stored in the normal map are in tangent space. When a normal-mapped model is being rendered, the light rays must be converted from world space into tangent space, using the tangent basis to get there. At that point the incoming light rays are compared against the directions of the normals in the normal map, and this determines how much each pixel of the mesh is going to be lit. Alternatively, instead of converting the light rays some shaders will convert the normals in the normal map from tangent space into world space. Then those world-space normals are compared against the light rays, and the model is lit appropriately. The method depends on who wrote the shader, but the end result is the same.

Problem for artists is that there are many different ways to calculate *tangent basis*. **Meaning that a normal map baked in one application probably isn't going to be shaded correctly in another.** When the renderer(game engine for example) renders your model, the shader must use the same tangent basis as your normal map baking application, otherwise you will get incorrect lighting on your model, especially across the UV seams.

***I want my normals to look great and I don't want to do extra work!***

There are plenty easy ways to ensure that normal maps you bake will work as they should the game engine of your choice. Let's see some examples:

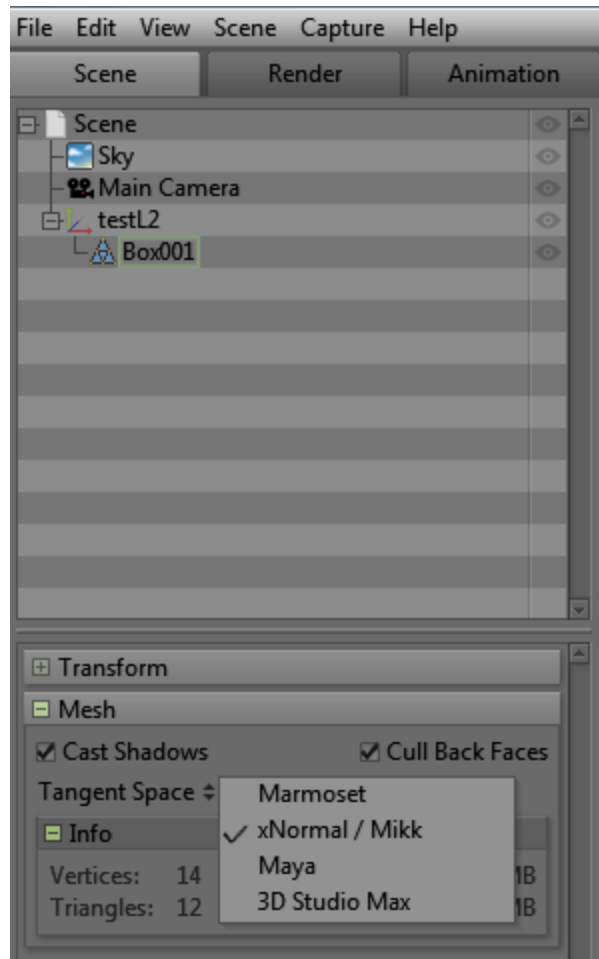
**Xnormal:**

This application bakes normal maps in tangent space called Mikk-Tspace.

Applications that support Mikk-Tspace:

Unreal Engine 4  
Marmoset 2

Marmoset 2 also supports maps baked in 3ds max and Maya

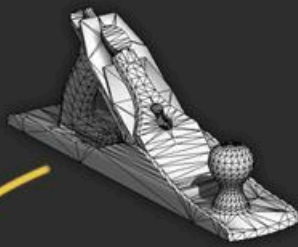


For Unity engine you will have to install a custom Unity tangent space plugin for Xnormal if you want to bake maps in that application.

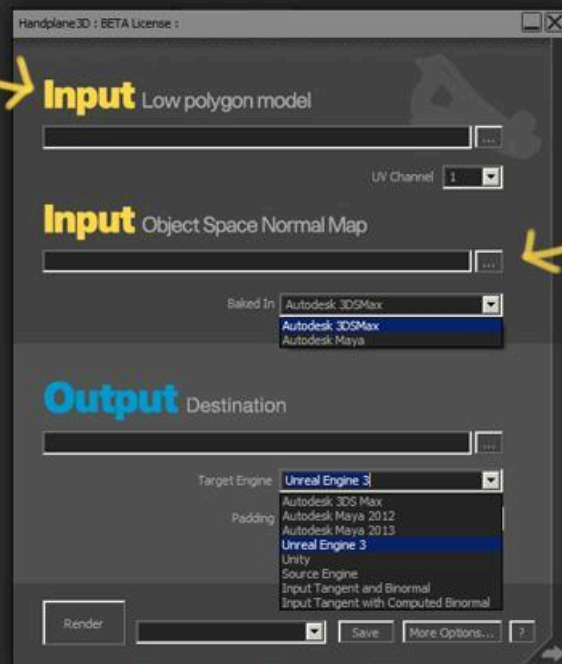
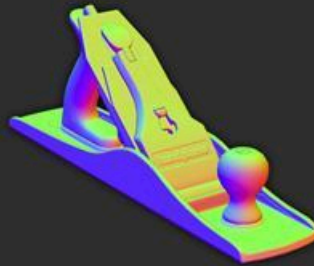
That's just a few examples for using Xnormal. But what if you need to use 3ds Max for baking and Unity engine for displaying maps?

For situations, where don't have a synchronized workflow, there exists Handplane <http://www.handplane3d.com/> . It's a freeware (completely free) software. You will have to bake an Object Space normal map that will be converted to a normal map with the tangent space you want.

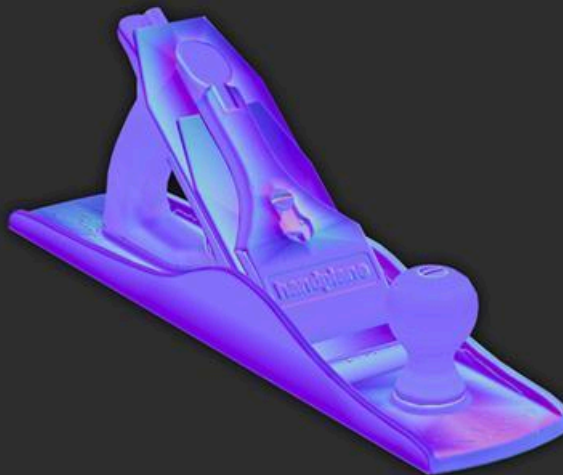
Lowpoly Model



Object Space Normals



Tangent Space Normal Map



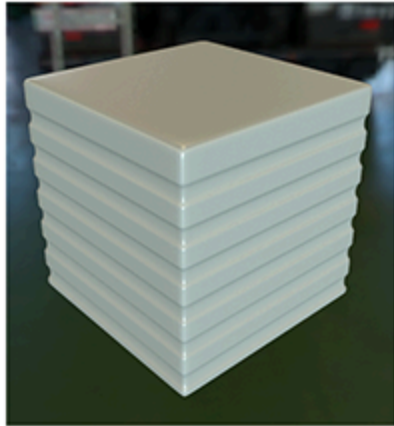
# Disclaimer

Throughout this guide I will use a **Synced Workflow**: baking normal maps in **Xnormal** and displaying them in **Marmoset 2** with the appropriate tangent space selected in the settings for a displayed object.

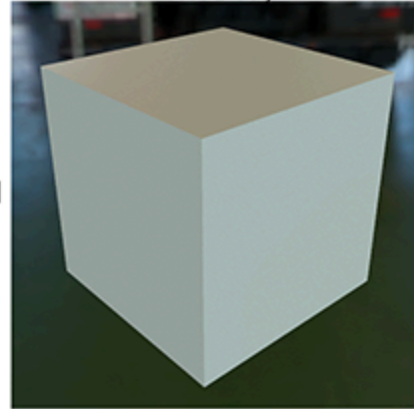
## Creating a Normal Map Simple overview

The process of creating a normal map is usually called "baking". To simplify the pipeline of creating a normal map, it looks like this:

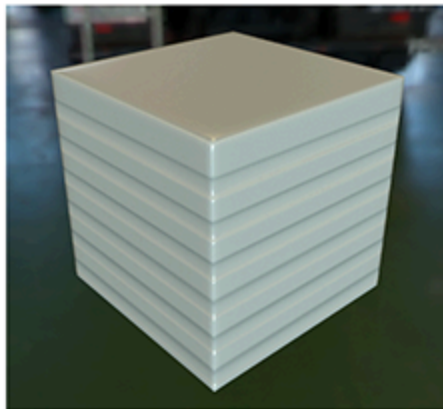
High polygon object



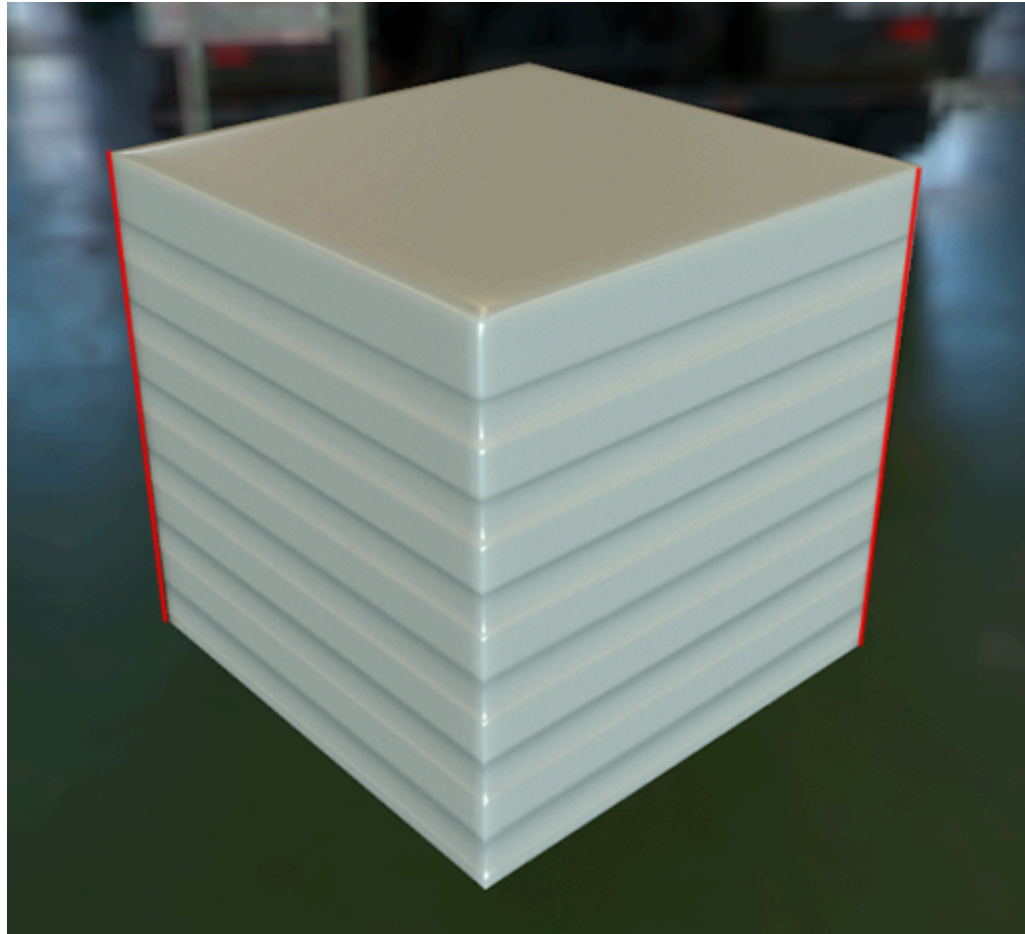
Low polygon object(game model)



Normal Mapped Low Polygon object



As you can see, the normal map is only capable of faking the definition of high polygon object. It is not capable of altering silhouette of your low polygon object.



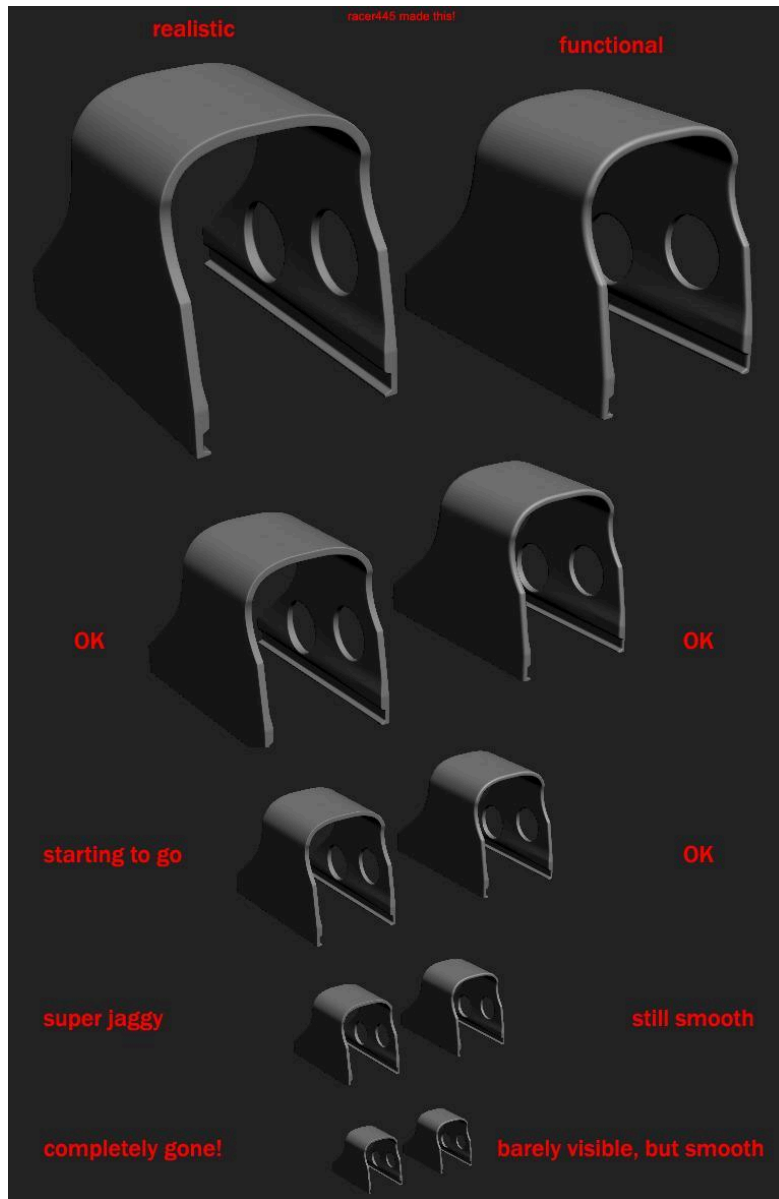
## Building High Poly

To create perfect normal maps you have to know how to construct your high poly models to take full advantage of normal mapping.

### ***1. Edge Thickness***

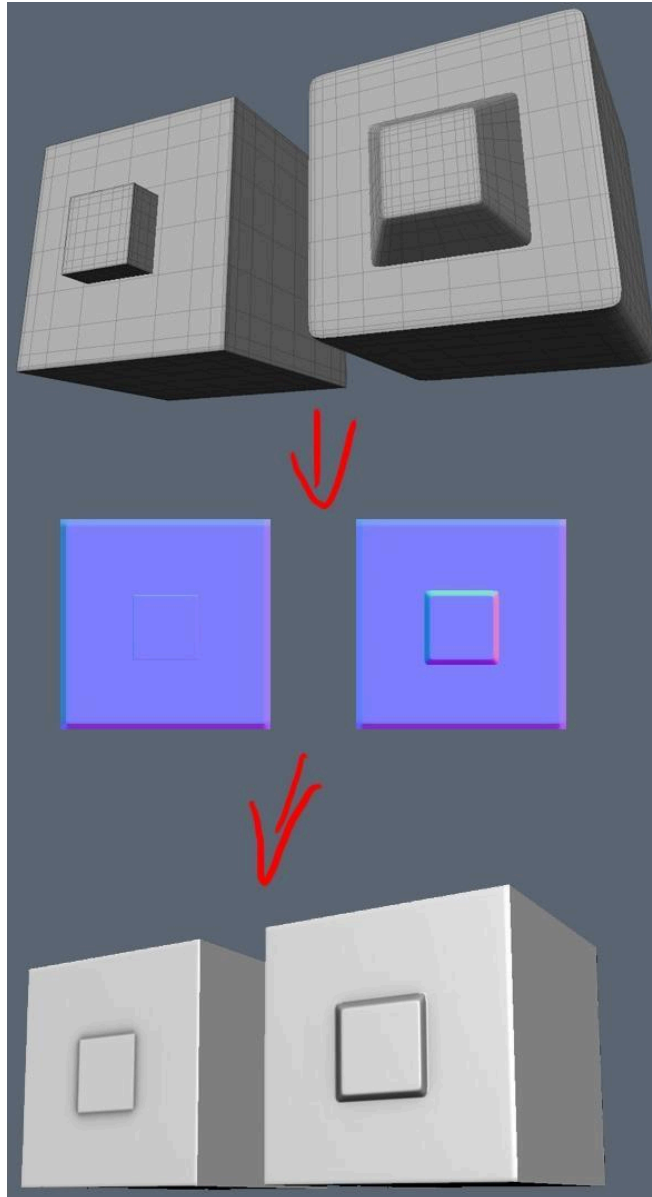
If you want to represent edges from your high poly model in the normal map, you should make them smoother. If your high poly edges are too tight, then they won't be very visible on the normal mapped mesh, and you want your edges to stay readable on every distance.





## 2. *Sloped extrusions*

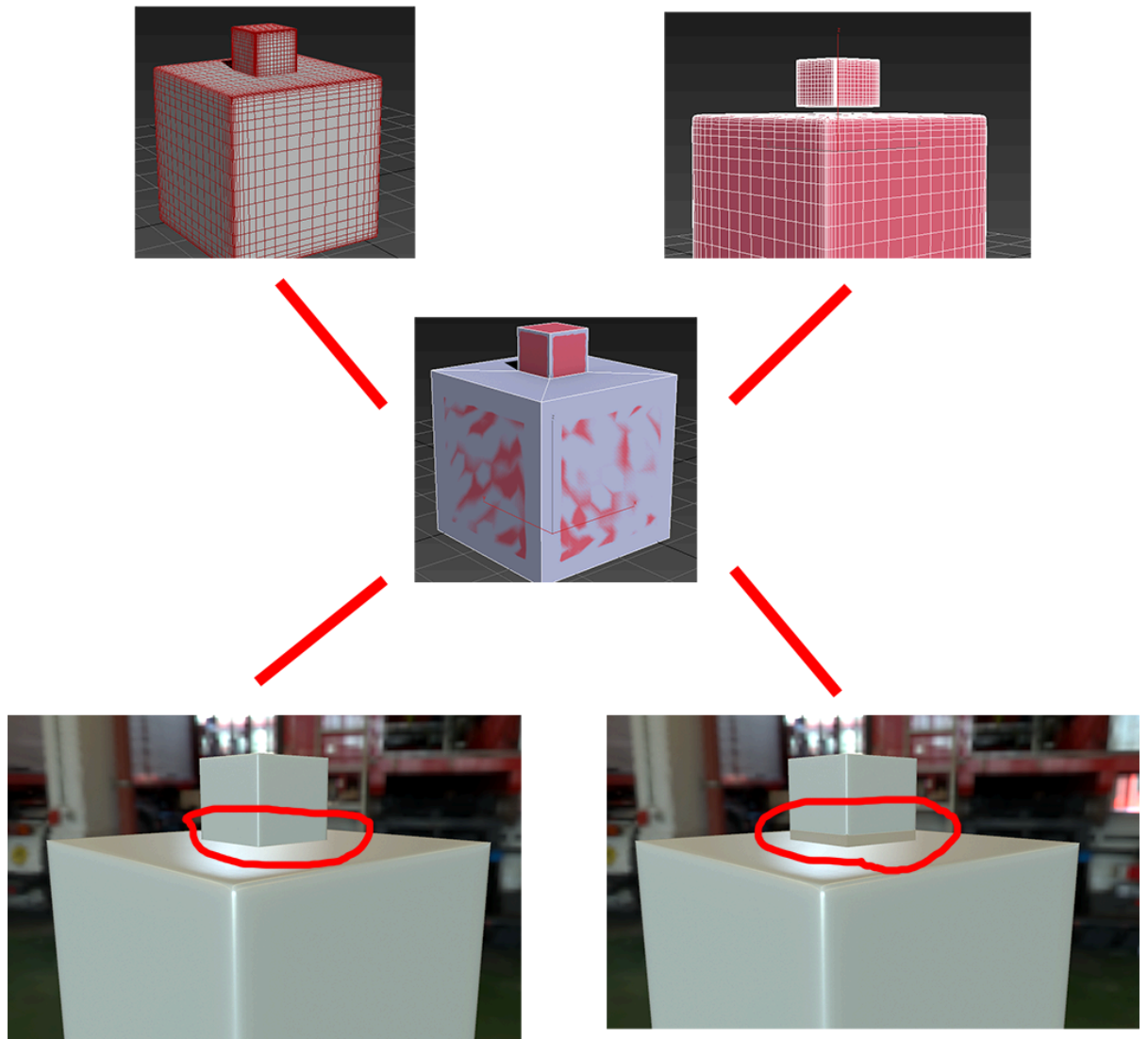
Because of the way the projection of high poly details on low poly works, the baking can't catch what it can't see. So if you want to represent an extrusion on your normal map, it should be slanted/beveled so it could be seen on the flat plane from the front.



### ***3. Intersecting geometry for combining in low poly***

It helps to constantly think about the final product when you build a high poly mesh. It's generally a good idea to build a continuous low poly mesh when two objects never animate or seen separately, like a belt on pants. It would be a good idea to intersect or match closely pants and belt together in high poly to build a continuous mesh around it later. If two meshes don't align properly it can give you

errors in normal map in places where there are holes between two meshes. Normal map can't display what isn't there.



You should keep that in mind when you have to merge two objects together in low-poly. In some cases it would be a better idea to use **interpenetrating meshes**. In this example I would have the little low poly cube intersecting with the main cube. These meshes will have to be baked separately.

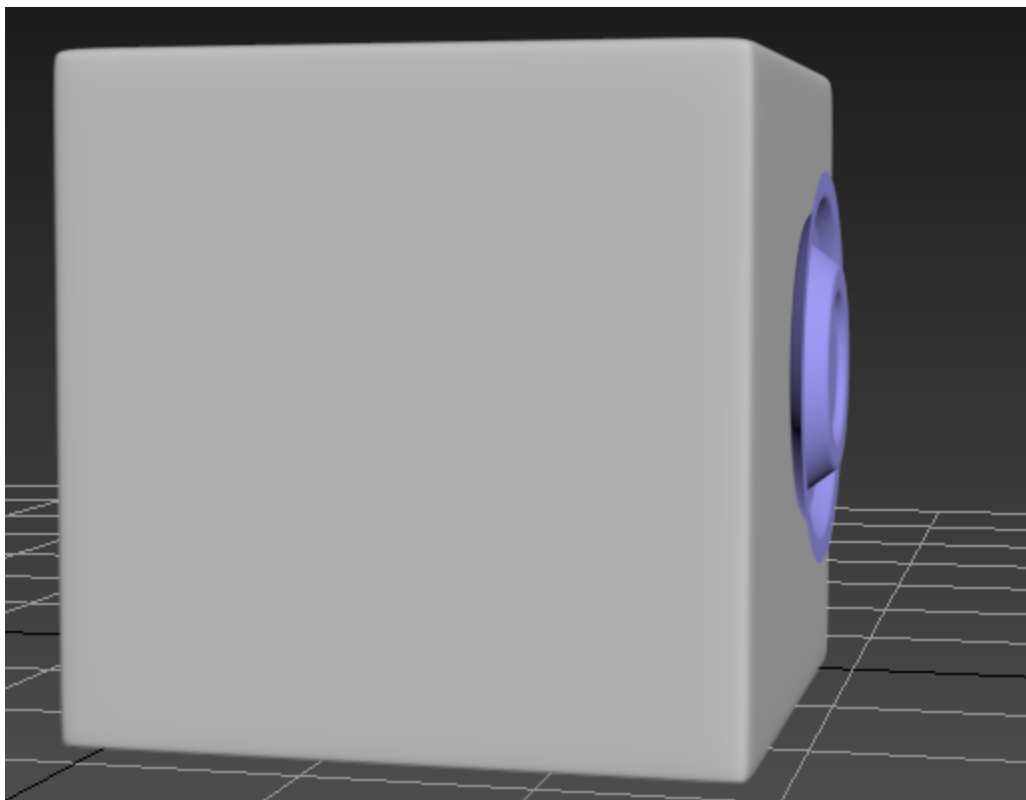
## ***4. Decimating***

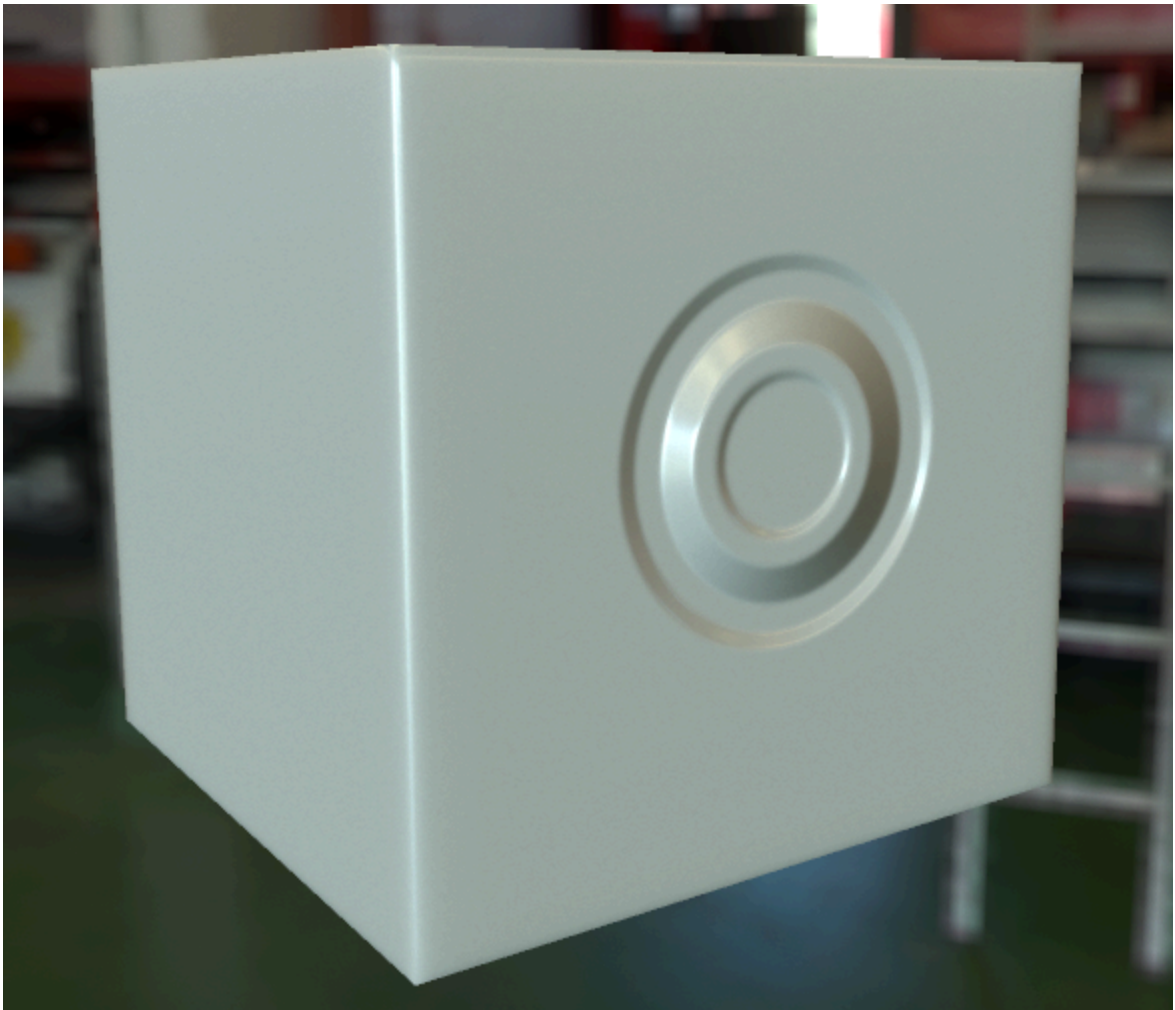
If you bring your high poly from Zbrush to other apps, it would be a good idea to decimate it first. There is no clear rule about how much you can decimate before you break the model, so it's recommended to do some test bakes before committing to anything. Most of the time decimating to 20% provides a pretty good result for baking. If a model looks good when decimated, then it's probably will work for baking.

## ***5. Floaters***

Floaters are high poly objects that act separately of the main object and work as easy-to-add details for baking a normal map. It's a good way to fake geometry without having to break underlying high poly model and it usually saves a lot of time. Downside is that you have to adjust your low poly model to accommodate floaters and it's very hard to place them on rounded or complicated surfaces, so they are mostly used on flat planes. If you place your camera directly in front of the floater object and you can't see visible seams, then it should work right when baked.

It's important for floaters that present indentations to have flat corner edges. It's a good idea to have floaters placed as close to the surface as possible.

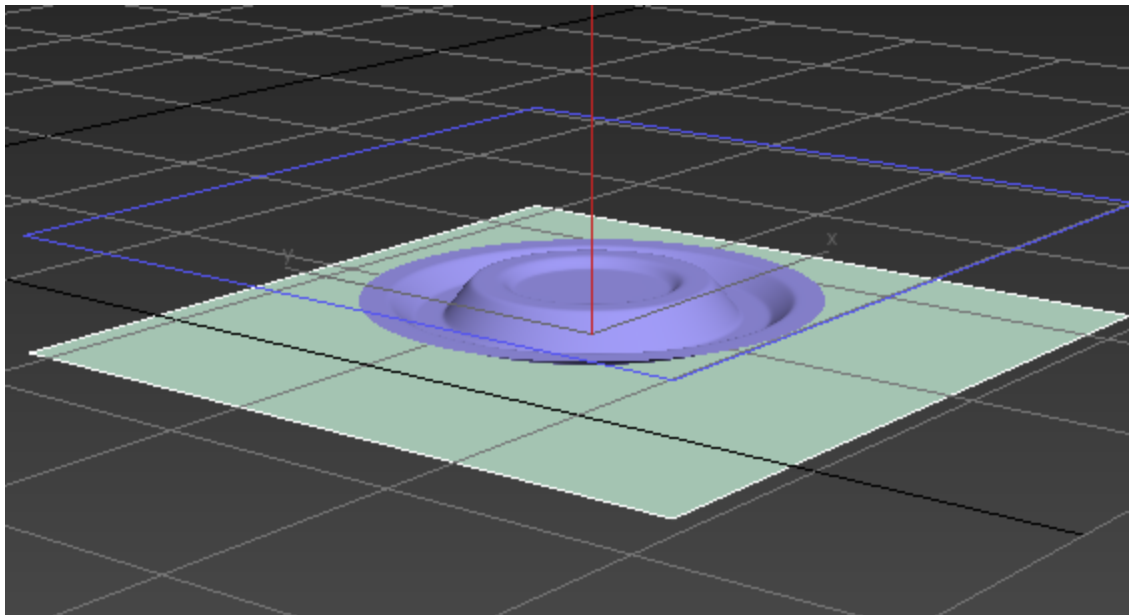




## ***6. When to model and when to paint***

Sometimes modeling and placing high poly geometry, whether it's a floater or not, is a bad idea and a waste of time. For example, floaters can be baked separately on a flat plane and combined into the normal map in Photoshop. That approach makes duplicating of objects very easy and helps to avoid problems, when there's not enough resolution when you do bakes to clearly represent small objects.

This is just one of the methods to do that. If you are baking your normal in PNG format it will save the map with transparency.



Then you bring it to Photoshop, and combine with your baked normal map and overlay. There are many ways to do that:

## ***Methods to combine normal maps***

Preferred and the most accurate method of combining normals would be:

<http://www.polycount.com/forum/showthread.php?t=131819>  
This Photoshop plugin or this RNM Normal Map Combiner  
<http://farfarer.com/resources.htm> . If you prefer to use  
Substance Designer, there's also this node that does the  
same correct combining operation  
<http://forum.allegorithmic.com/index.php?topic=262.0>

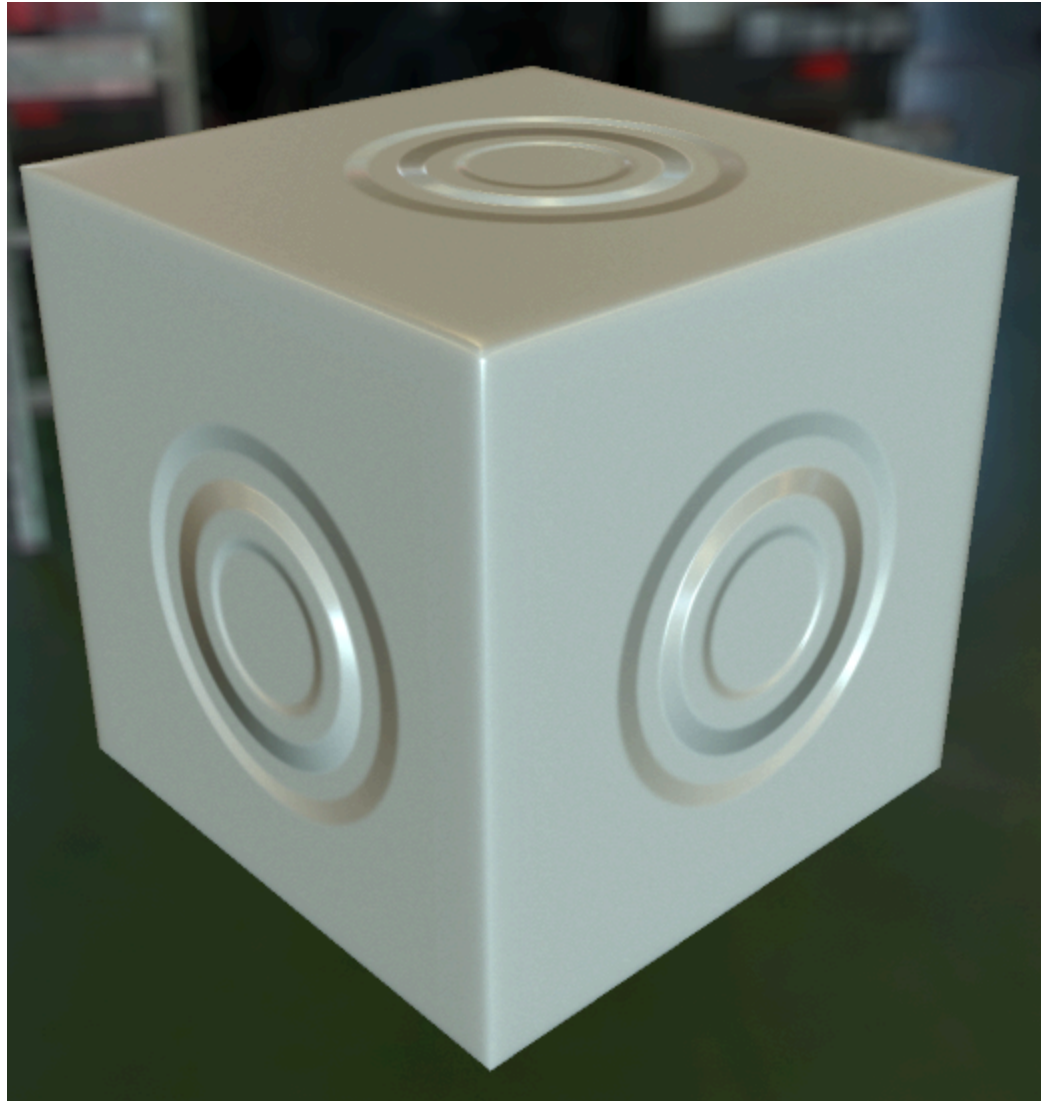
A few other methods, but they are not as accurate and reliable:

**Quixel NDO2** software can do the process of combining effortless, but it doesn't work in every case.

You can do it manually in Photoshop using this method:  
<http://vimeo.com/95516153>.

**Crazybump** also provides a feature to overlay normals.





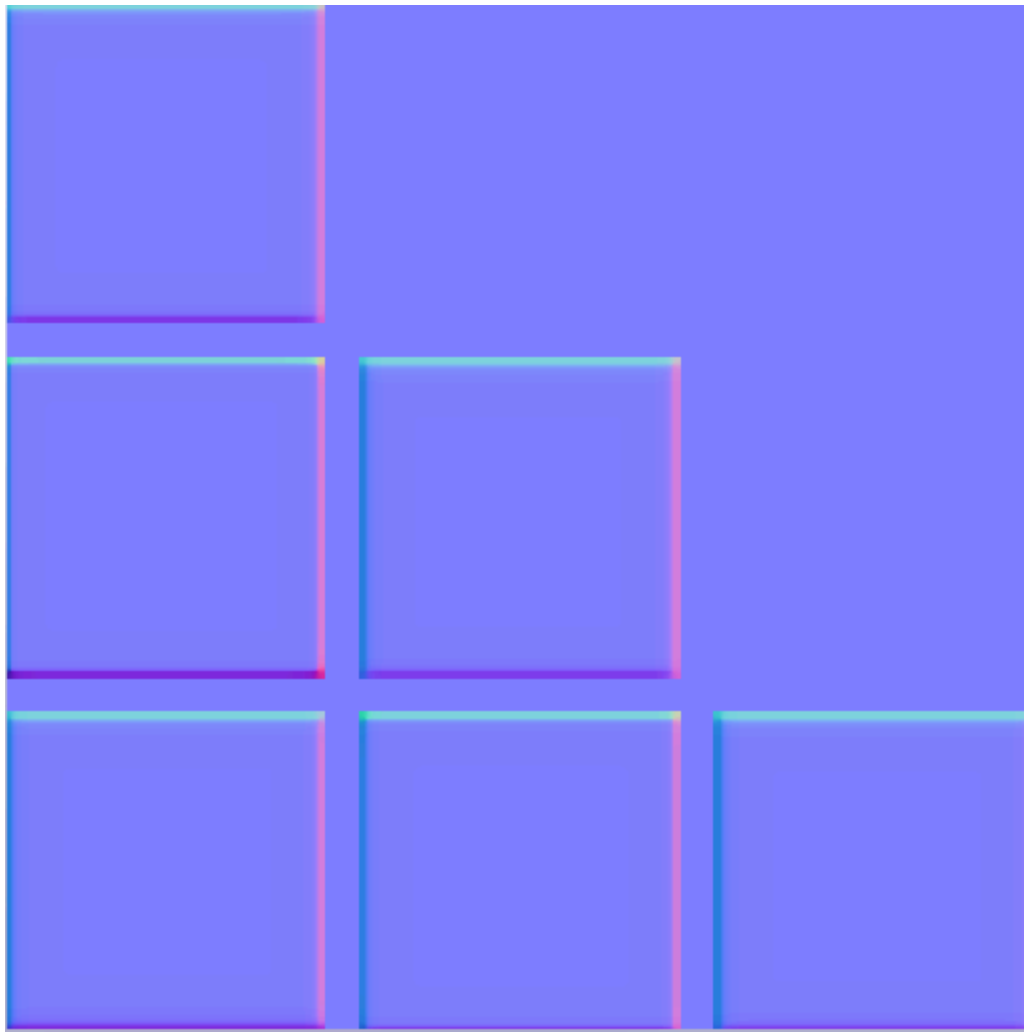
# Building Low Poly

## *1. Gradients*

Before baking anything you have to understand the concept behind normal map gradients. Where a normal map is

a light blue color (R/G/B: 128/128/255) it means that the normal points in exactly the same direction as the interpolated vertex normal at that pixel. When normal map has to adjust to extreme changes in geometry, gradients start to appear. The less normal map has to adjust the normal, the better. But that doesn't mean that gradients are evil.

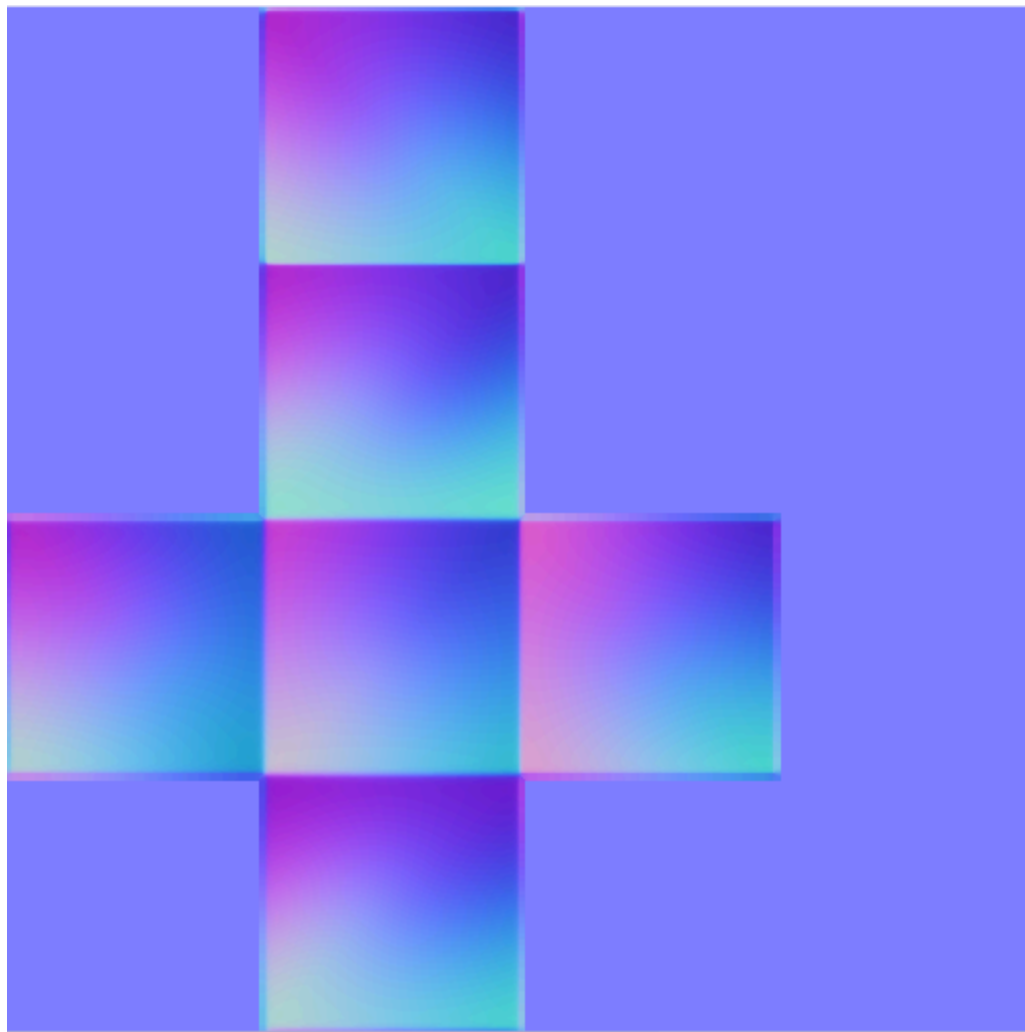
Let's do a test. Just our regular test bake with a cube. Every face of the cube was separated in UV layout and each had a unique Smoothing Group applied to it:



Looks all pretty and blue, right? No gradients, essentially a perfect bake! But what does it mean and how to achieve that? Or a more interesting question: should I bother?

What it means is that your normals point in the right directions, no gradients, and the map doesn't have to work extra time to adjust for hard angles in your low poly mesh. Practically, it's a good idea to keep gradients in your normal map as clean as possible.

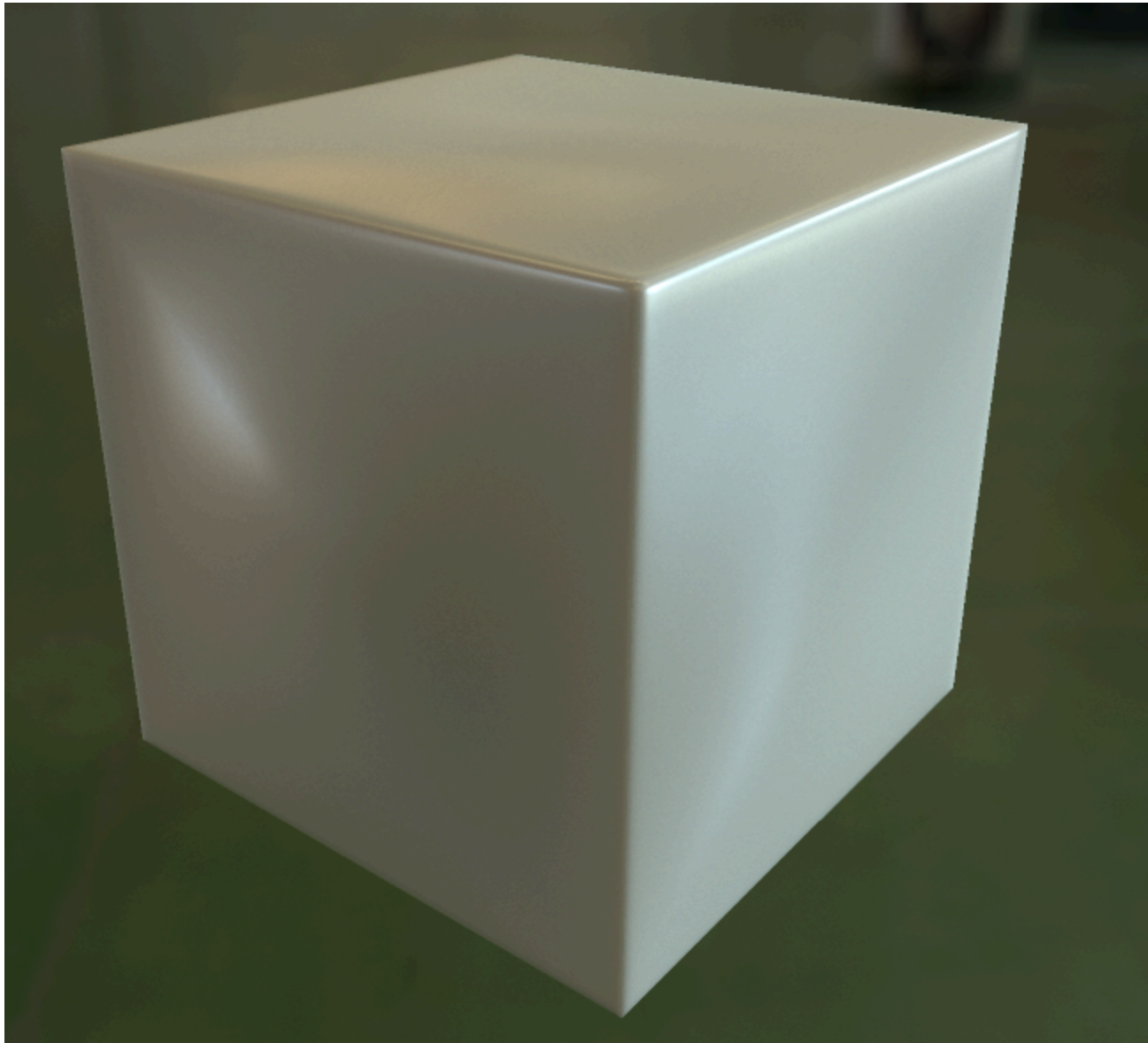
Alright, let's introduce some extreme gradients. I stitched all UV islands together and applied 1 Smoothing Group to the entire object.



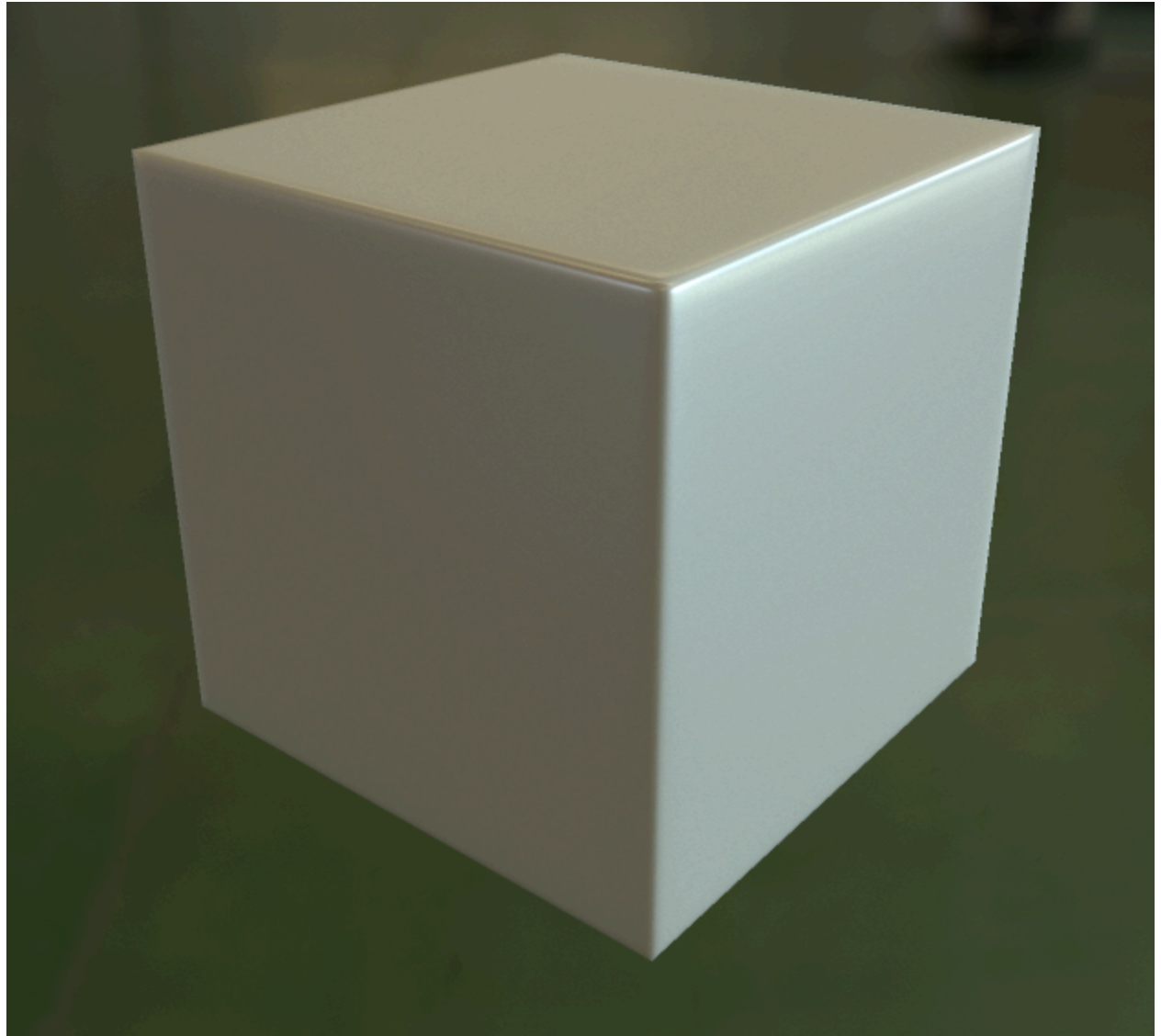
It's the same cube, but you can see that it's way different and scary looking now. I'll tell you what it's all about a bit later. Now let's see the difference between two bakes:

This is the cube with extreme gradients in the normal

map. Clearly, the normal map has a hard time dealing with 90 degree corners of the cube.



And this shows how the map without extreme gradients looks much better and clean.



But that doesn't mean that you should avoid gradients at all costs! Now way, that's just not a good idea. The most extreme gradients can provide an issue for you, but most of the time you can get away with gradients, especially if your normal map baking engine is synced with your game engine. Most of the time you should do test bakes to see what you can get away with in your particular engine of choice, where you will present your game assets.

There are various benefits of having clean bakes, such as:

You will get better results when doing LOD meshes that

share the same texture, because the normal map won't have to rely so heavily on the exact mesh normals. You may need to have a separate normal map baked for LOD meshes otherwise, which uses up more VRAM.

Better texture compression

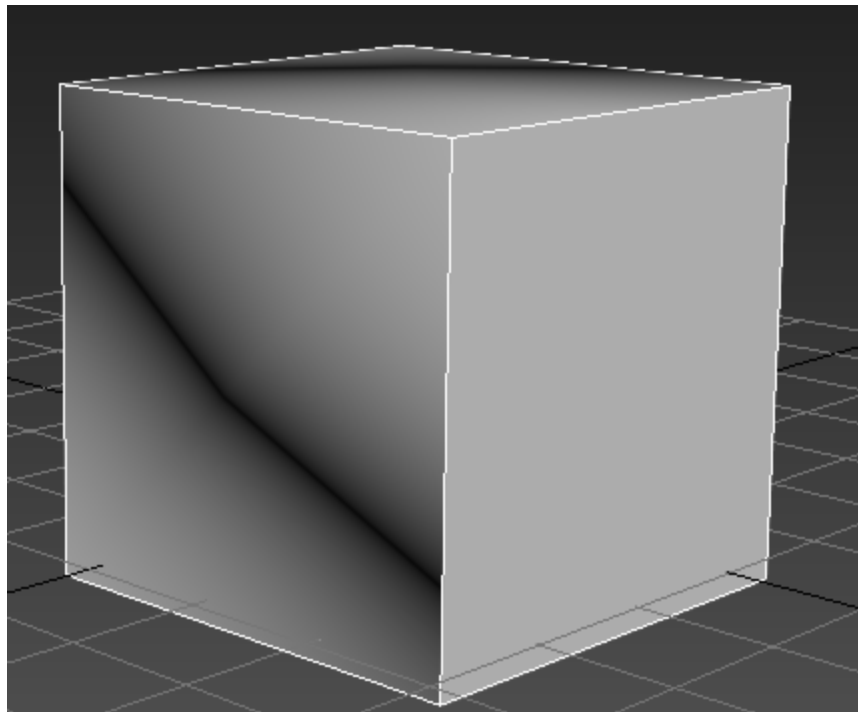
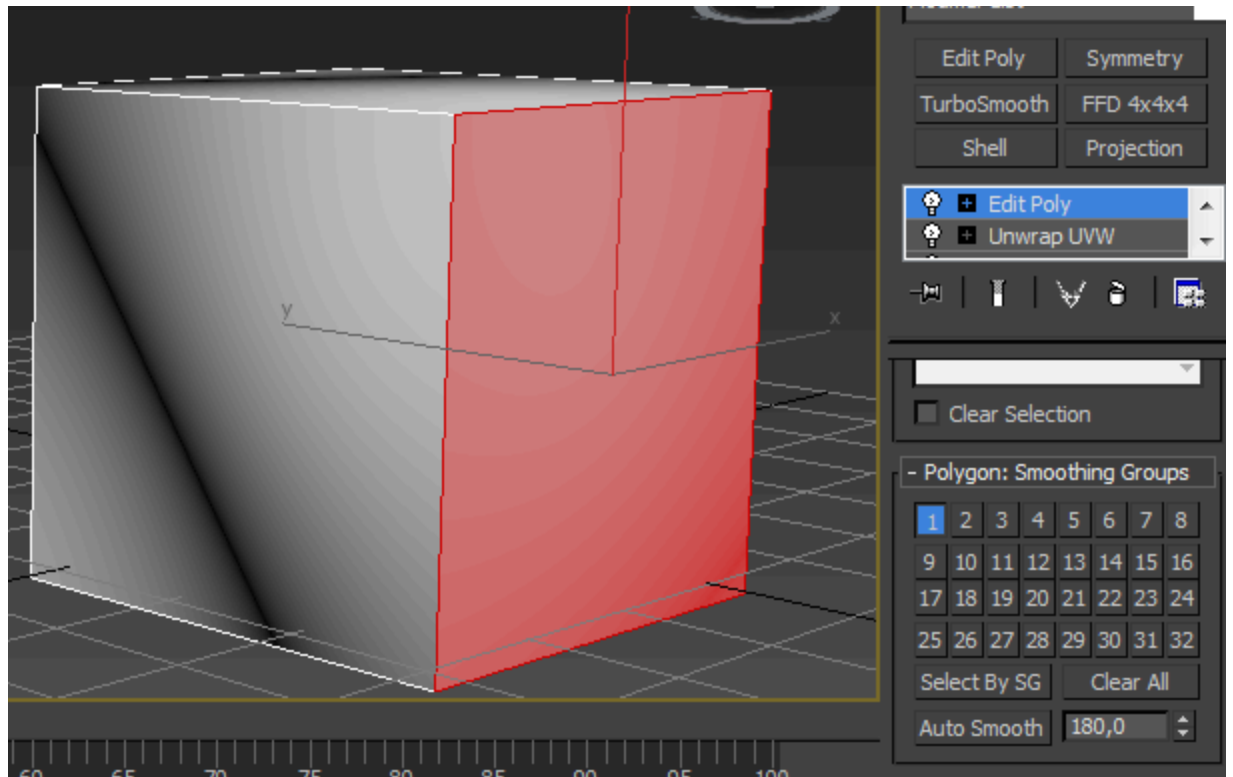
Will reduce "resolution based smoothing errors" that happen when you have a small triangle but not enough resolution to properly represent the shading. These usually show up as "little white triangles" in-game. In the same regard it improves how well your normal map will display with smaller mip maps.

## *2. Hard edges and UVs*

So let's say you don't want gradients and need to have clean and pretty bakes. Then you will have to introduce hard edges to your low poly mesh. Each time you harden edges on your lowpoly, the normal map will ease on the gradients on that edge.

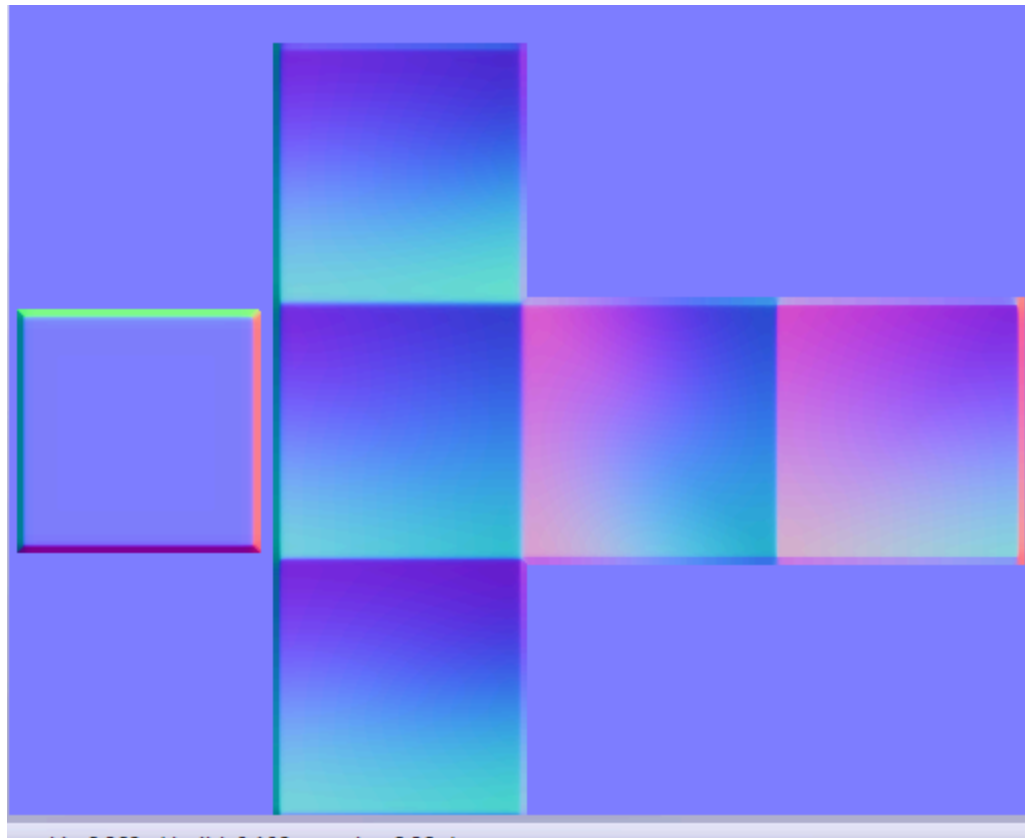
How to do hard edges:

In 3ds Max, you'll have to use Smoothing Groups to separate faces of your low poly mesh that you want to have hardened. In polygon sub-object mode, select polygons and apply a smoothing group to them from the menu at the very bottom of the



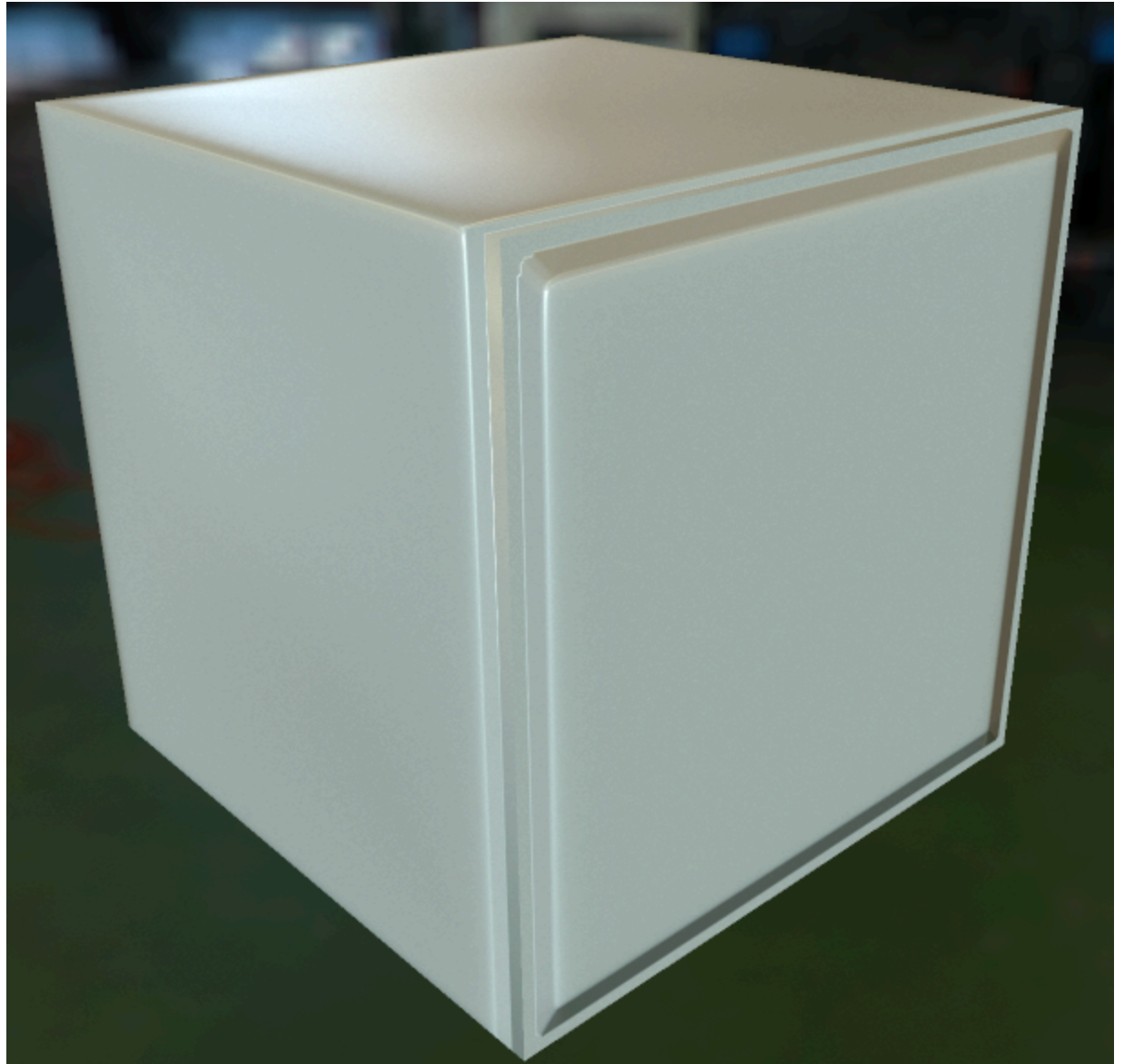
Right now this mesh has 2 polygroups: 1 for the selected face and 1 for every other face. You can see that the faces that share a polygroup appear to have distorted shading. If you can see that it will mean that your normal

map will show extreme gradients after baking. Let's see how that works.

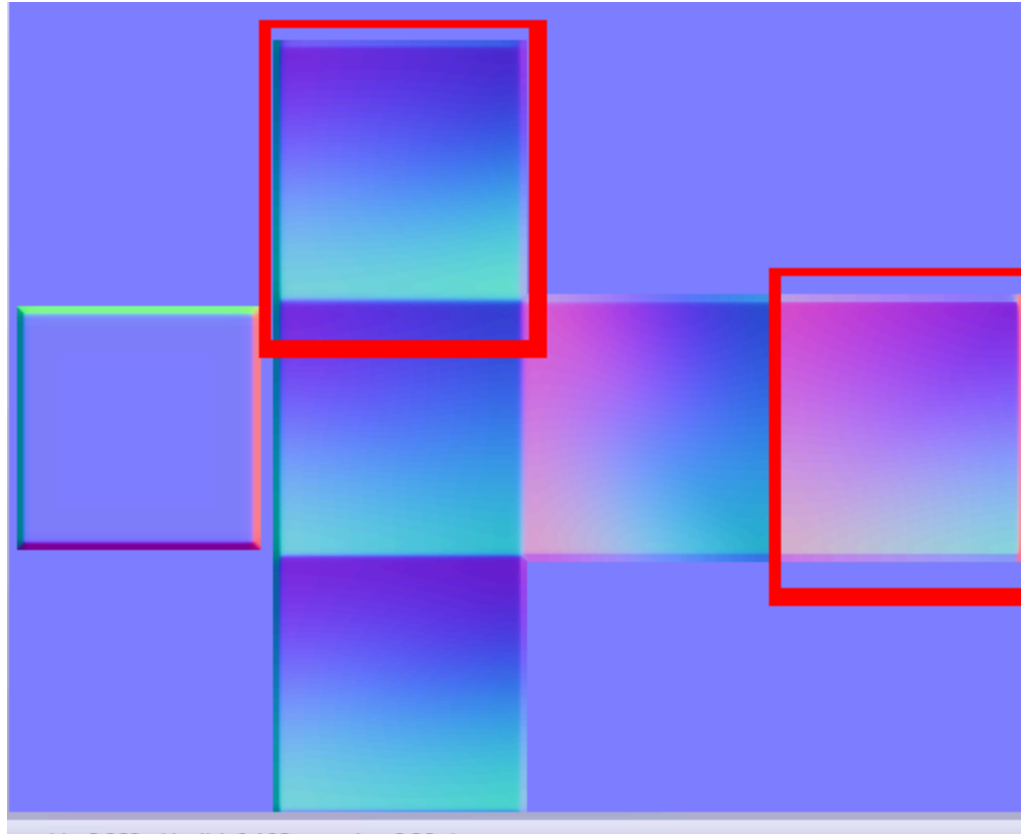


You can see on this screenshot that one face became separated from the rest. Keep in mind that I never did anything with UVs, I only applied a different polygroup to that face. Now the face is clean and pretty, but this is how it appears in-engine:





Scary looking, huh? But, forget about the weird looking side for now. If you look at the rest of the mesh you can see that even separating one face from the rest eased up the gradient in the normal map a bit not only for the separated face, but it affected the faces that share the edge with the hardened face too.

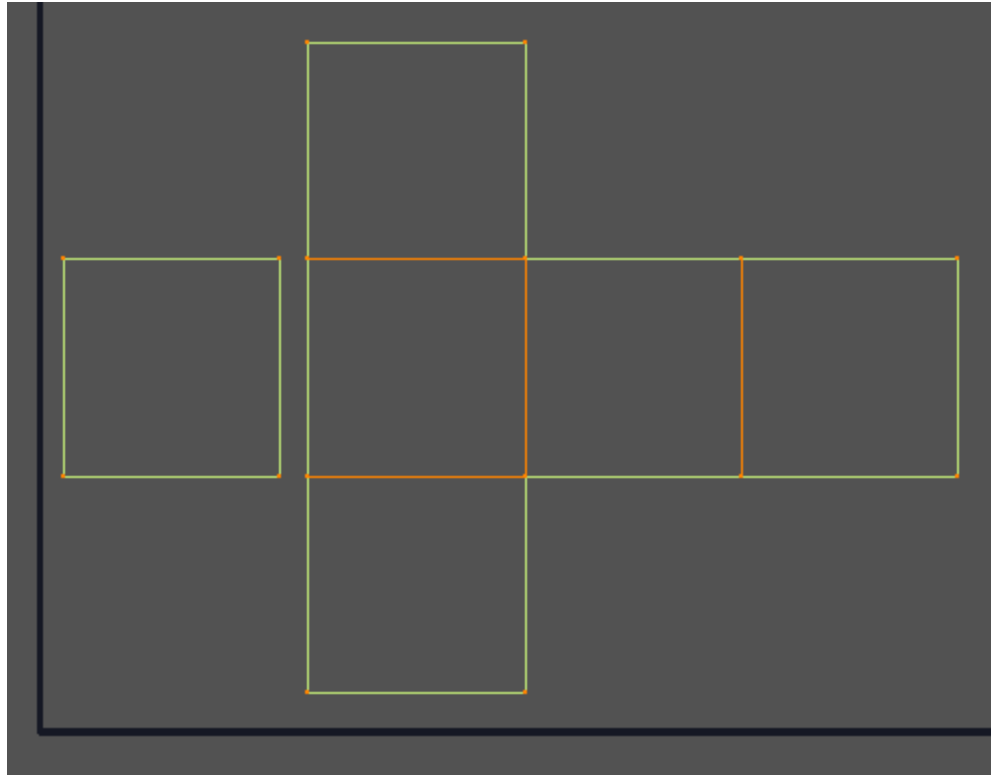


See? The faces changed the gradients. The bluer it is, the better it will look in-engine.

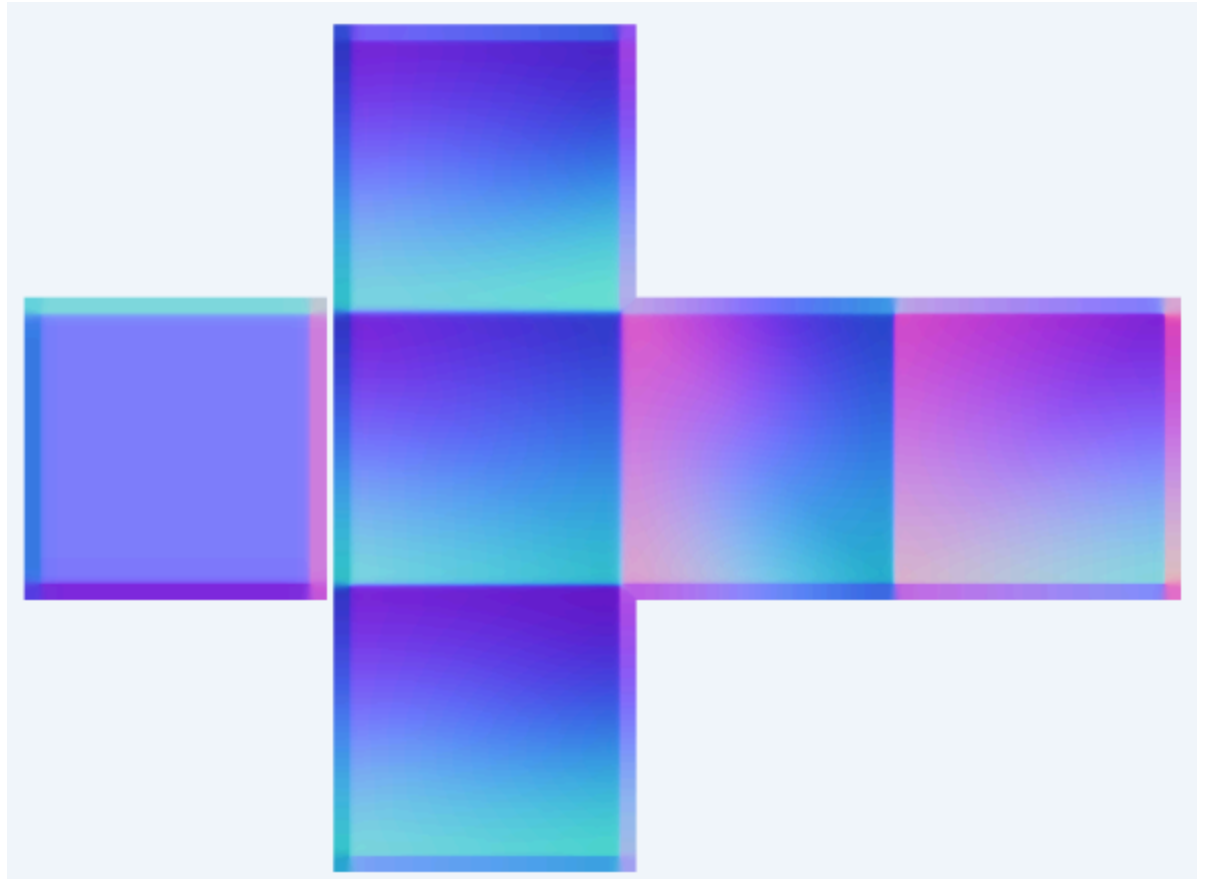
So what about that weird looking... thing? That is an issue that appears when you separate faces in smoothing group, but don't separate them in UVs. The face on the layout can appear separated, but it's not and that gives you that peculiar result.

**If you want to use hard edges, you have to separate UVs where you have hard edges.**

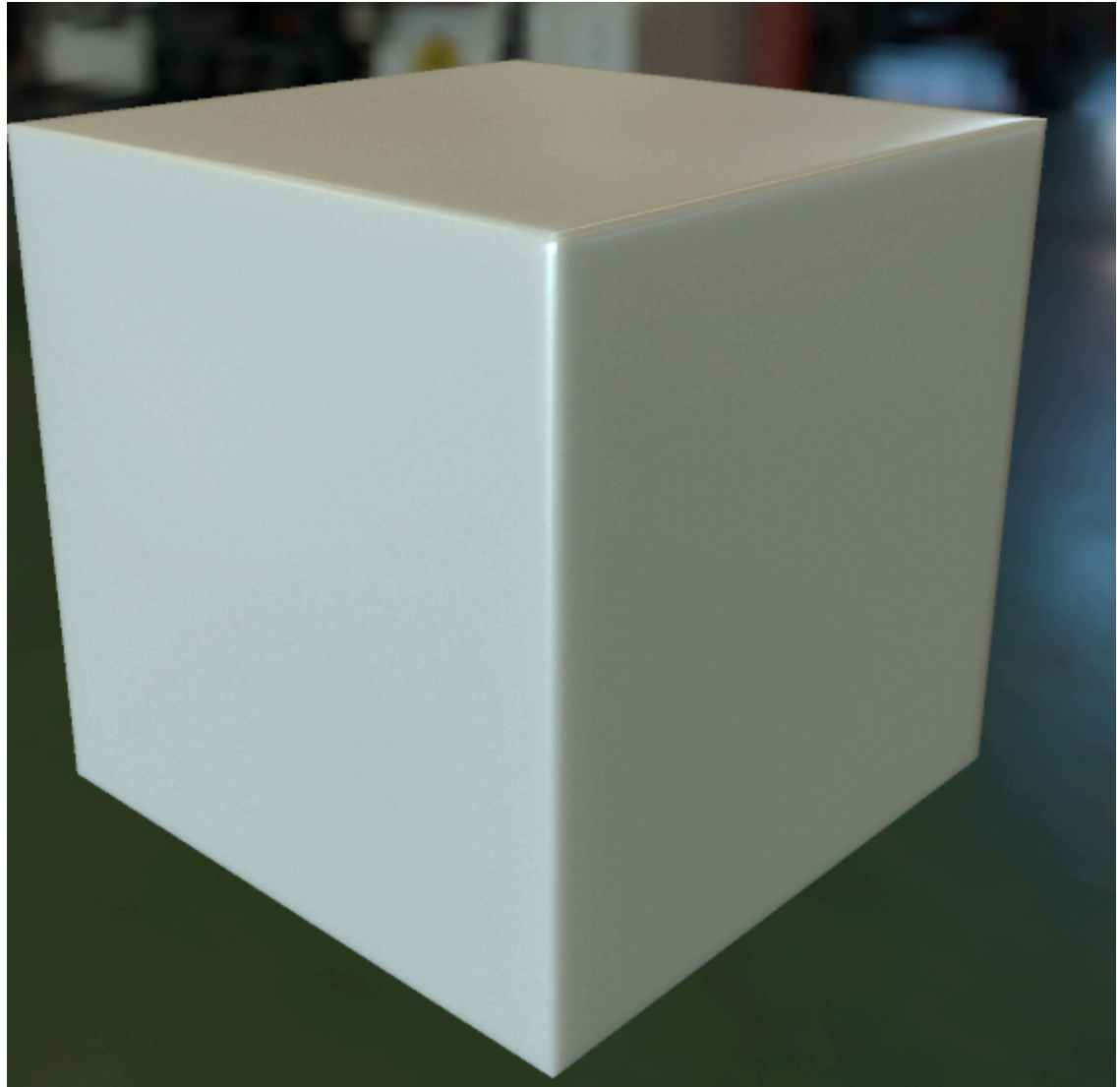
I separated the hardened face in UV



Did a rebake and this is the result:



Here, with transparency, you can see how the hardened face really became separated from the rest and now nothing is negatively affecting it and there is no issue in-engine.



Now if I apply a unique smoothing group to every other face and detach their UVs I'll get a perfect looking normal map. But what if I have not 6 faces, but a lot more? There's an easy answer for that. Download <http://www.renderhjs.net/textools/> Textools plugin for 3ds Max, it can do an operation called "Smoothing groups from UV shells".

Basic workflow for me looks like this:

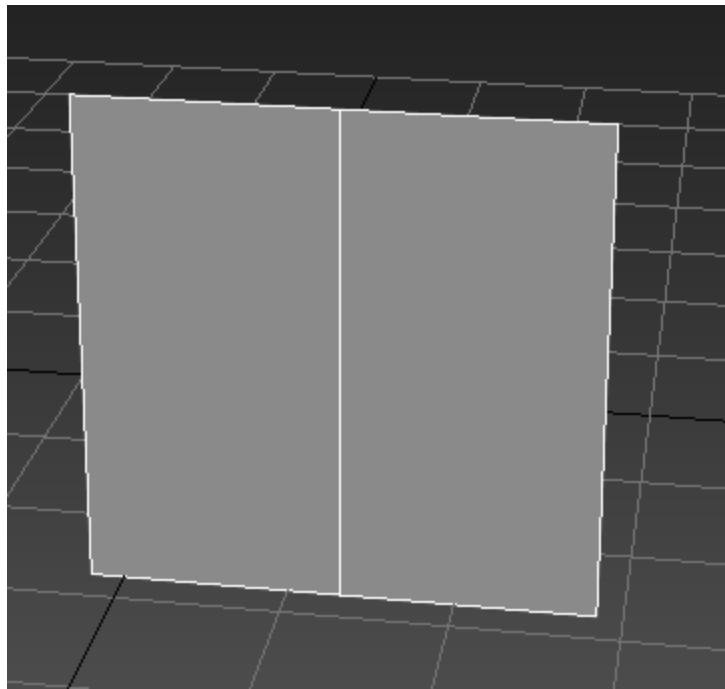
1. Separate UVs where I want hard edges to be
2. Apply "Smoothing groups from UV shells" to apply separate smoothing group for each separate UV island.

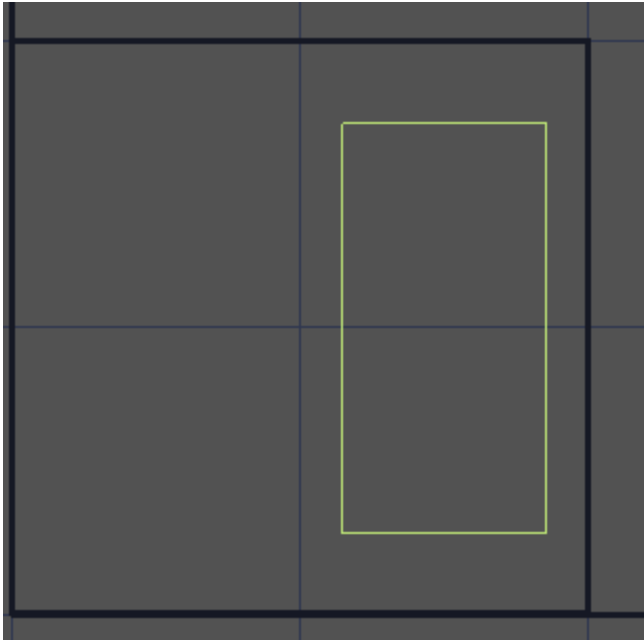
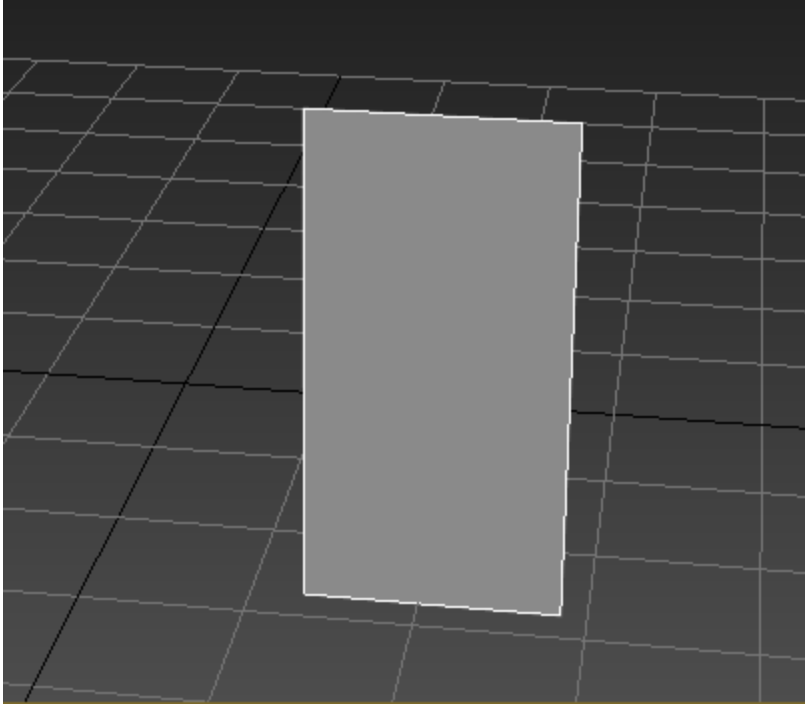
However, it's very easy to become a victim of this easy workflow. Sometimes you may not need separate smoothing groups on UV shells. For example, if you have mirrored UV islands or a rounded object that was unwrapped to several UV shells, in situations like that you don't want to have breaks in smoothing groups.

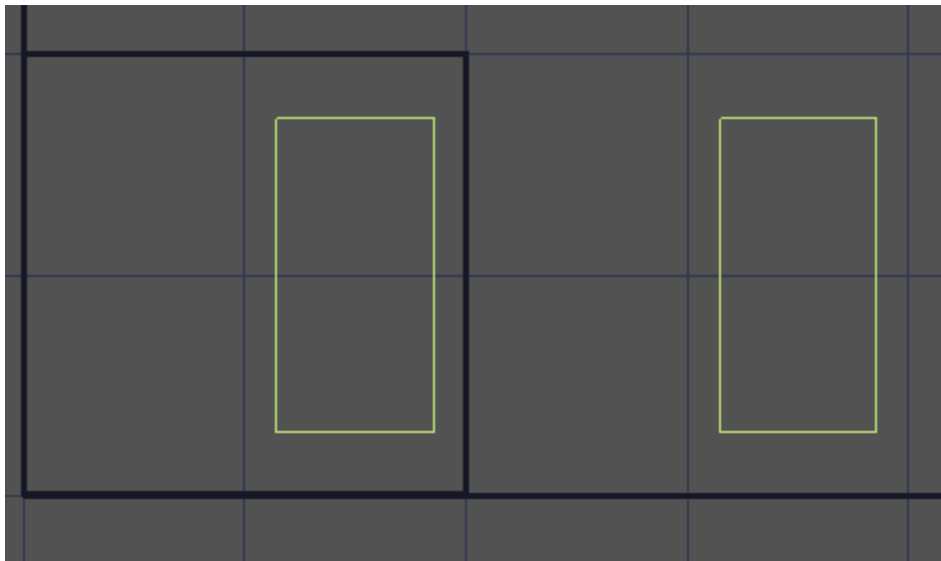
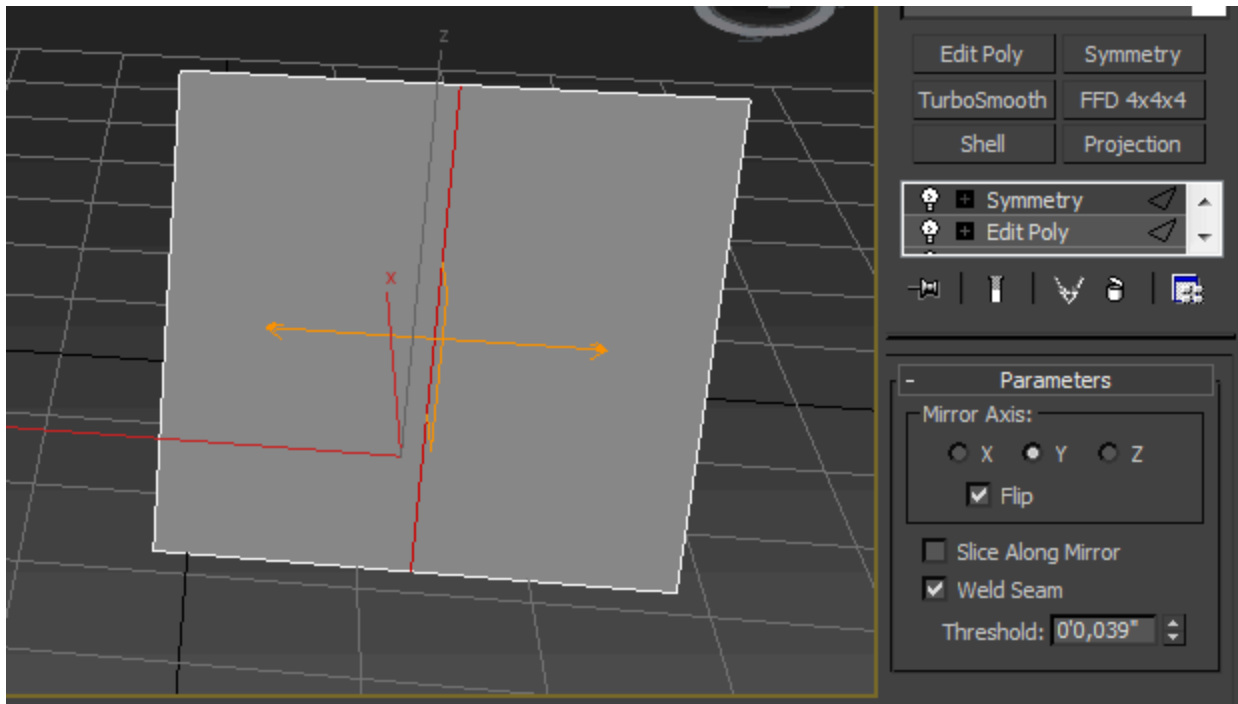
### ***3. Mirroring UVs***

If you need to mirror half of your low poly mesh, a good way to do it would be like this:

1. Delete half of your symmetrical mesh
2. Unwrap UVs
3. Apply Symmetry to the mesh
4. Offset UVs of the mirrored part
5. Bake normal map





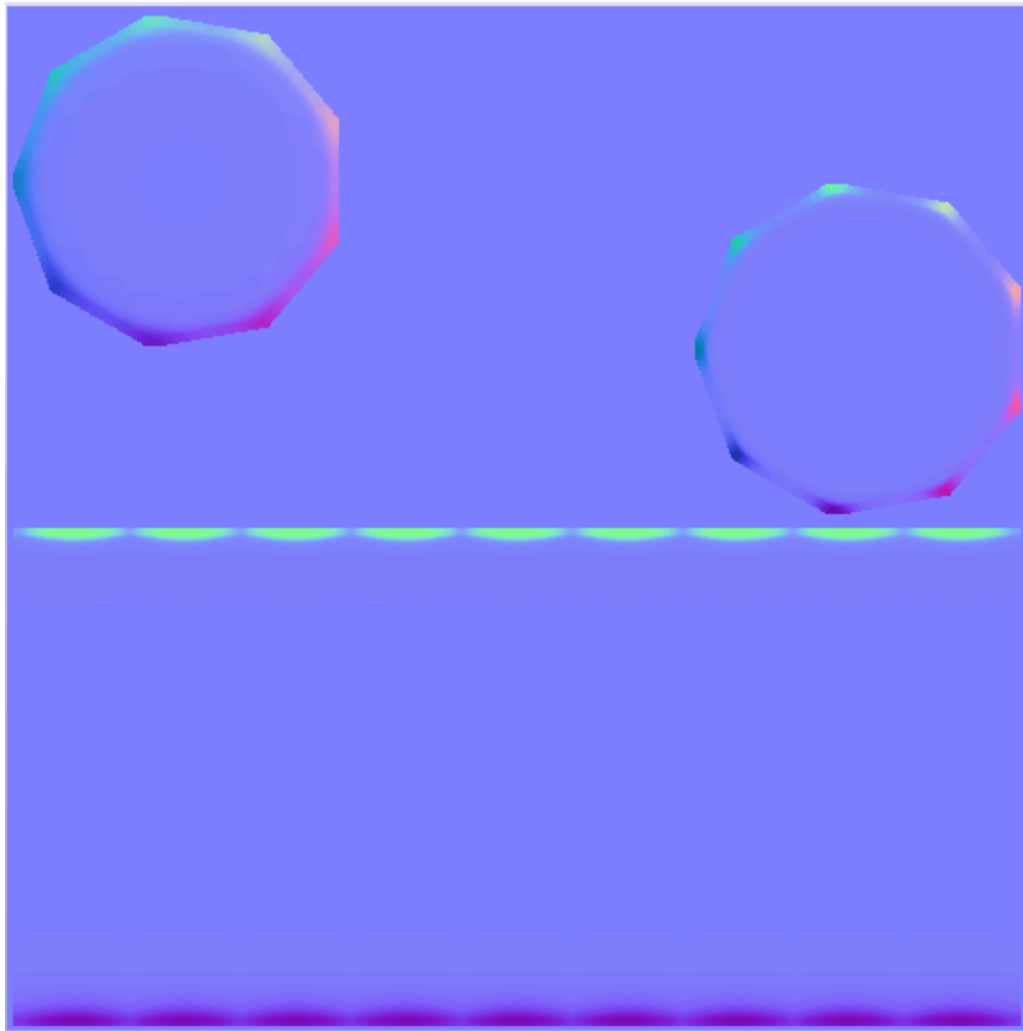


Same offsetting is done when any two or more shells share UV space. You have to offset them before baking, otherwise there will be issues.

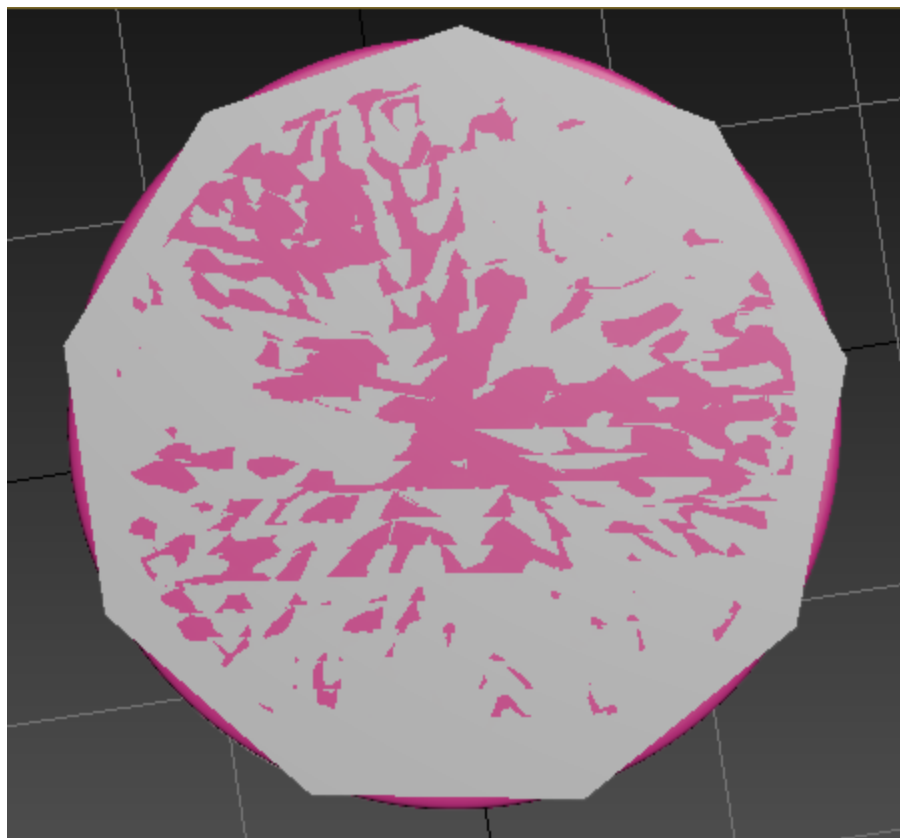
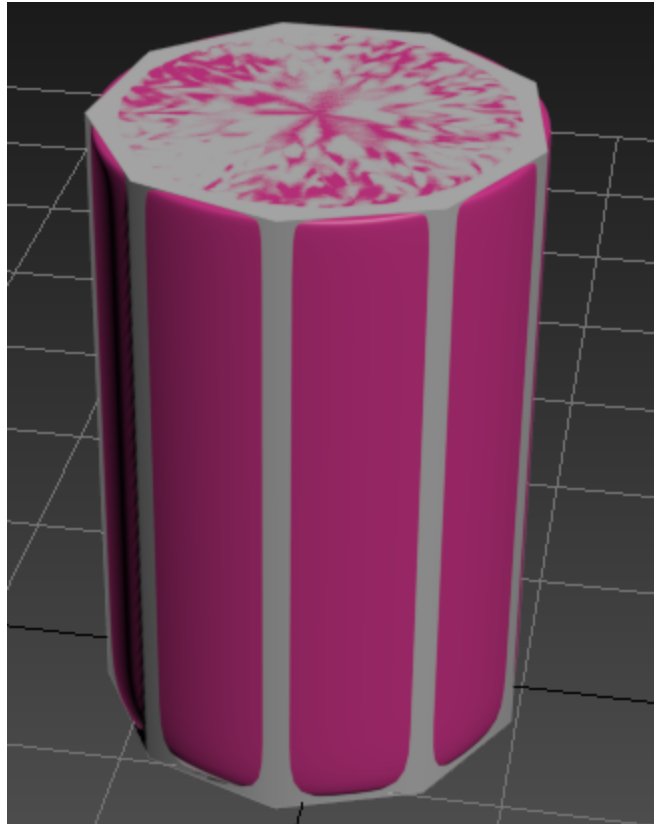
#### *4. Roundness and waviness in normals*

With rounded object, there is a persistent issue of waviness in normal maps.





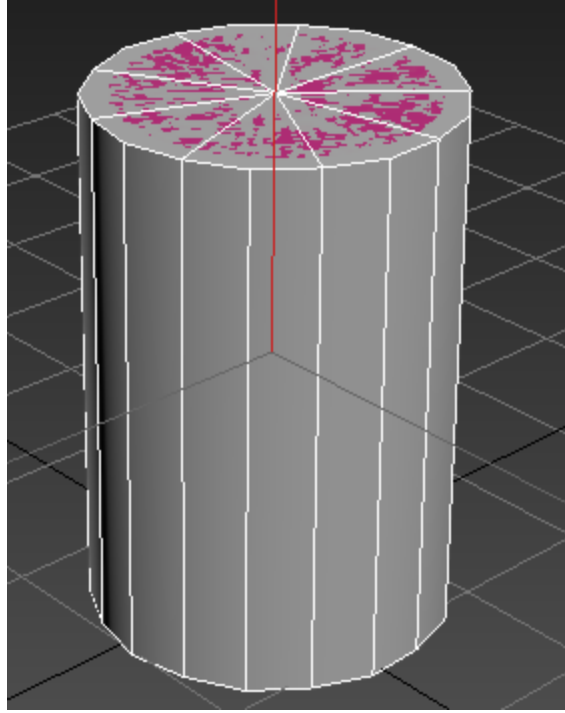
You can see "waves" on the edges of UV shells. They happen when rounded low poly doesn't conform properly to the high poly.

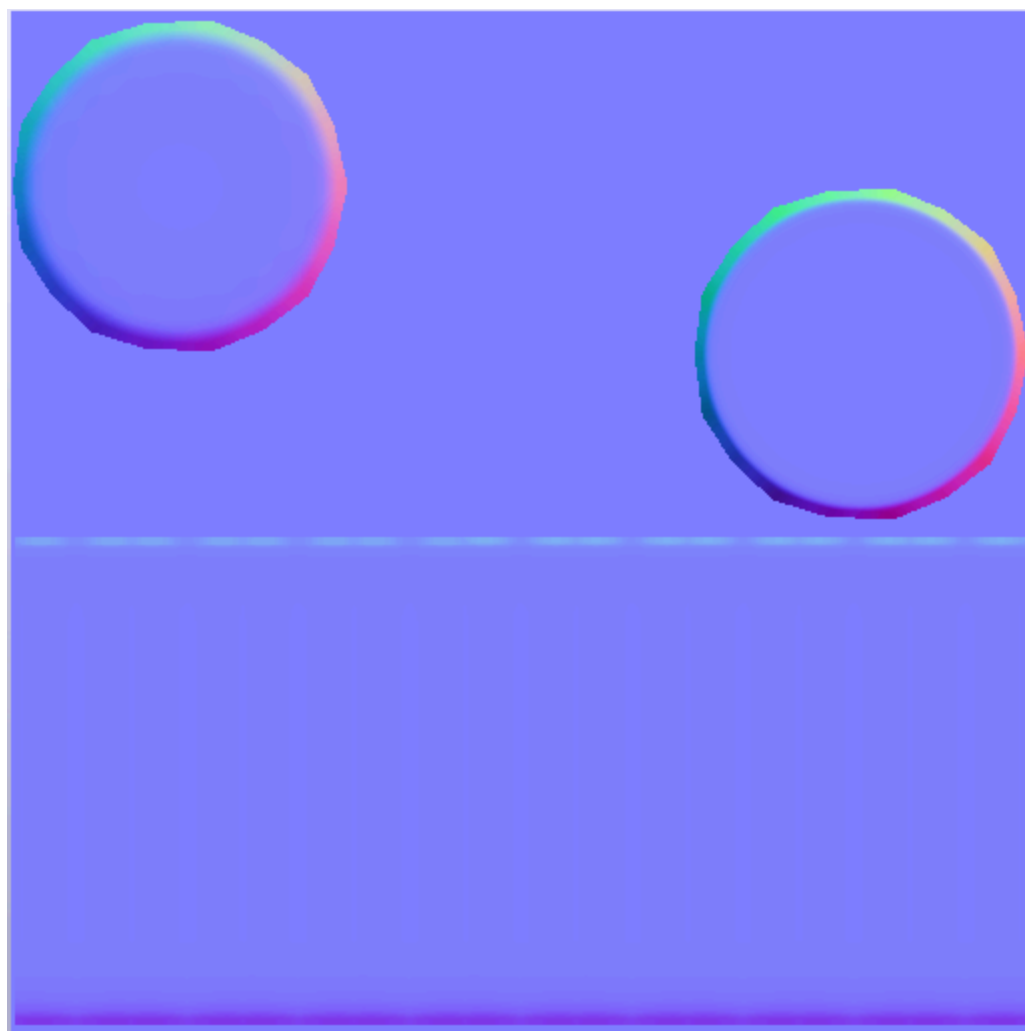




So how do you get rid of these nasty issues?

A good way to fix this would be to add more geometry/sides to the low poly cylinder.







After baking the mesh with more sides, you can delete added edges and you'll get a bit better result than just baking with less sides. It's not recommended, but it's up to you.



## ***5. Proper edge placement on retopology***

When you do retopology, it's important to place edges where your high poly edges are, otherwise you will get projection issues on baking and normal map will not be displayed correctly. The cage creation will be harder too.

## ***6. Triangulating***

Because different baking and game engines triangulate

meshes differently on import, it's important to triangulate your low poly mesh before baking, so there would be no shading differences between various applications.

In 3ds Max, it's very easy to do it:

Select Vertices sub-object, press Ctrl+A to select all vertices and hit Connect to triangulate. Use this triangulated mesh for baking and for displaying in-engine.

## ***7. Best UV practices for normal baking and painting***

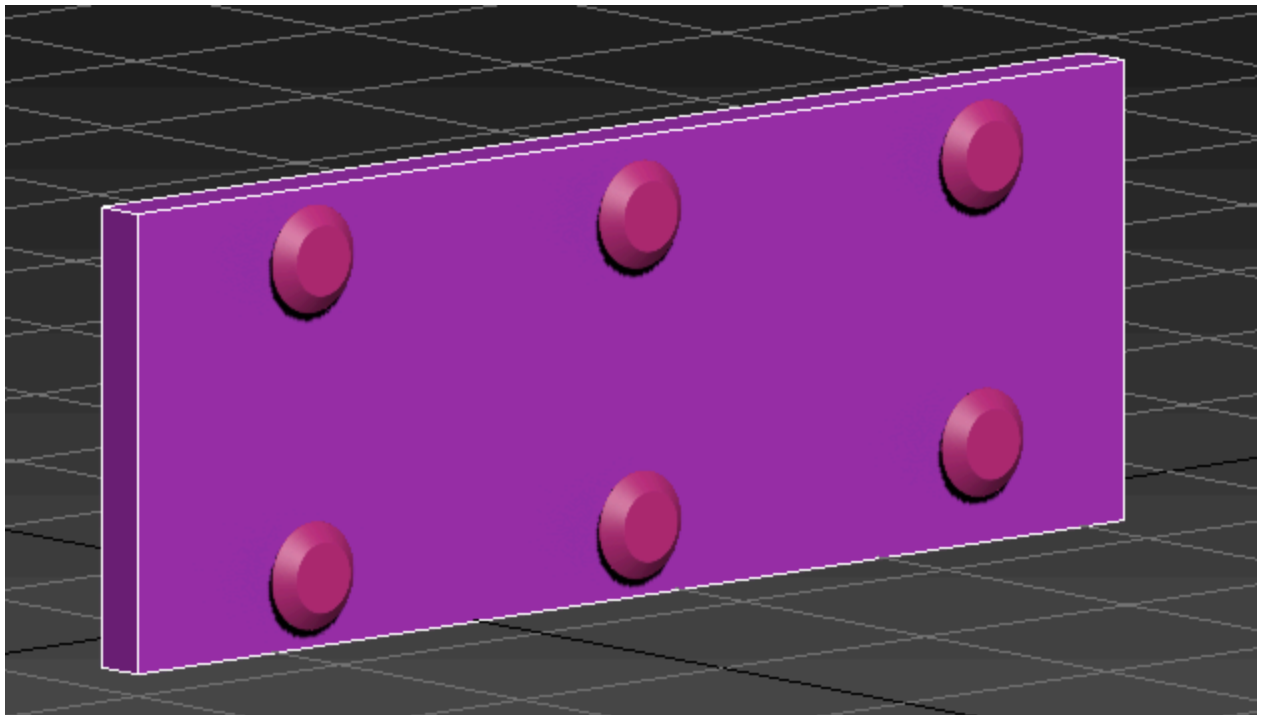
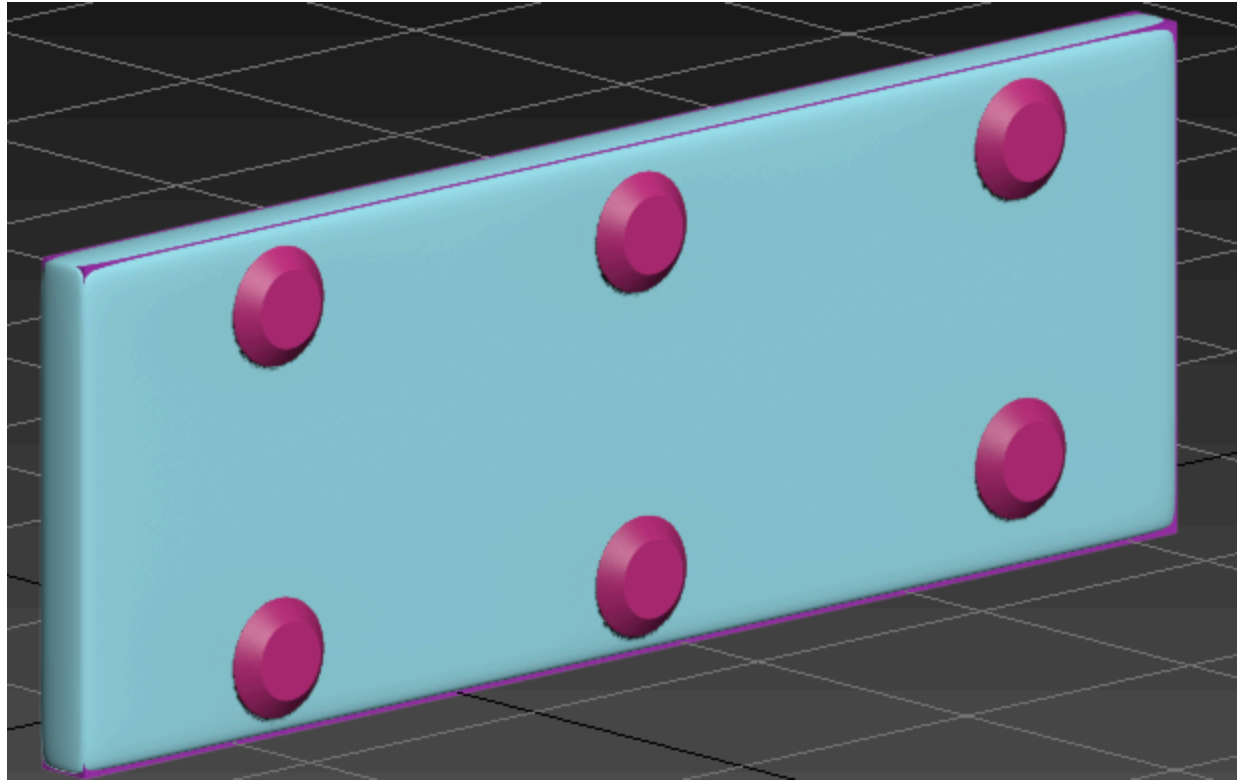
It's important to think about texturing process when you do unwrapping. If you need to paint uniform straight details on a belt, it's important to have belt's UV shells straightened.

When thinking about stitching UVs you should think about if that's part of the mesh will be visible or not, whether or not you can afford to have gradients in your normal map for that part

## ***8. Slanted details in bakes***

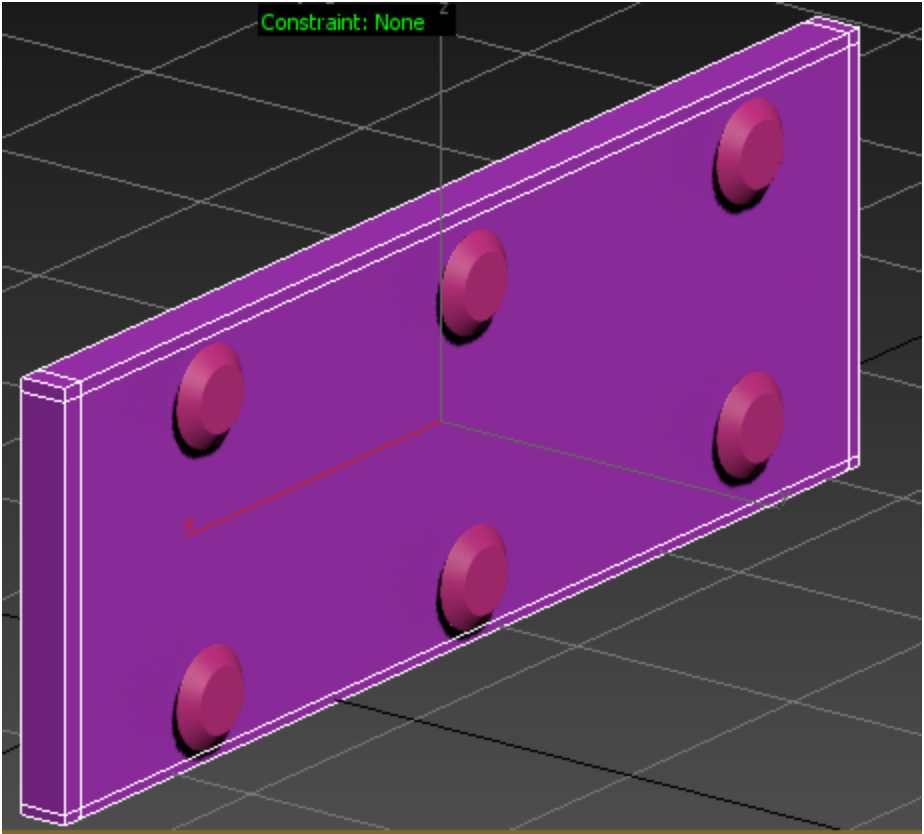
With floaters, you have to keep in mind slanting of those details in normal maps.

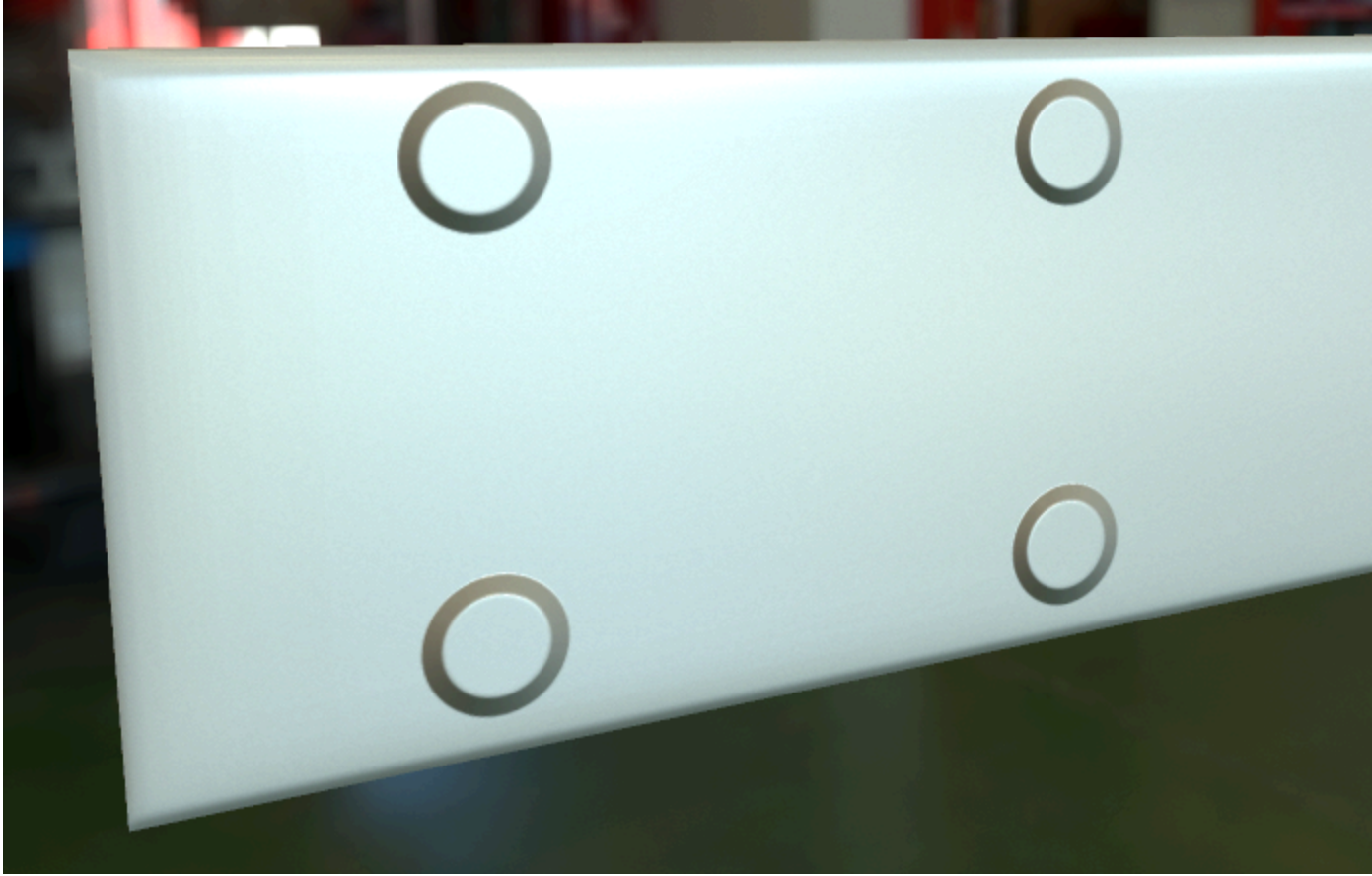






Low poly can also be constructed like that:





It gives you a similar result, but much faster. With the expense of having more polygons/vertices in your low poly mesh.

	Total	Box001
Polys:	41 862	6
Tris:	79 884	12
Edges:	87 564	12
Verts:	39 958	8

	Total	Box001
Polys:	41 886	30
Tris:	79 932	60
Edges:	87 612	60
Verts:	39 982	32

See how big a difference is in Box001's polygon count and verts after you add edge loops? So use this method on your own risk.

In some cases it will help to just add a vertex in the place where you have a floater.

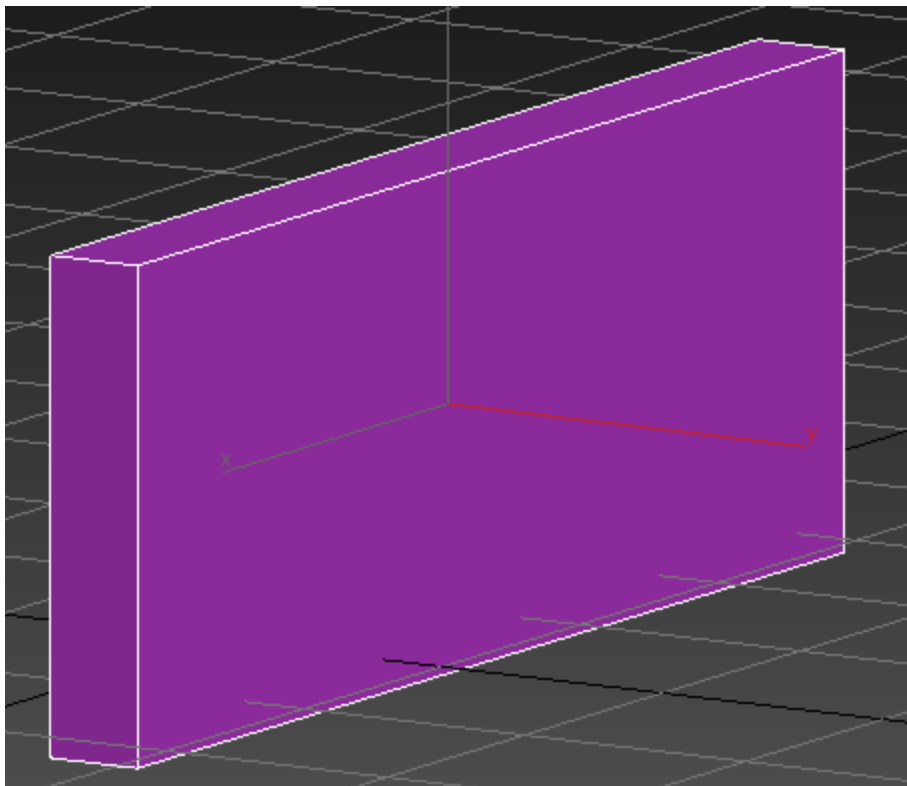
***9. What if I don't want to add more polygons to my low poly, but don't want slanted details either?***

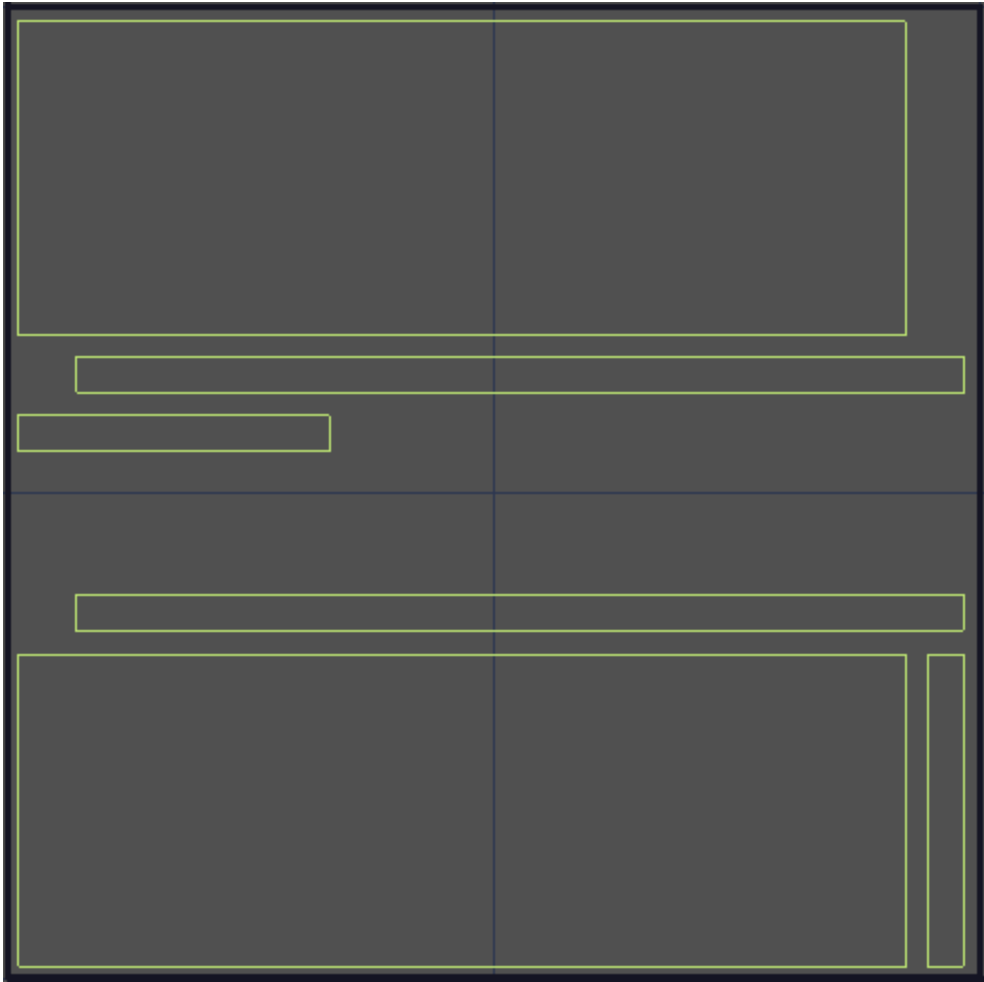
That's an interesting question and there is an answer for that!

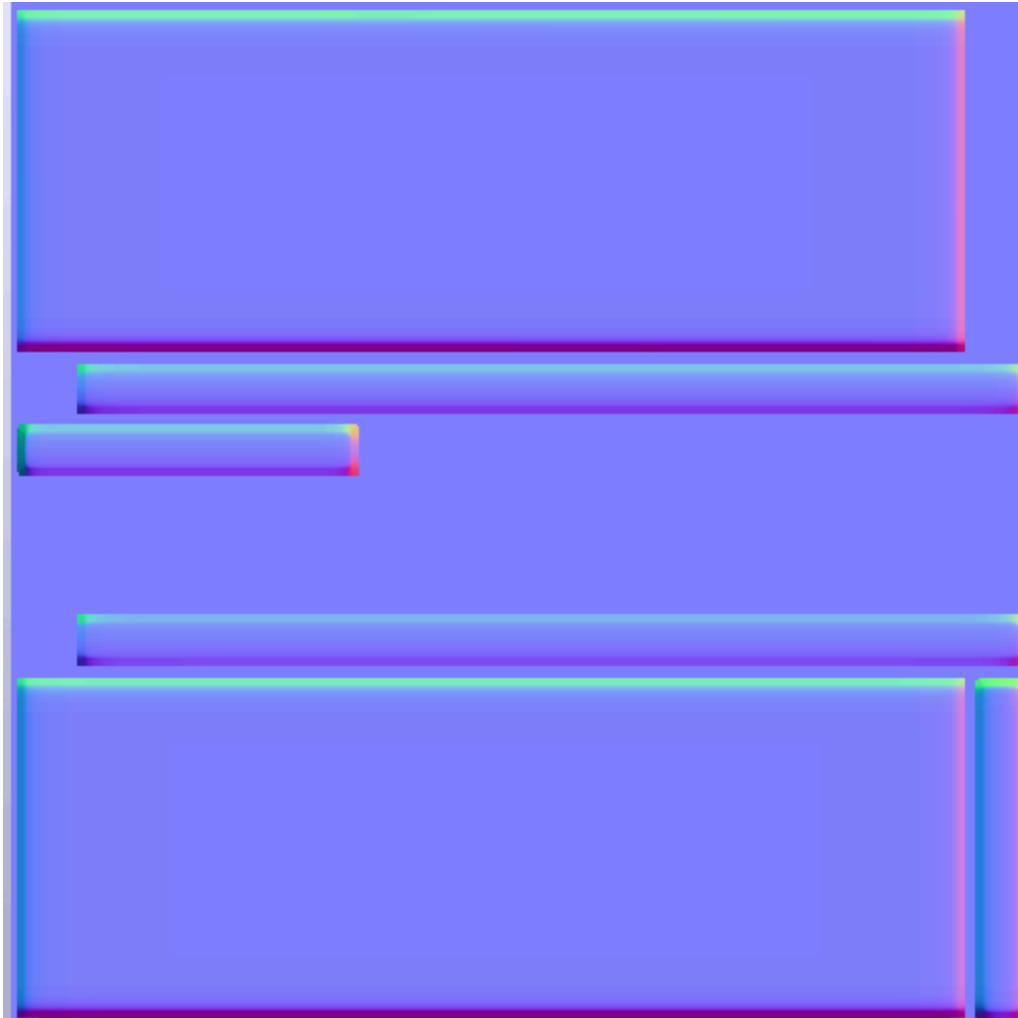
Basically, you place edge loops to help with baking floaters and delete them after baking. But, unfortunately, that won't work in every case.

First of all you have to decide how your gradients will look on the normal map.

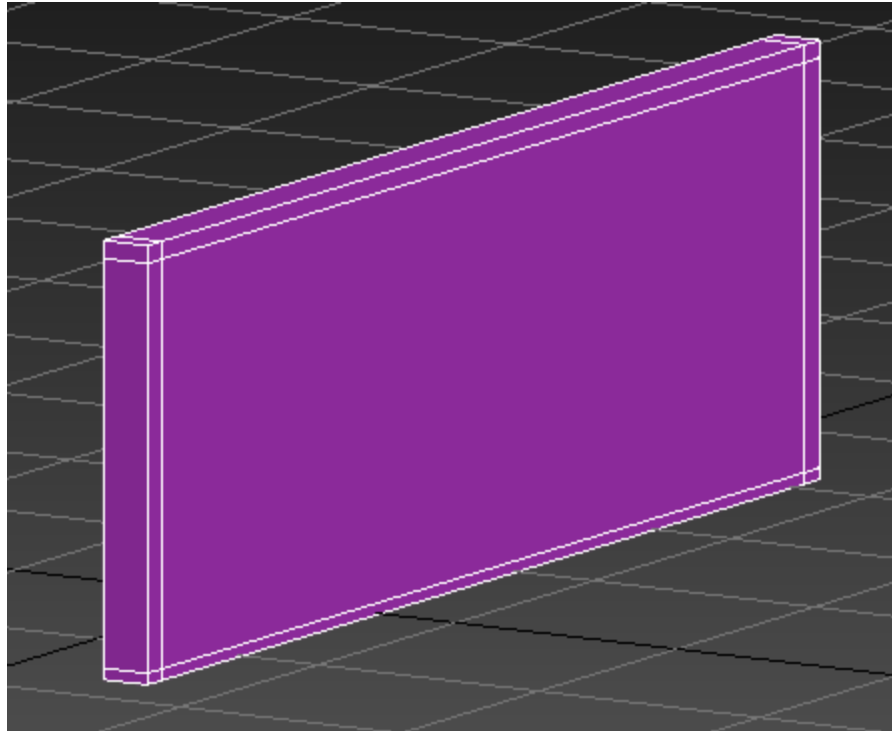
Do two bakes: one without extra edges and one with them and see the difference in gradients in the normal map.



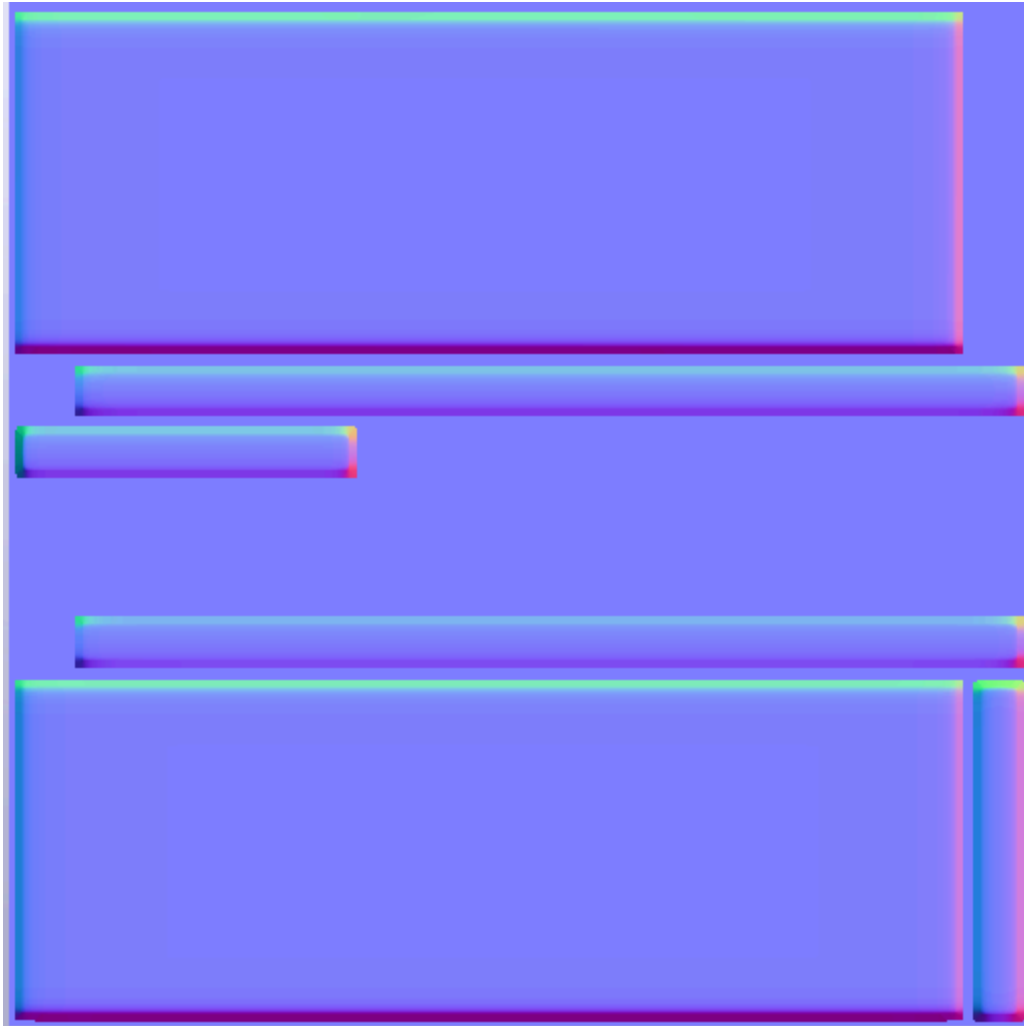




And this is what we get after adding edge loops to help with baking:





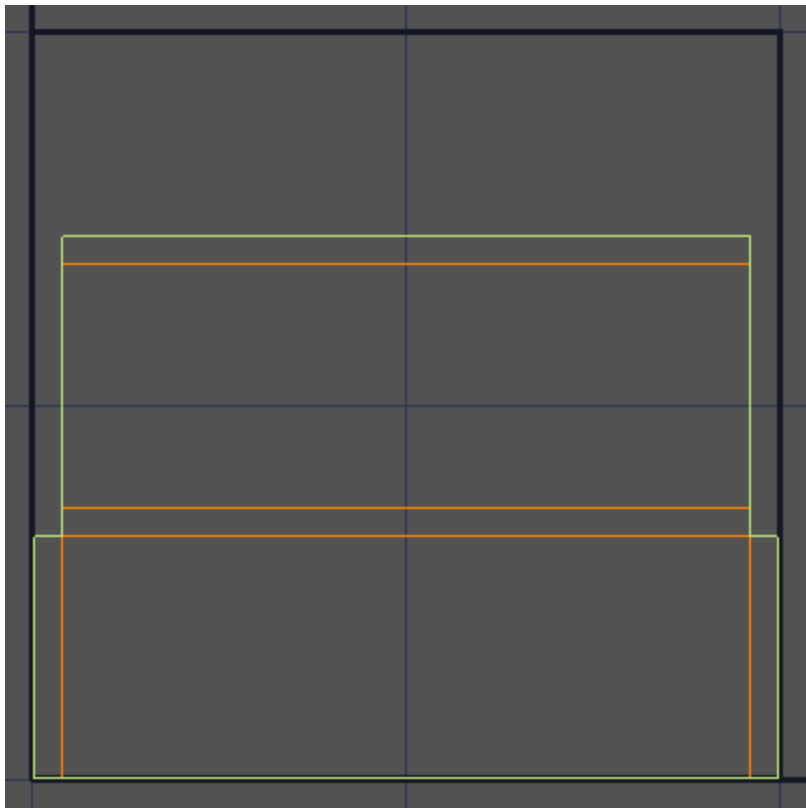
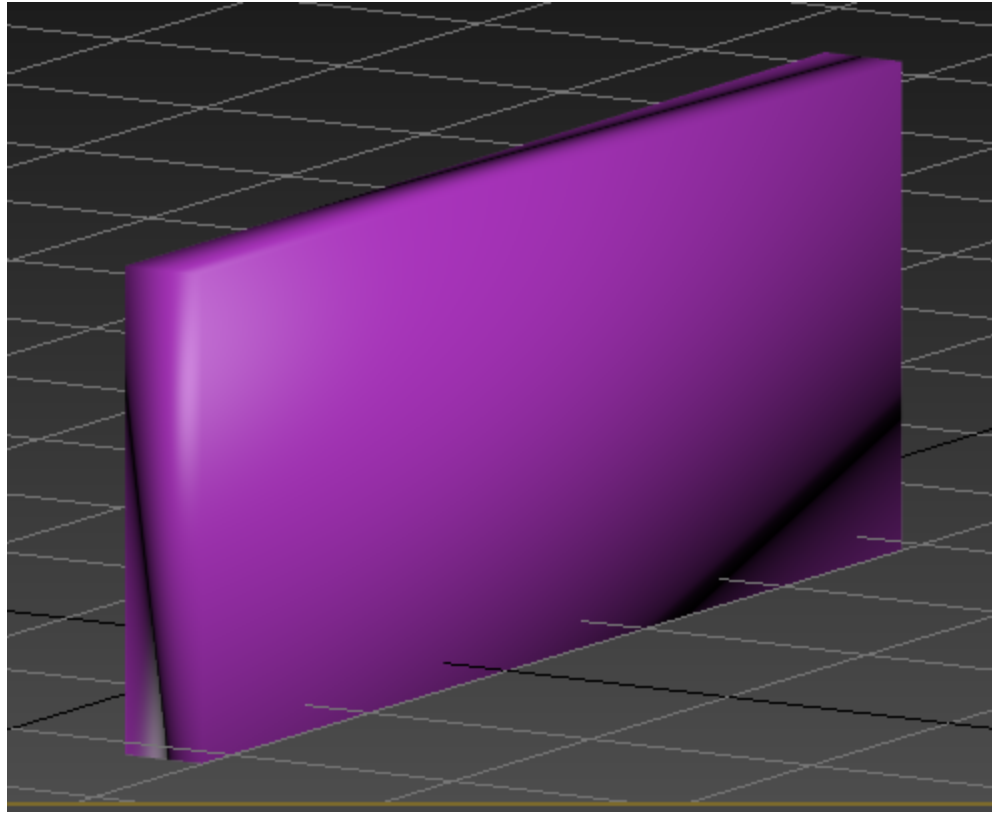


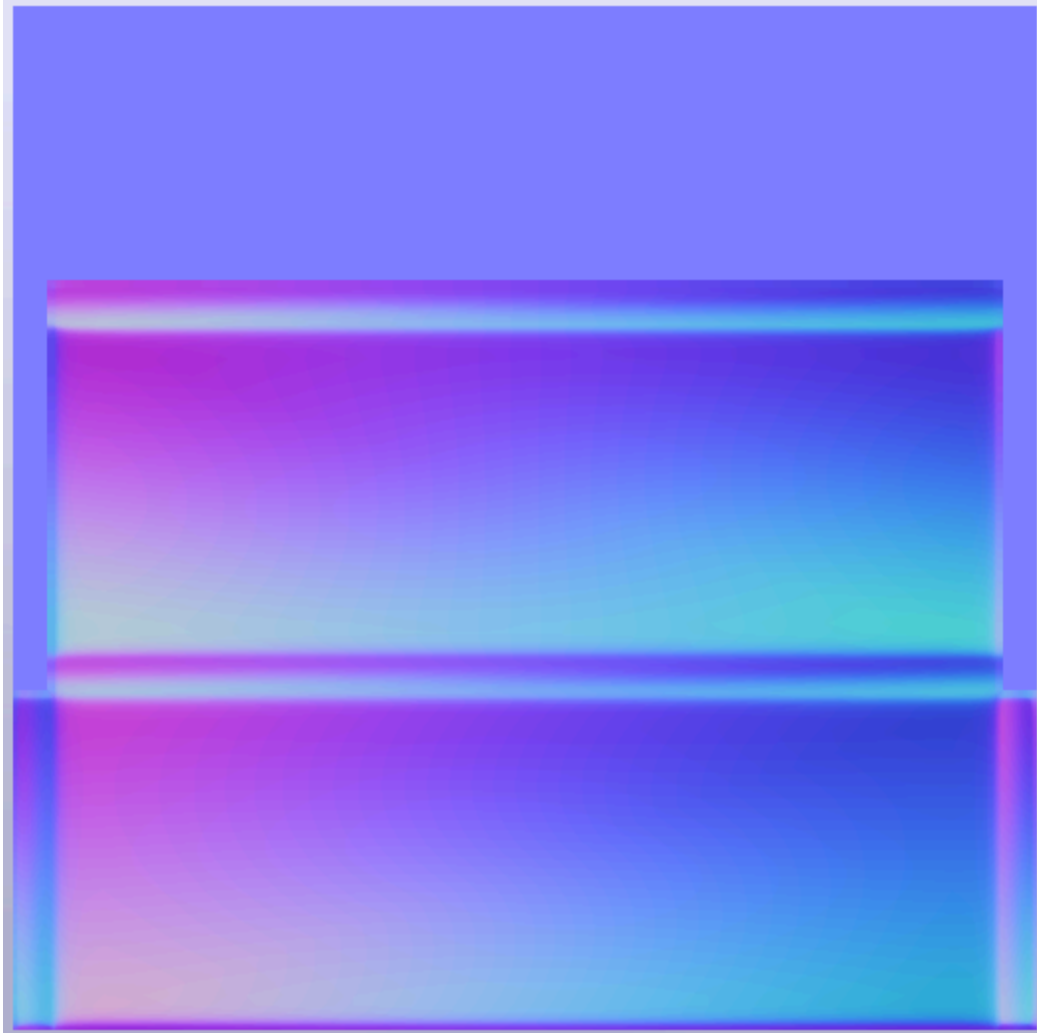
They are practically the same. So why is that? Didn't I just add more loops? Thing is, control loops help ease out gradients on baking and more, but in this case I already had separated UVs and hard edges, so edge loops didn't have to do any work.

So why this is important? If I delete added edge loops and apply normal map in-engine, this is what we get, a perfect result. There's no difference in normals between the two meshes, but one of them has our desired polygon count. Neat.

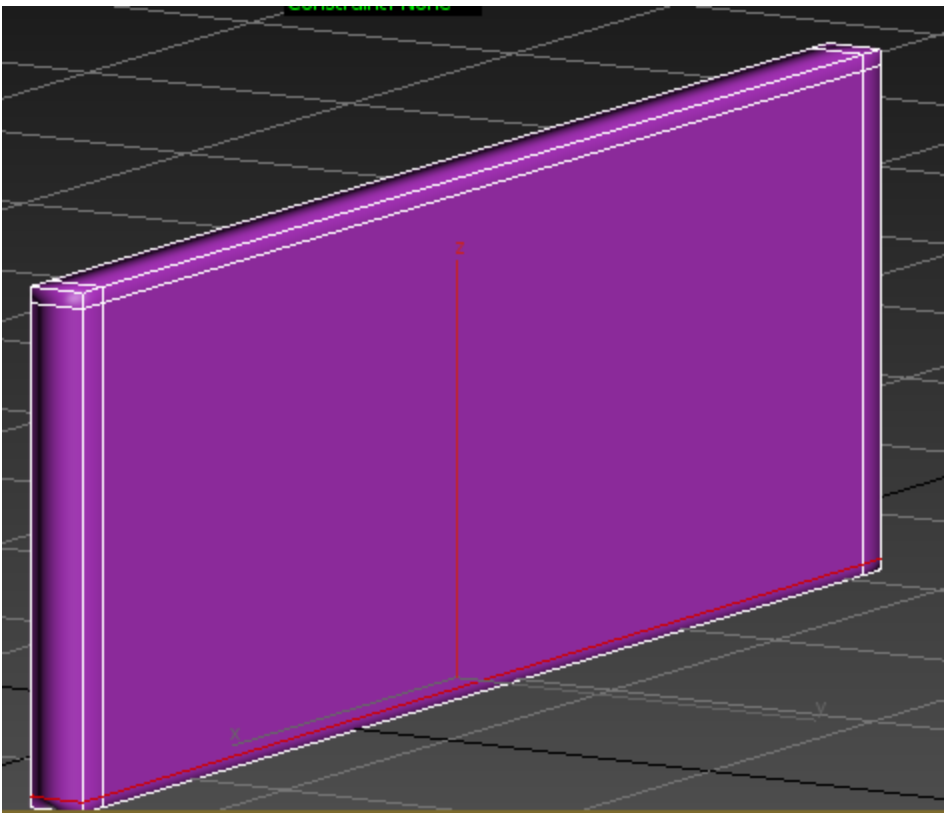


But now let's do another test. This time I stitched UVs and applied 1 smoothing group to the whole mesh, just to introduce extreme gradients to the normal map.





Pretty nasty looking, right? Let's see what happens when you add edge loops to this mesh.





This one is obviously much better looking, this is what we want! But after applying this normal map to two meshes, this is what happens:



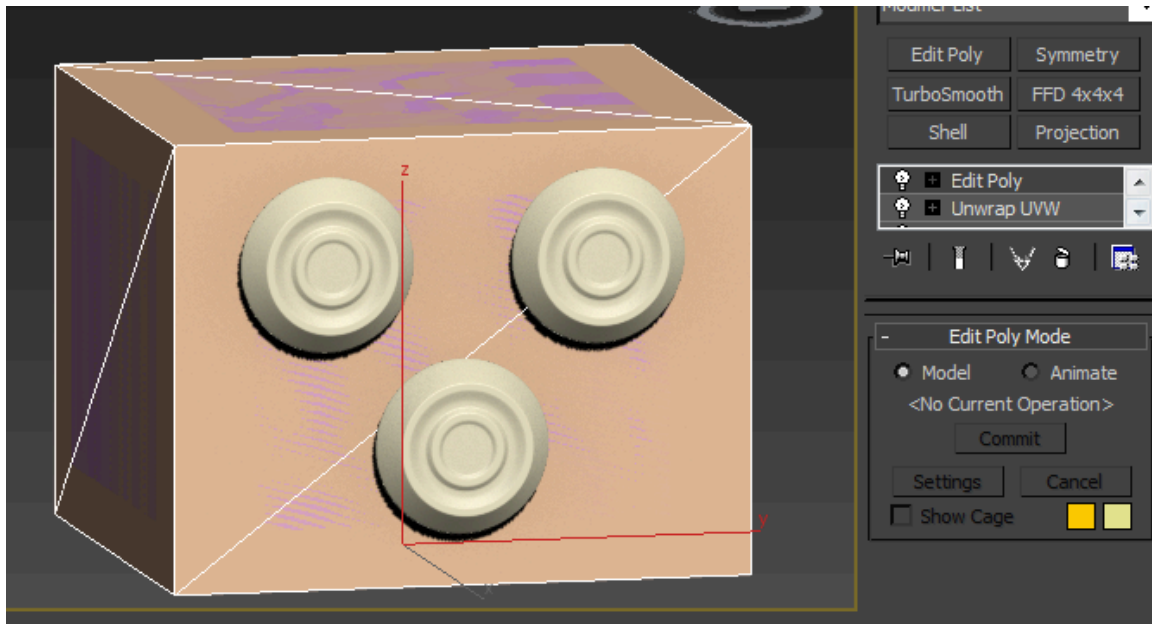
Wow, this is ugly! We broke something!

The thing is that gradients of two meshes became just too different and when you delete extra edges, it doesn't understand how to adjust to it.

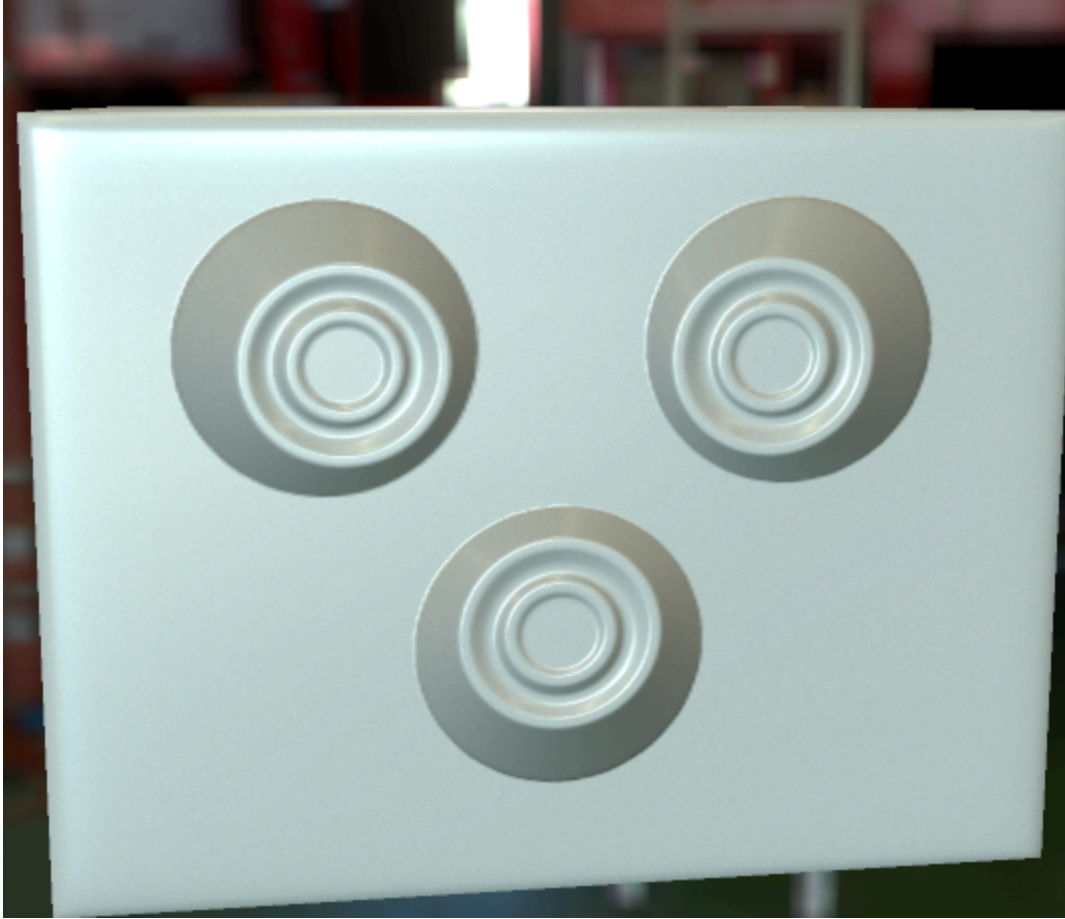
So what to do if I have gradients, but I don't want to do hard edges, split UVs etc.? There is a good way to do it!

## Method #1

It's the easiest way to get rid of all the slanted details even in a mesh where other methods prove to be very troublesome. It's called a Skewmesh method and it works like that:

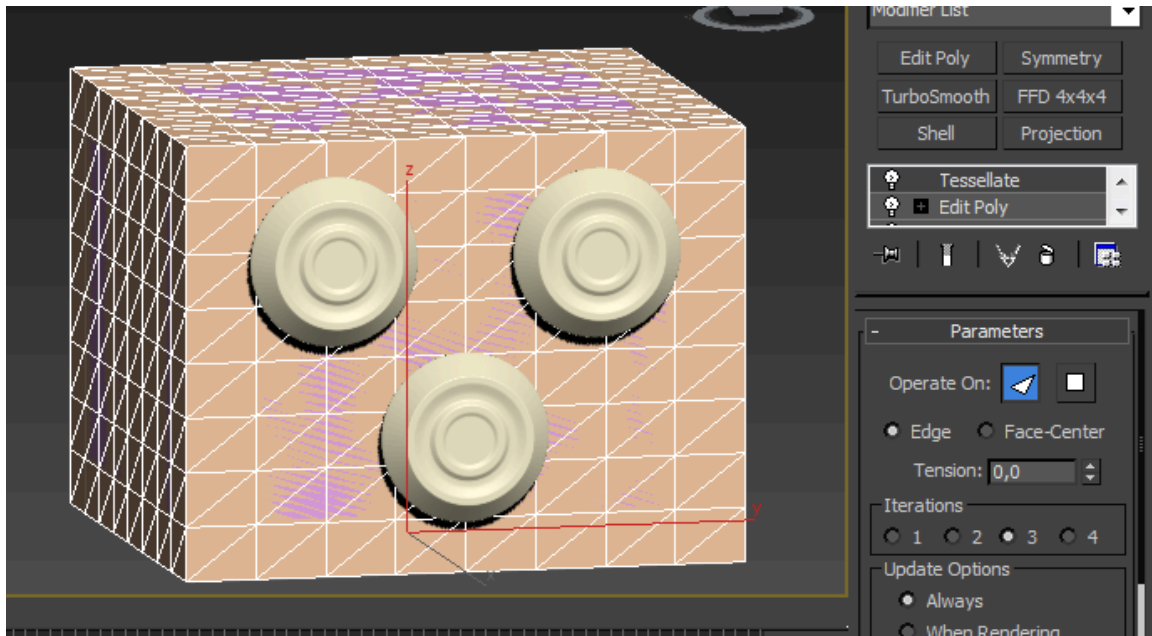






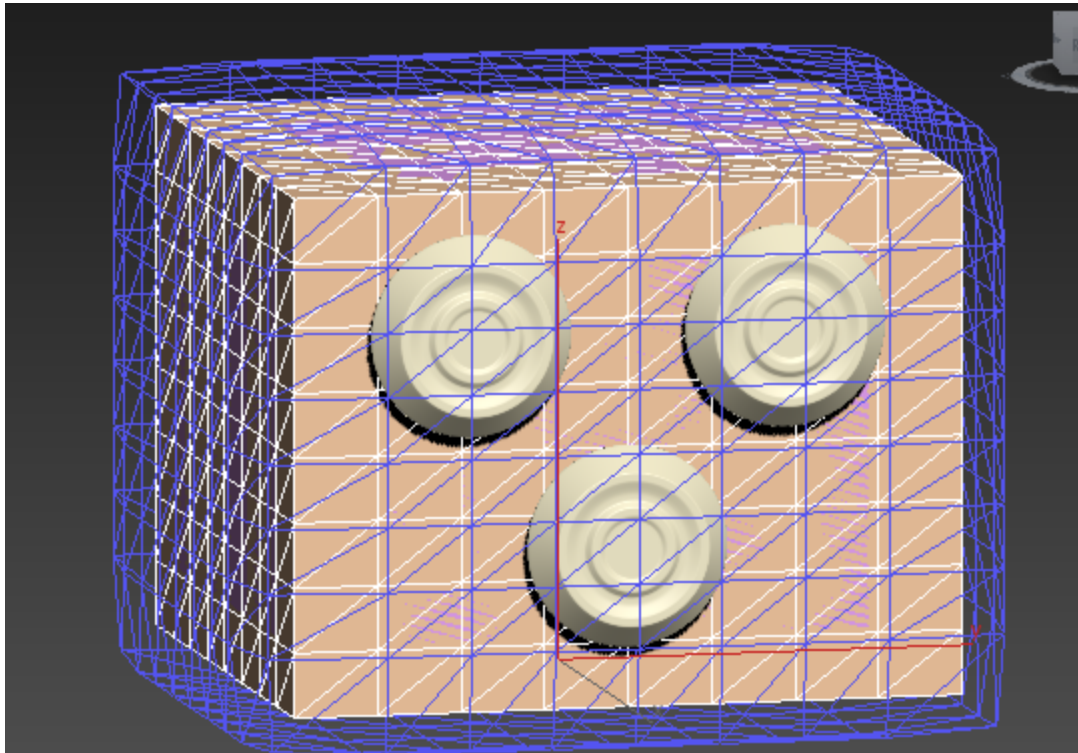
That's what we expected. Now let's fix it.

1. Triangulate your low poly mesh.
2. Then apply Tesselate modifier on top of it

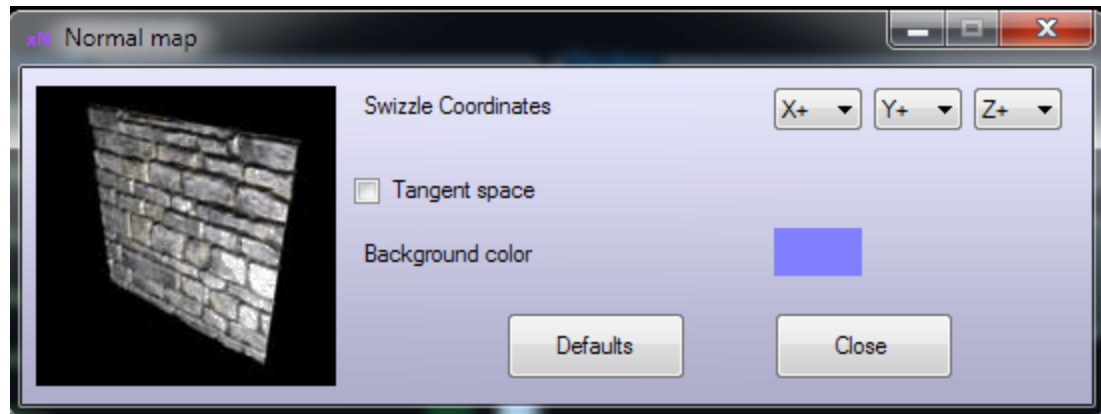


Make sure you have 0 in Tension slider and Iterations are 3 or 4

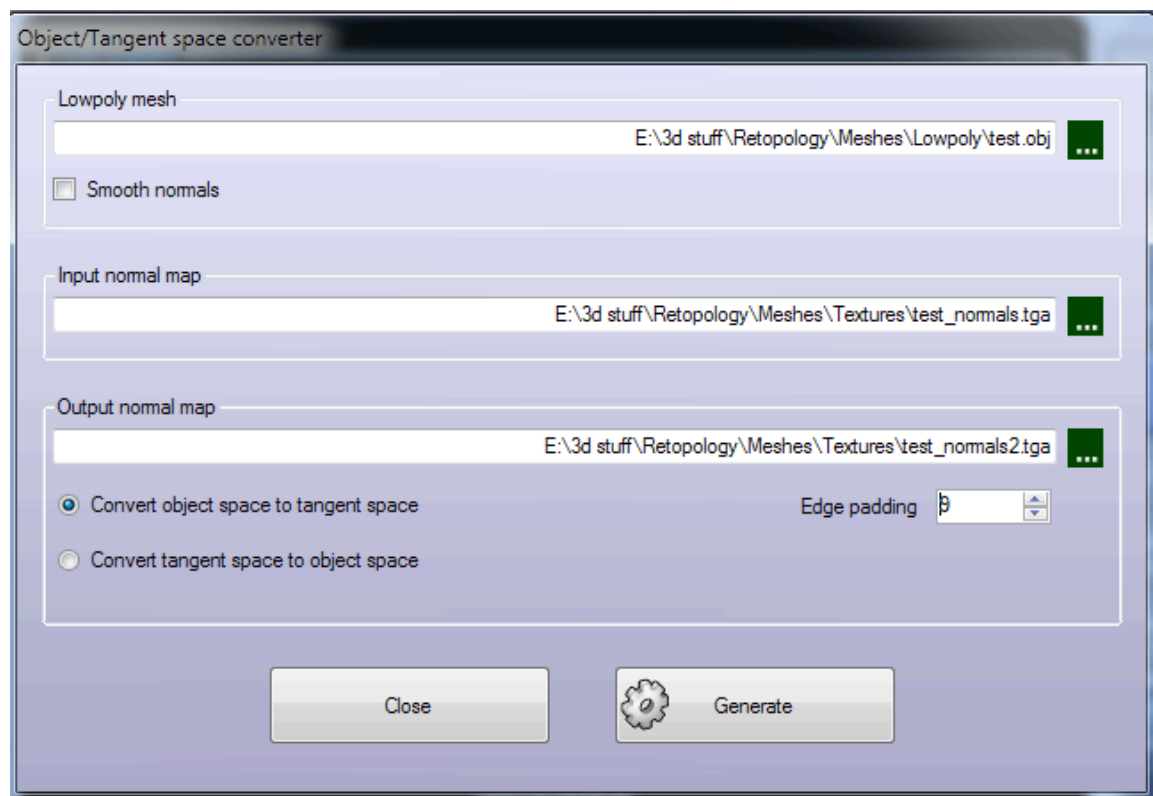
3. Then make a cage



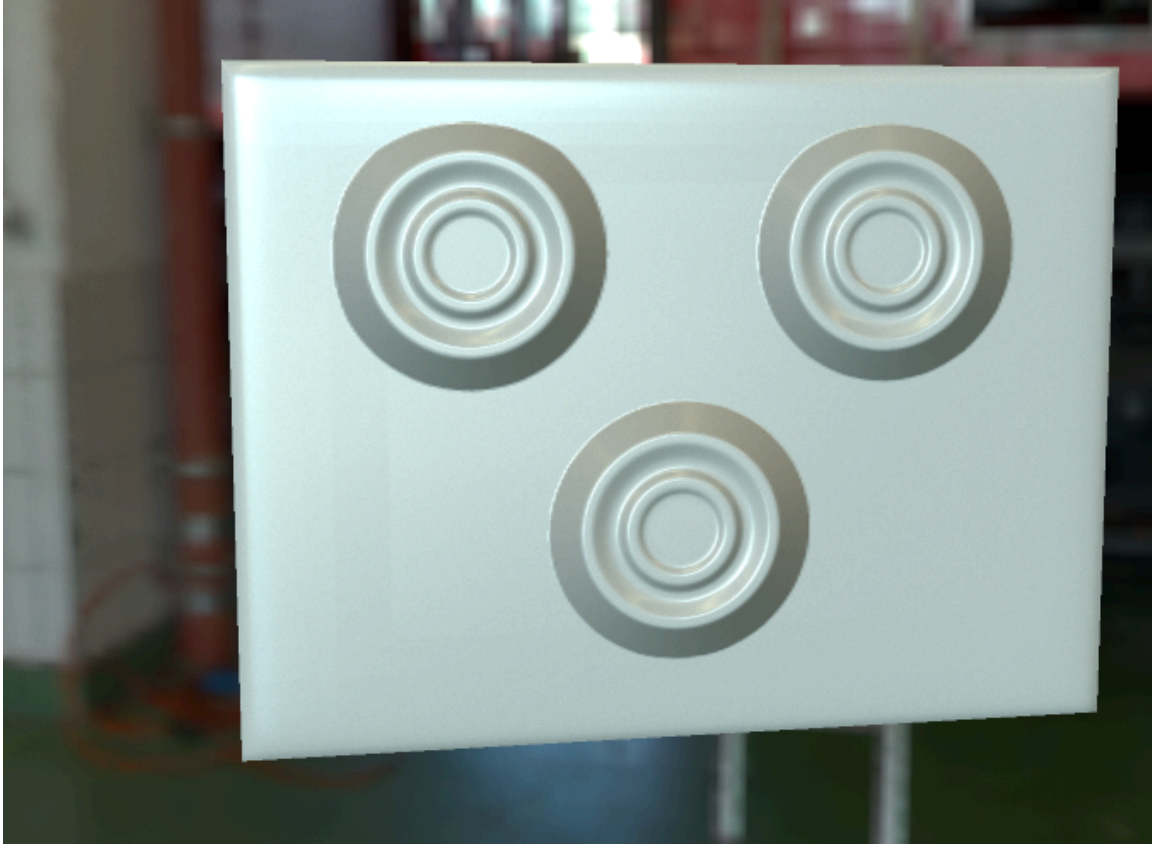
4. Export tessellated low poly and its cage separately from the final geometry mesh and import them to Xnormal.
5. In Xnormal, bake Object Space Normal map



6. Then convert this map to Tangent Space map, using your Object Space and your final geometry mesh as inputs in the converter.



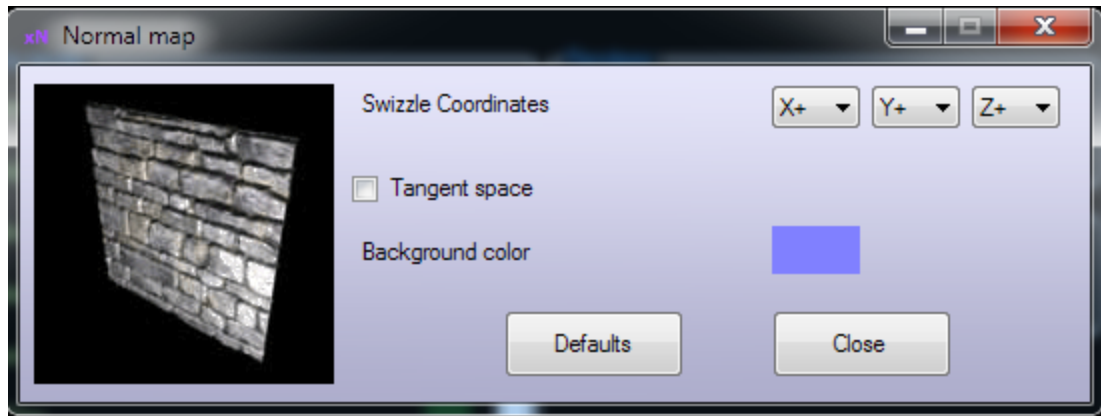
Done!



## Method #2

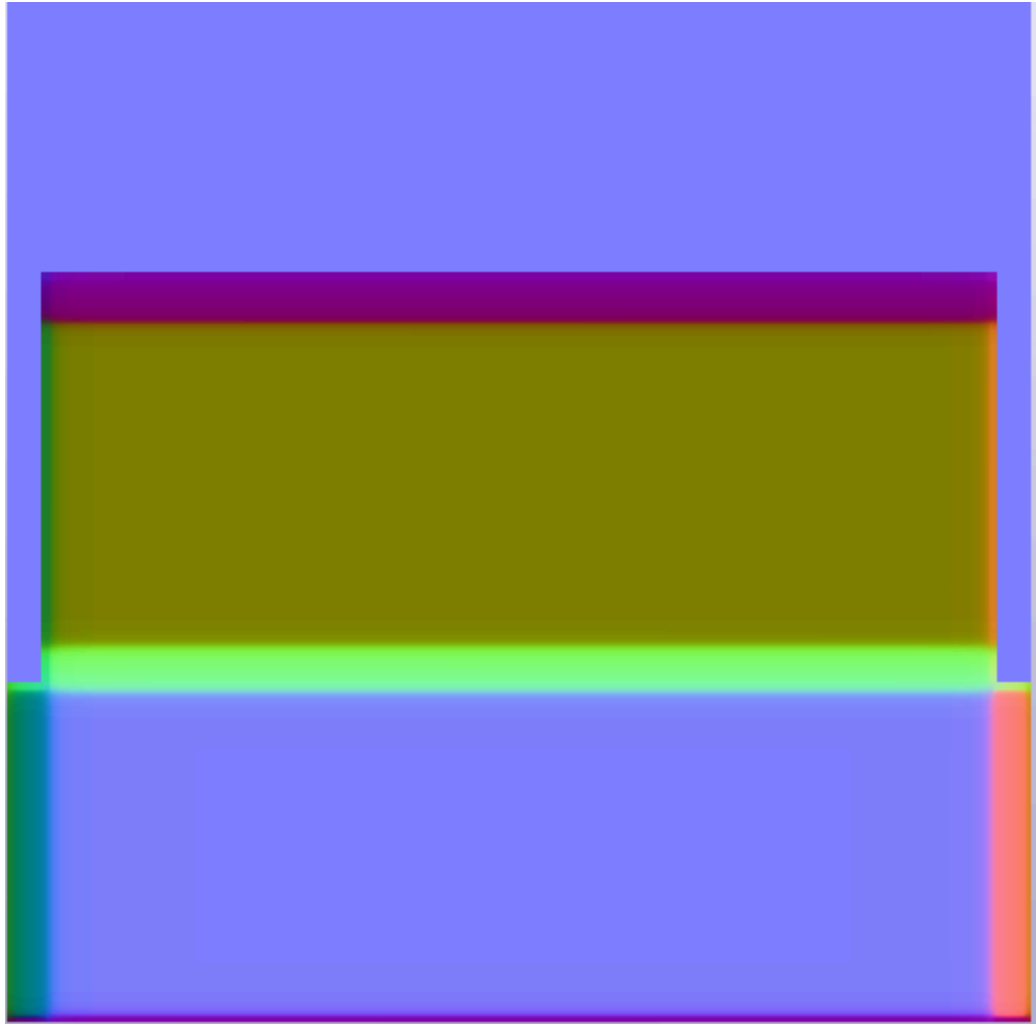
1. Prepare two meshes: one without additional edge loops (your final game model) and a baking model with edge loops.

2. You will have to bake an Object Space Normal Map instead of our usual Tangent Space Normal Map.

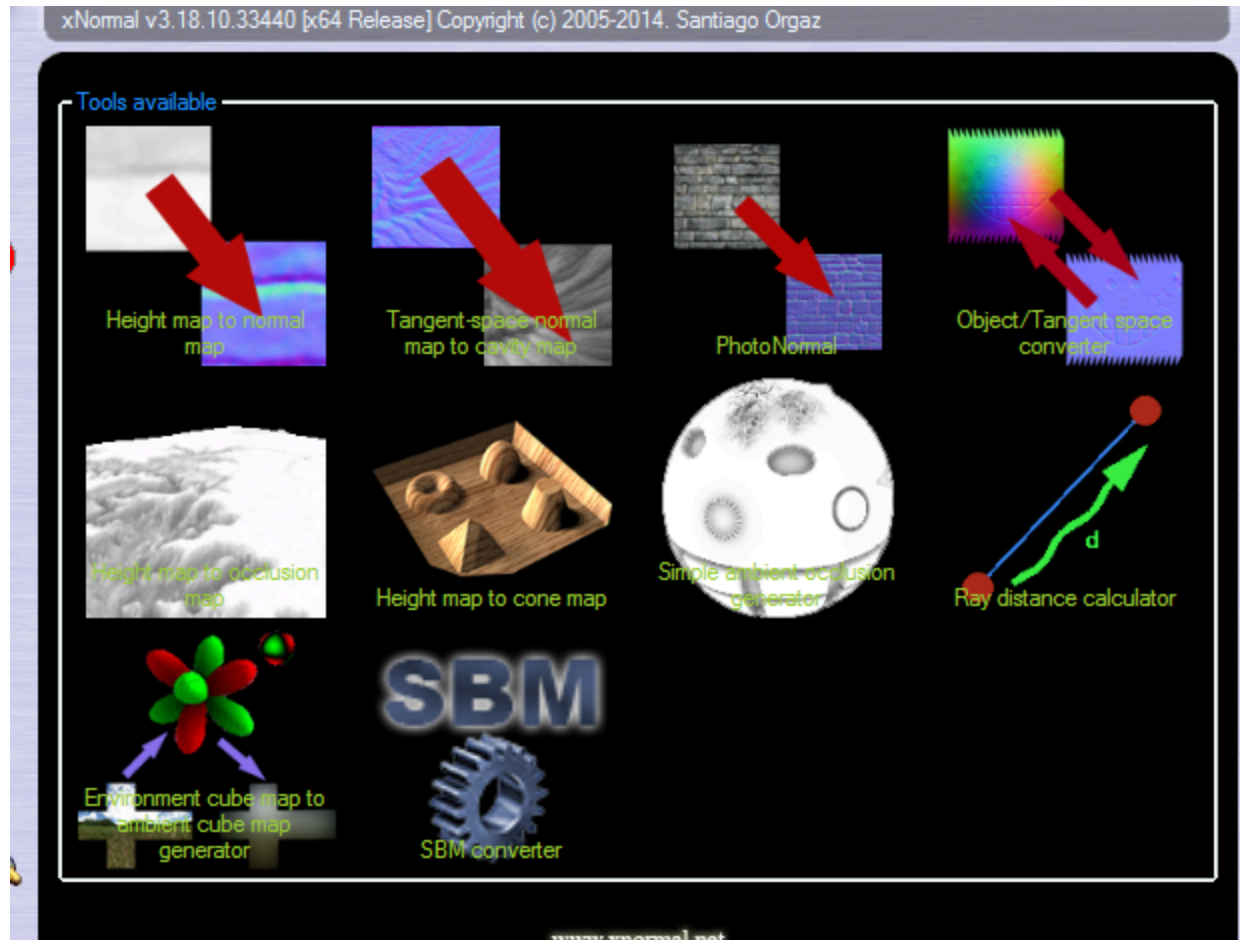


I will show the process for Xnormal. To bake Object Space map you'll need to uncheck the Tangent Space box in Normal Map settings.

3. Choose your baking model (the one with edge loops) and bake it.



This is how an Object Space map looks, it all weird and yellow. But that is not important. What is important is that it can be converted to Tangent Space map now.



#### 4. Select Object/Tangent space converter

Load your final game model in Lowpoly mesh slot.  
Object space map in Input and choose format in which you  
want to save the final map.

Lowpoly mesh

E:\3d stuff\Retopology\Meshes\Lowpoly\1.obj ...

☐ Smooth normals

Input normal map

E:\3d stuff\Retopology\Meshes\Textures\test\test2-4\_normals.png ...

Output normal map

E:\3d stuff\Retopology\Meshes\Textures\test\bible3\_normals3.tga ...

☒ Convert object space to tangent space

☐ Convert tangent space to object space

Edge padding 6

Close

Generate



This is our final result with all its gradients in place. So, what the difference, ask you?





You see, now there's no difference in normals of two meshes. This is what happened:

The converter takes Object Space map that was baked from baking mesh and created Tangent Space map with respect of the shading from game model. Now you have all the benefits of the edge loop baking without having to take care of your gradients first!

## Method #3

You can avoid going through that painful method by doing this:

1. Make hard edges for low poly and split UVs to relax gradients
2. Bake in Xnormal without a cage, use Ray Distance Calculator for better results.

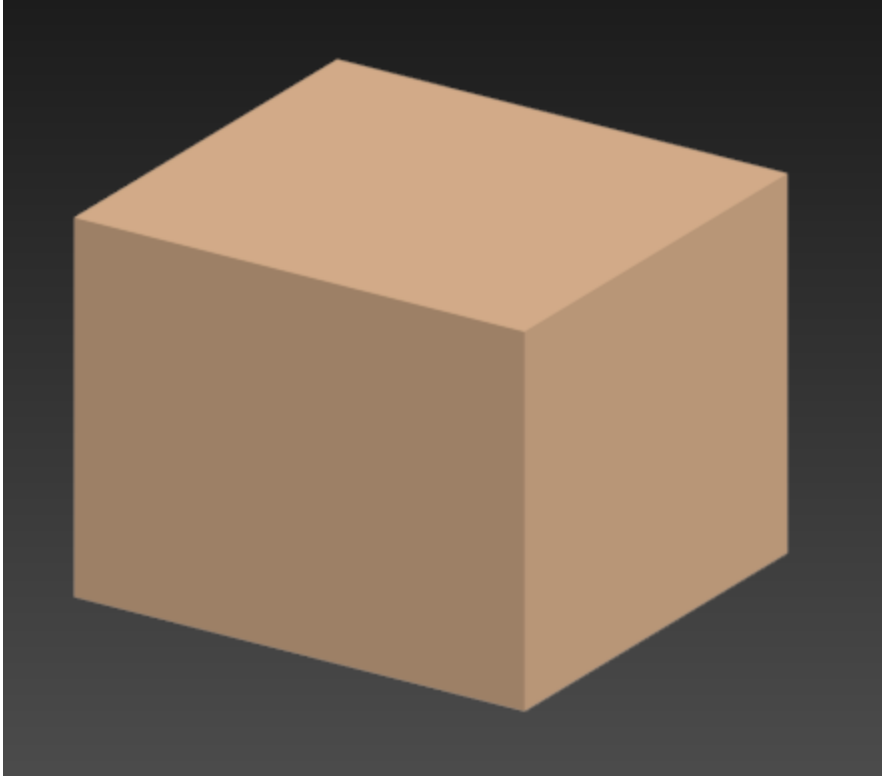
Now you have a clean bake without floater details being slanted. Now you can just overlay the portion you need in Photoshop or use the Object Space method written above if you need to transfer bake to a differently smoothed low poly mesh.

## ***10. Overlapping UVs non-mirrored UV shells***

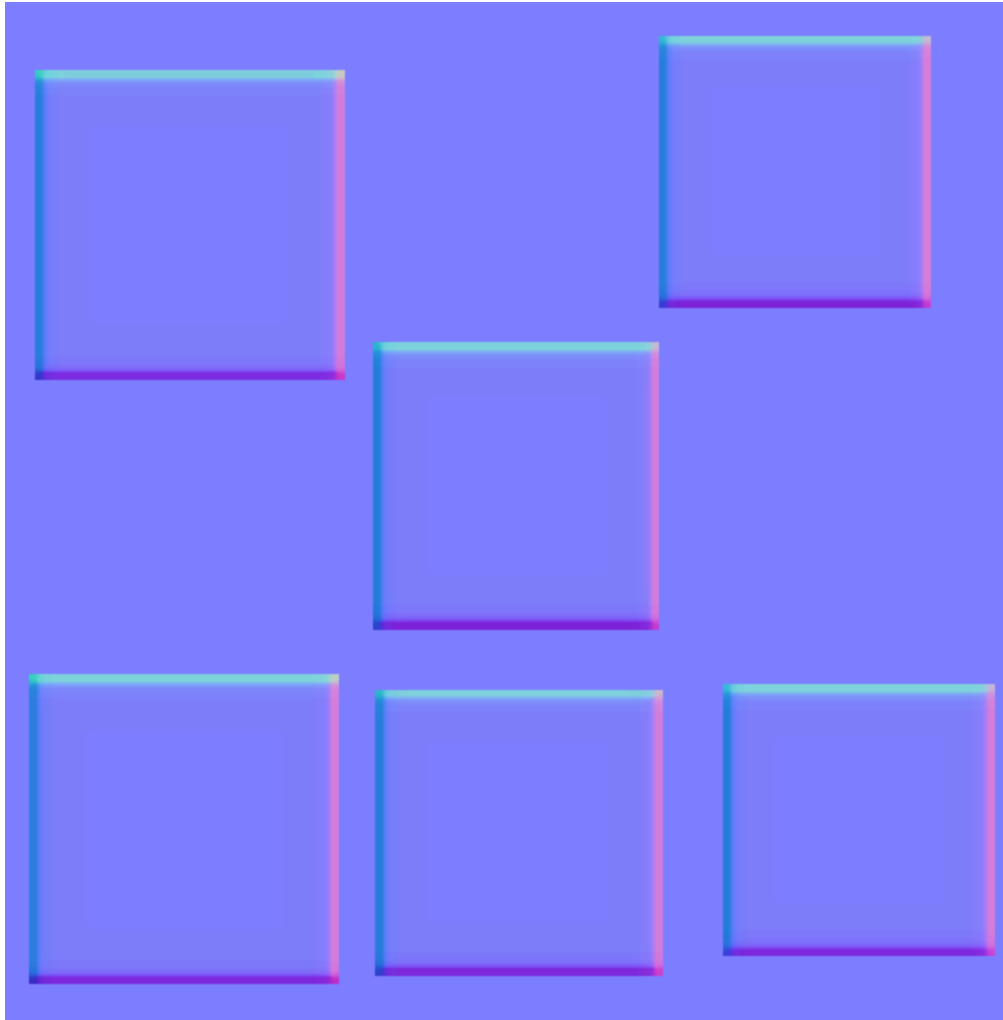
Sometimes (or every often) you have a UV layout where you see a lot of identical or near-identical UV shells and you think "wow, that's a waste of my precious UV space!"

Luckily, there's a simple method to overlap UV islands that belong to the same mesh and get acceptable results.

Let's say you have a stretched cube.



If you separate every face in UVs to take advantage of hard edges you will get 6 islands. This is what you see when you bake it:



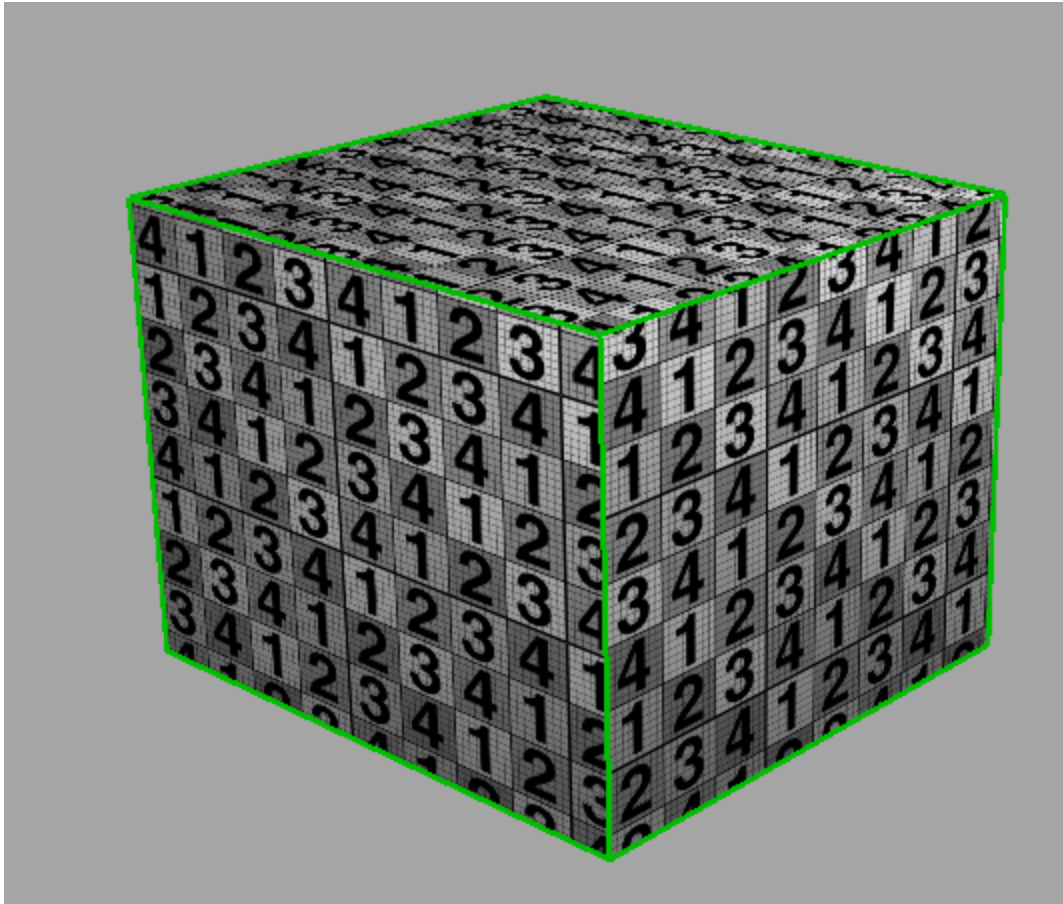
You can see how the islands look practically identical here? One square island looks exactly like other three and two stretched islands also look identical.

### **I really need all these islands? Let's stack!**

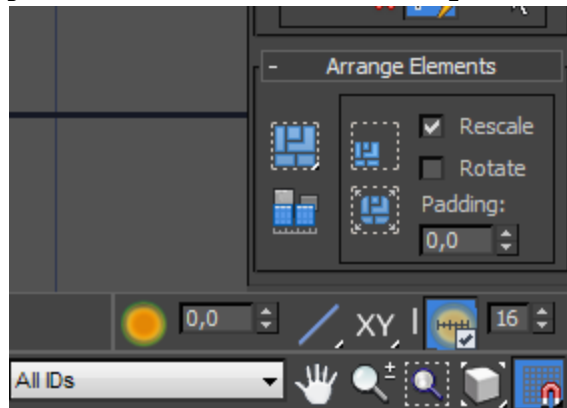
We will do that, but first you have to think about how you are going to texture this asset. Do you need to have unique texture details on every face? Is it important to keep them separate for normal details, scratches etc.? Will mirroring be easily seen?

If you decided to overlap islands, then the process is actually very easy, though there are things that you should keep in mind working with overlapping.

1. Keep in mind the direction of faces you overlap. Especially with simple square faces like that it's easy to rotate them unintentionally and so your textures will be inverted where you don't need it.



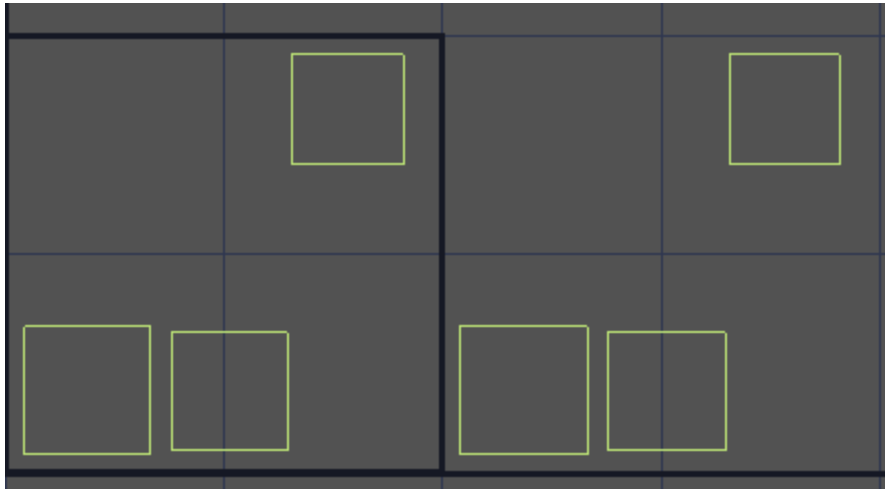
2. Overlapping UV islands is very easy. In 3ds Max you just need to make sure you have Snap activated.



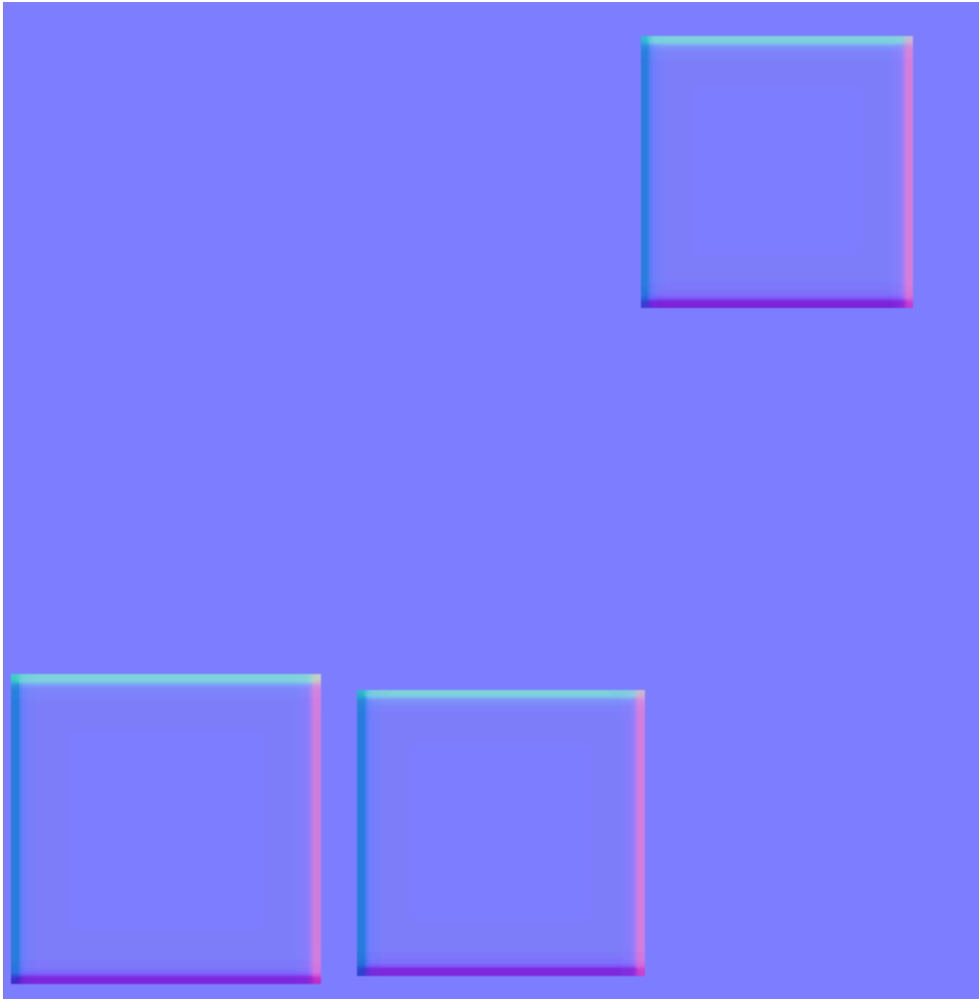
The icon at the very right bottom of the UV Editor window.

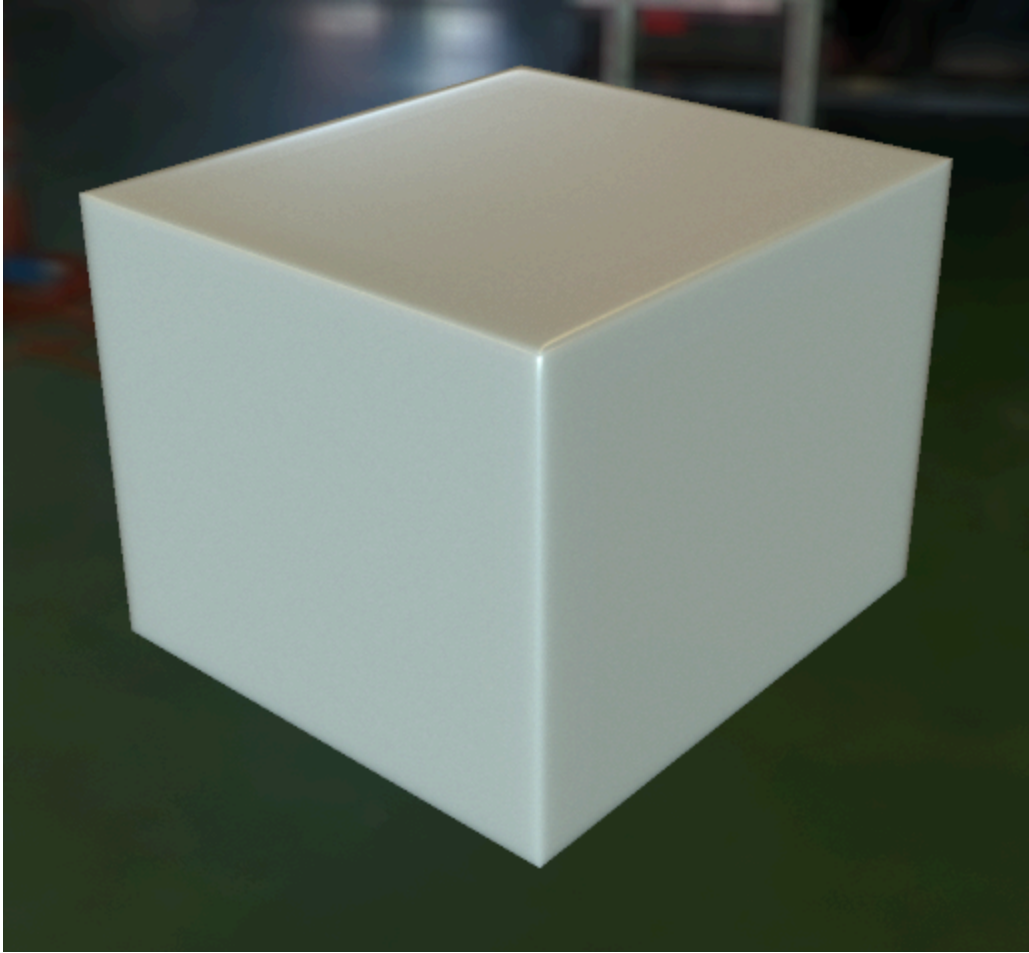
Then you snap two shells together by selecting Select by Elements and dragging one island to another to match the vertices. Done.

### 3. Offset UVs to the near 1-0 space



### 4. Bake







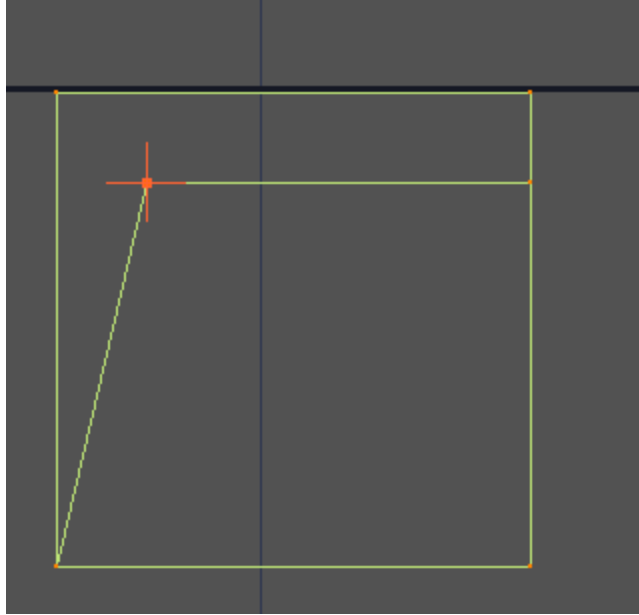


A clean and good looking result. I applied a quick normal map for the second screenshot so you could see how it looks like.

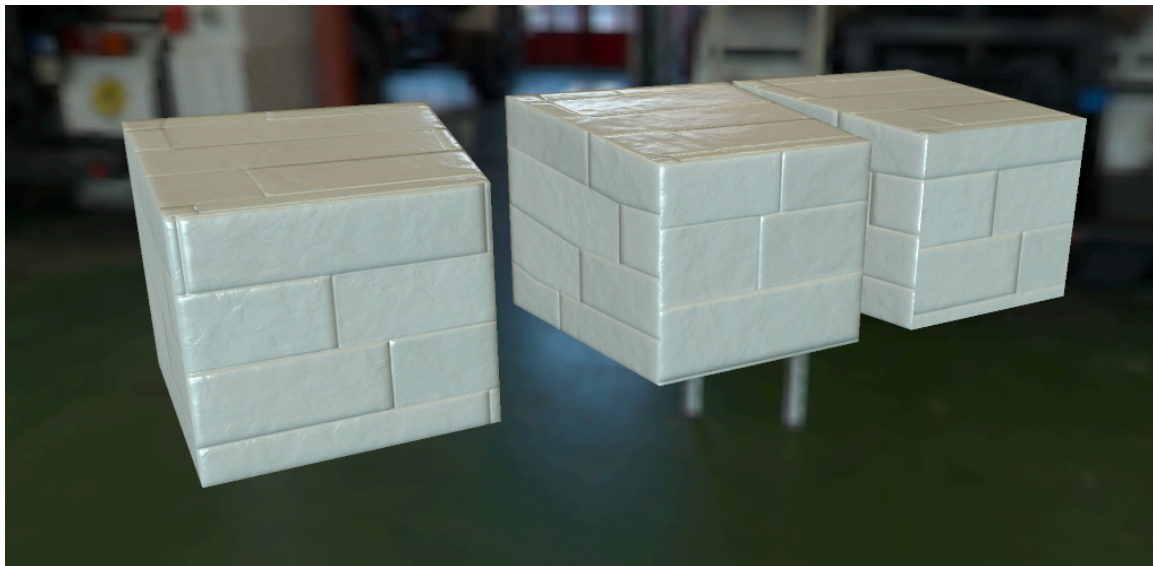
**But what if my islands are not identical, but I want them to overlap?**

You can do that, but you have to be extra careful with that. You have two choices when it comes to overlapping UVs in that case:

Stretching one island to match another:



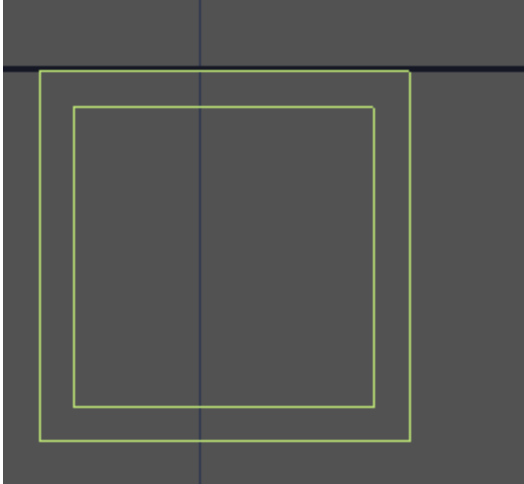
This works fine with a simple square UVs, but will probably be an issue if you try to stretch something more complex and it introduces UV skewing and stretching in the process. Try it and see for yourself. You also have to be aware, that scaling and stretching an island makes it a different size, so it takes unproportionate amount of texture space and you get this:



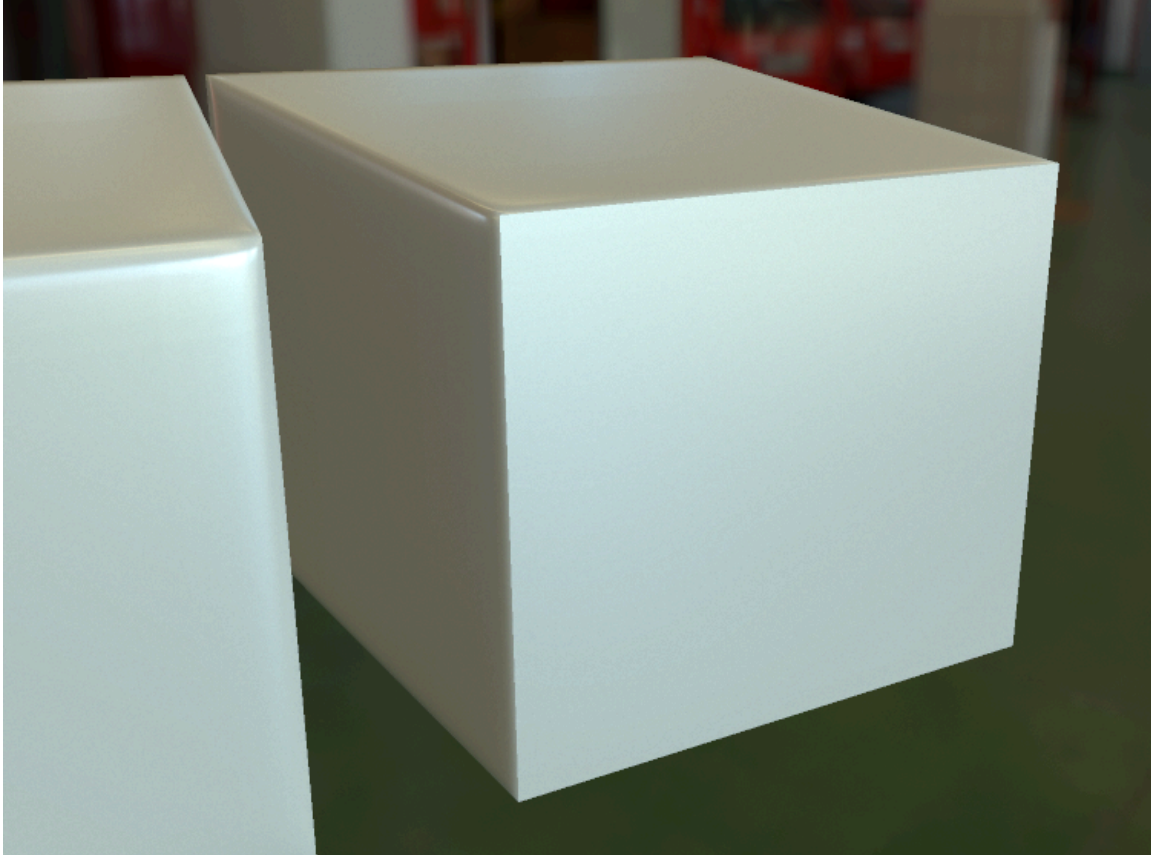
Left- stretched UV; Middle - original UVs; Right - one island inside another

You can see that the size of normal details on the left is different from the original UV layout. It can be used to your advantage, if you are smart about it. Just be aware of this when you overlay your islands.

Then you have this, positioning one small island inside the one bigger.



This is what you get for doing this



Because the small island doesn't share the same edge normals, it can't properly display them in-engine and so you get a nasty looking very hard edge.

# Baking

## *Reset Transforms*

Before baking, make sure your low-poly model's transforms have been reset. ***This is very important!*** Often during the modeling process a model will be rotated and scaled, but these compounded transforms can create a messy local "space" for the model, which in turn often creates rendering errors for normal maps.

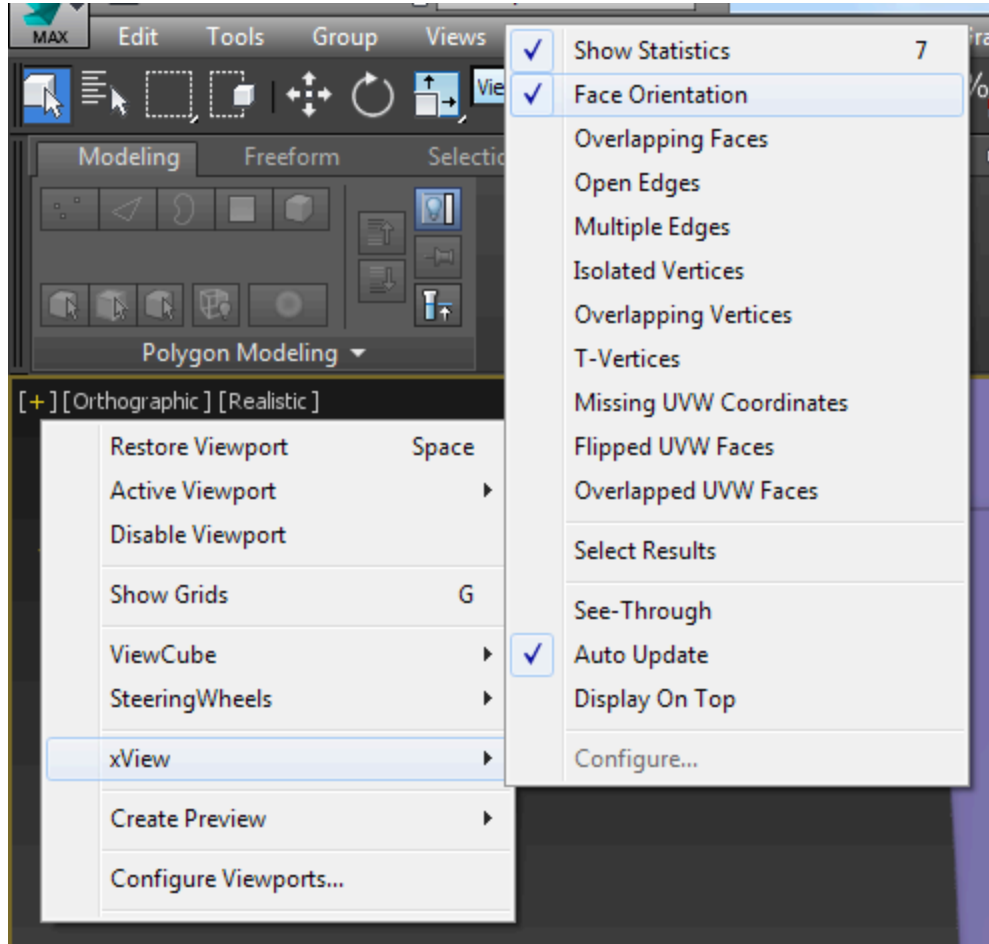
In 3ds Max, use the Reset Xforms utility then Collapse the Modifier Stack. In Maya use Freeze Transformation. In XSI use the Freeze button.

## Edge Padding

If a normal map doesn't have enough [Edge Padding](#), this will create shading seams on the UV borders.

## Normal direction

Before doing any bakes, make sure that both your HIGH POLY and LOW POLY have correct normal directions, i.e. no inverted faces. In 3ds Max you can check that by turning on Face Direction and selecting a mesh. Inverted faces will be colored green. If you have inverted faces you will get all kinds of projection issues, so be careful!

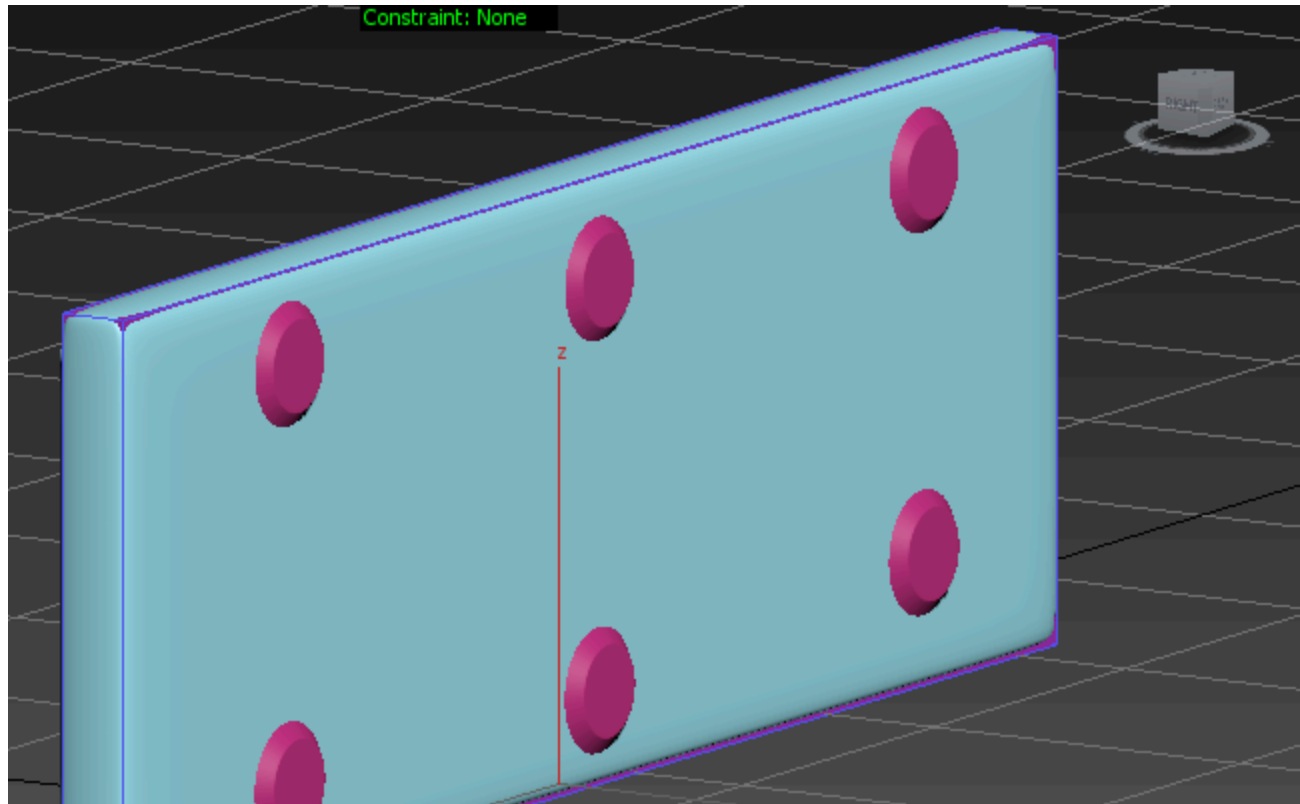


### ***1. The whole process***

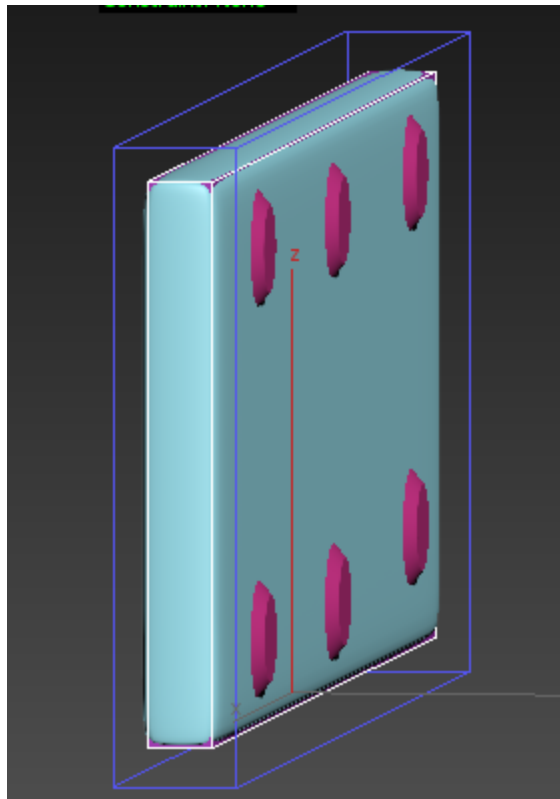
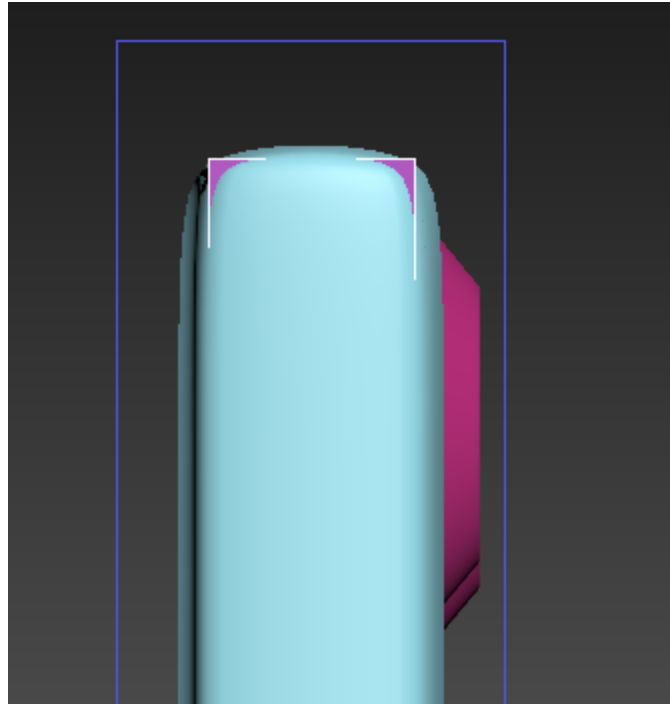
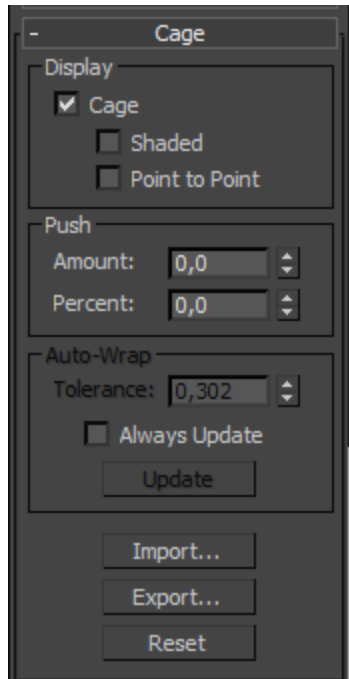
When baking a normal map, to take advantage of hard edges you will have to use a projection cage. There are a few ways to do that, I will show only 3ds Max and Xnormal ways to do that.

In 3ds Max:

Apply a Projection modifier to your low poly mesh



Here you can see how the cage completely covers the high poly mesh AND the floaters. It is very important that the cage covers everything, otherwise you will get projection errors.

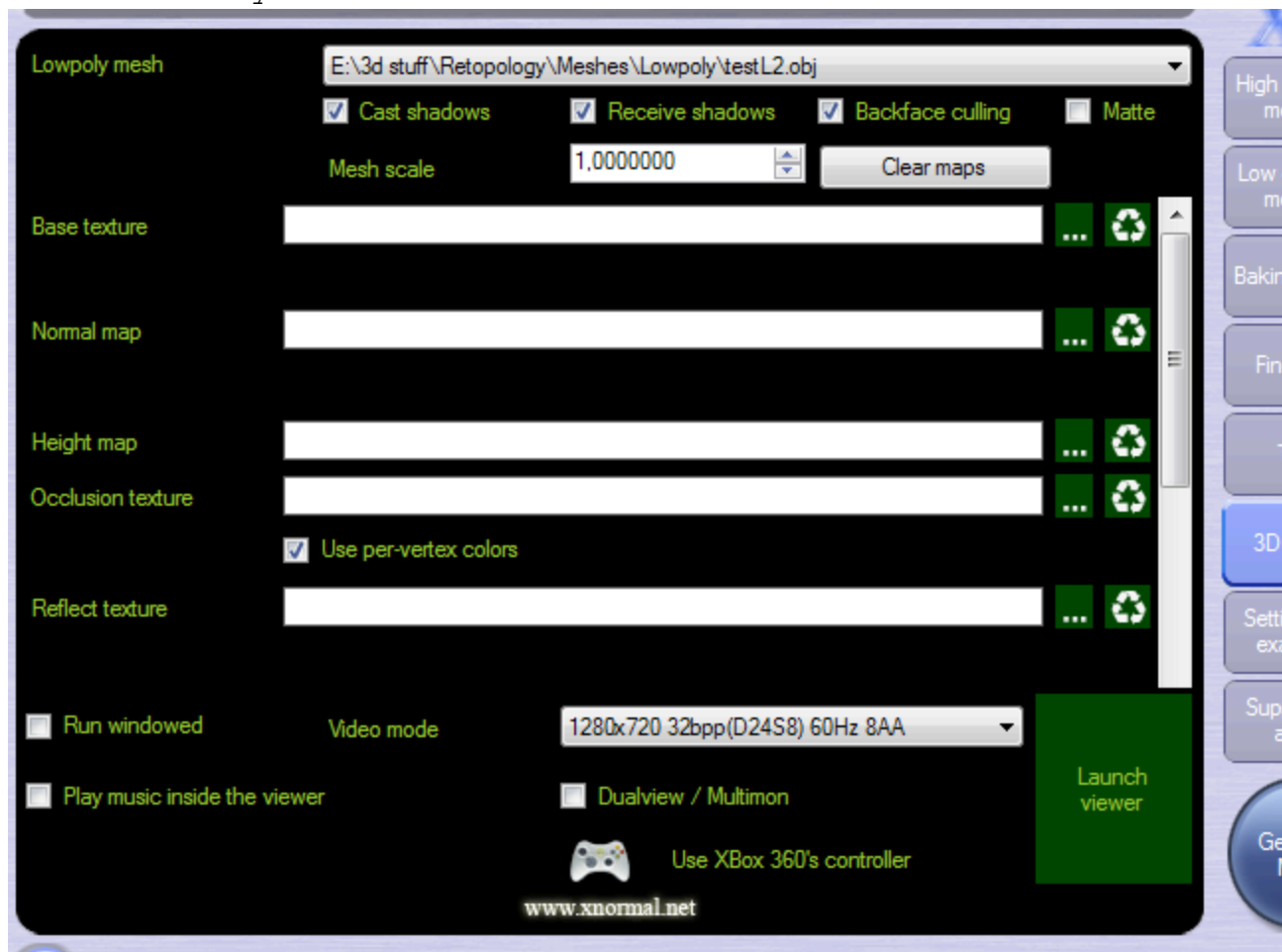


In Push many adjust the Amount to push the cage outwards from your mesh. The Amount = the distance. You can also adjust the cage manually, by entering the sub-object

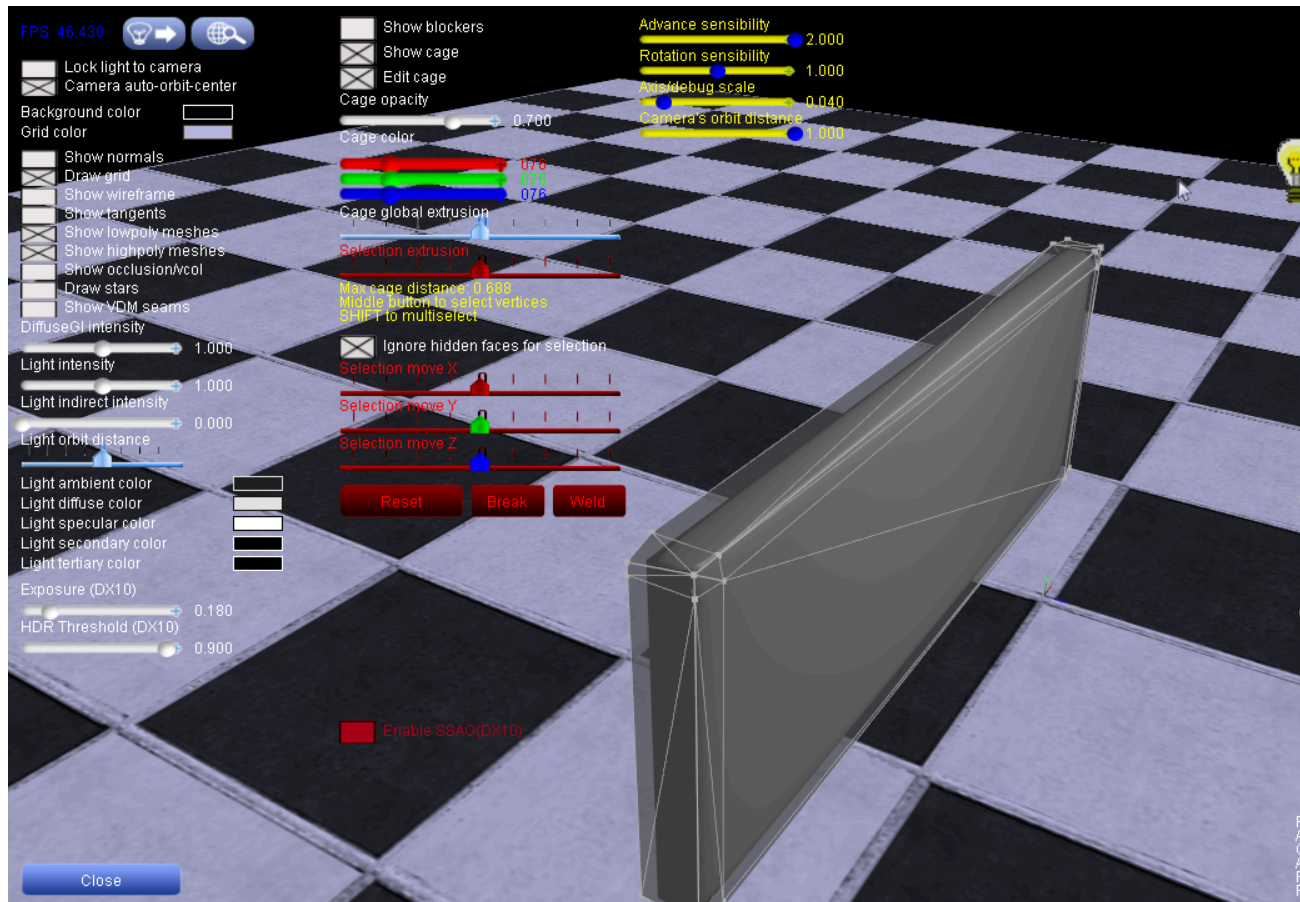
mode for Projection modifier. In 3ds Max, cage affects distance of the projection and the direction, so moving vertices by hand can be tricky and non-reliable.

Xnormal:

You can create a cage for your mesh directly in Xnormal. For that you'll need to enter 3d Viewer



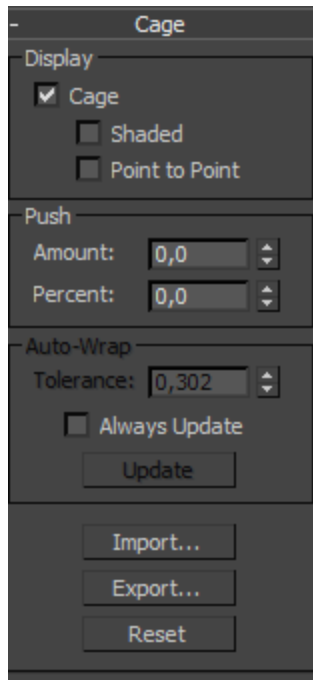




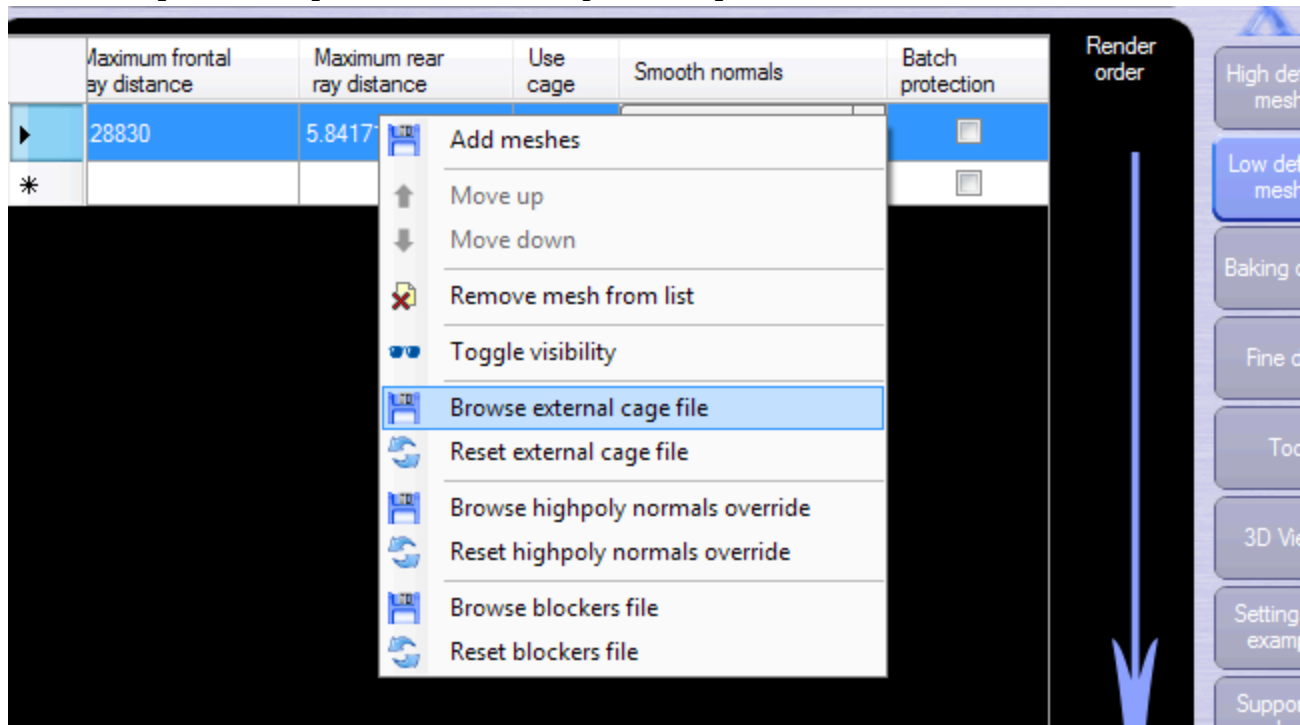
Check Edit Mesh box and edit the cage. Save meshes when done.

But it's not an ideal way to make cages, because you have to switch applications and the controls in 3d Viewer can be tricky.

Instead, you can import cage from 3ds Max.



Press Export and it will create a separate Cage mesh for you to export. Export it as .obj and open Xnormal.



Click with RMB on your low poly mesh and select Browse external cage file, choose your exported cage object from 3ds Max.

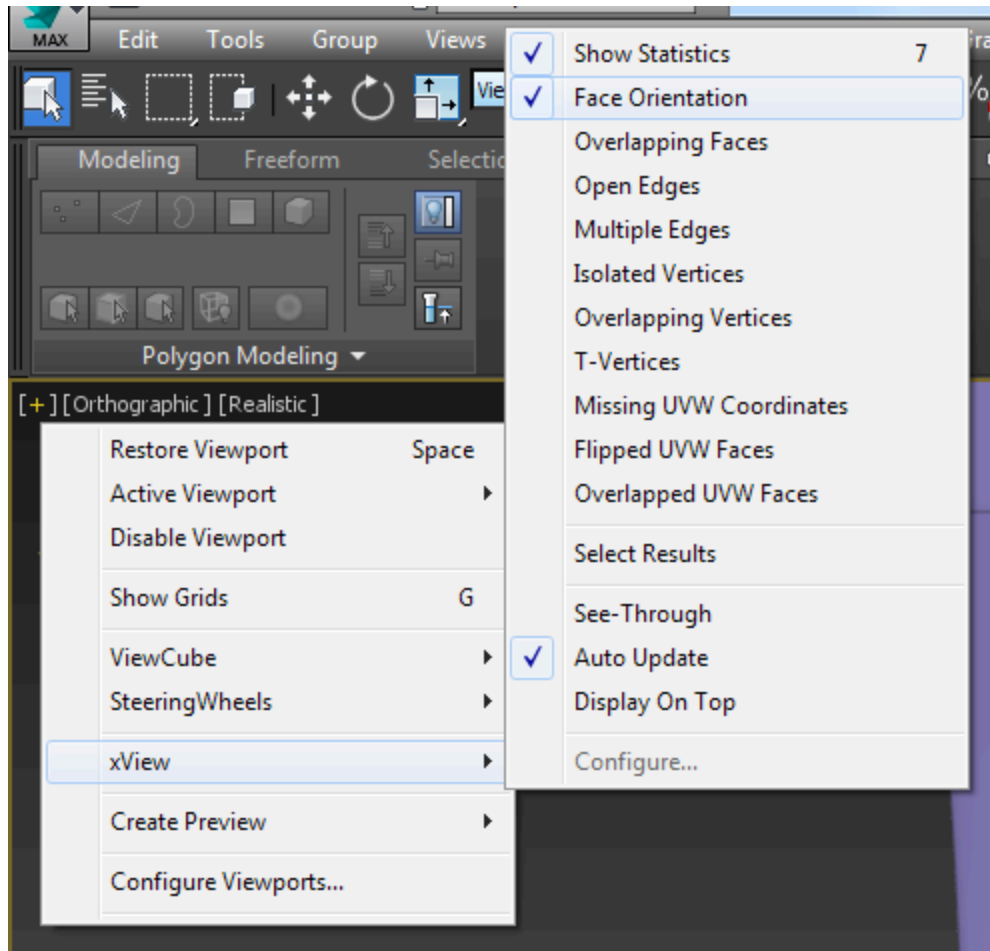
# ***Troubleshooting***

Even with an in-depth guide in your hand, it's very easy to make mistakes when it comes to baking normal maps. In this section I will show you how to prevent many of them and how to solve common and uncommon problems.

## **Be prepared**

If you have encountered a problem that you don't know a solution for, don't panic. Take these simple steps to ensure you properly prepared your mesh for baking. I will demonstrate that using 3ds Max, but these steps can be used in other modeling applications.

1. Make sure normals on your low poly and high poly meshes face in the right direction and aren't inverted. There are many ways to do that.
2. Make sure your low poly mesh has no topology issues. Check the mesh for any possible mistakes - unwelded vertices, overlapping faces, inverted faces, isolated faces, ngons etc. In 3ds Max you can apply STL Check modifier to quickly check for common problems and there's Xview feature in 3ds Max that helps to see various problems.



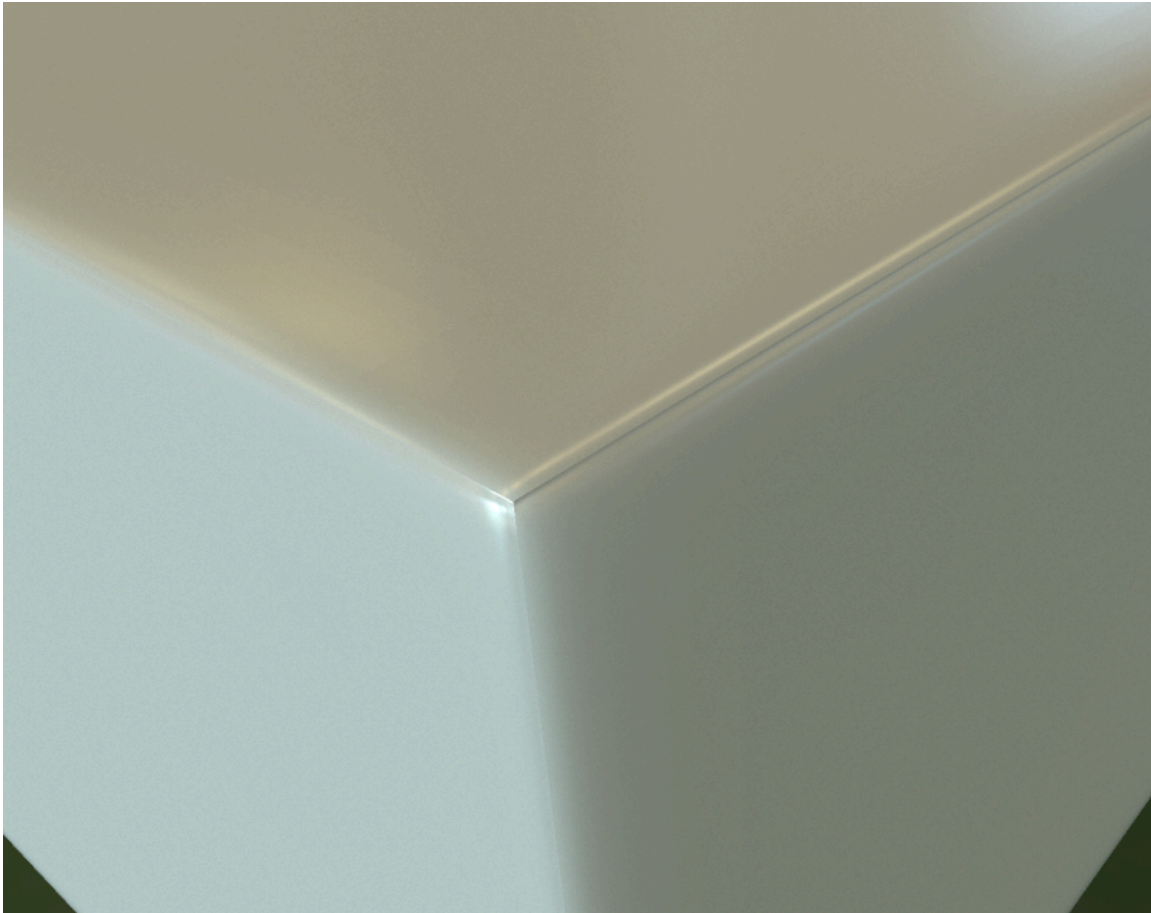
3. Collapse modifiers in your mesh (delete history in Maya)
4. Make sure your mirrored/overlapped UVs are offset to the near 1-0 space
5. Do Reset Transforms
6. Apply smoothing groups again
7. Triangulate
8. Create cage
9. Export low poly mesh as .obj or .fbx with Smoothing Groups, Normals, Tangents(option for .fbx) and Texture Coordinates

That's pretty much what you need to do to ensure your mesh

is 100% ready to be baked.

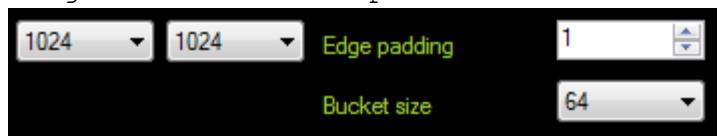
*Let's take a look at some specific problems when it comes to normal maps.*

1. I did everything right, but I can still see the seams on my normal map!



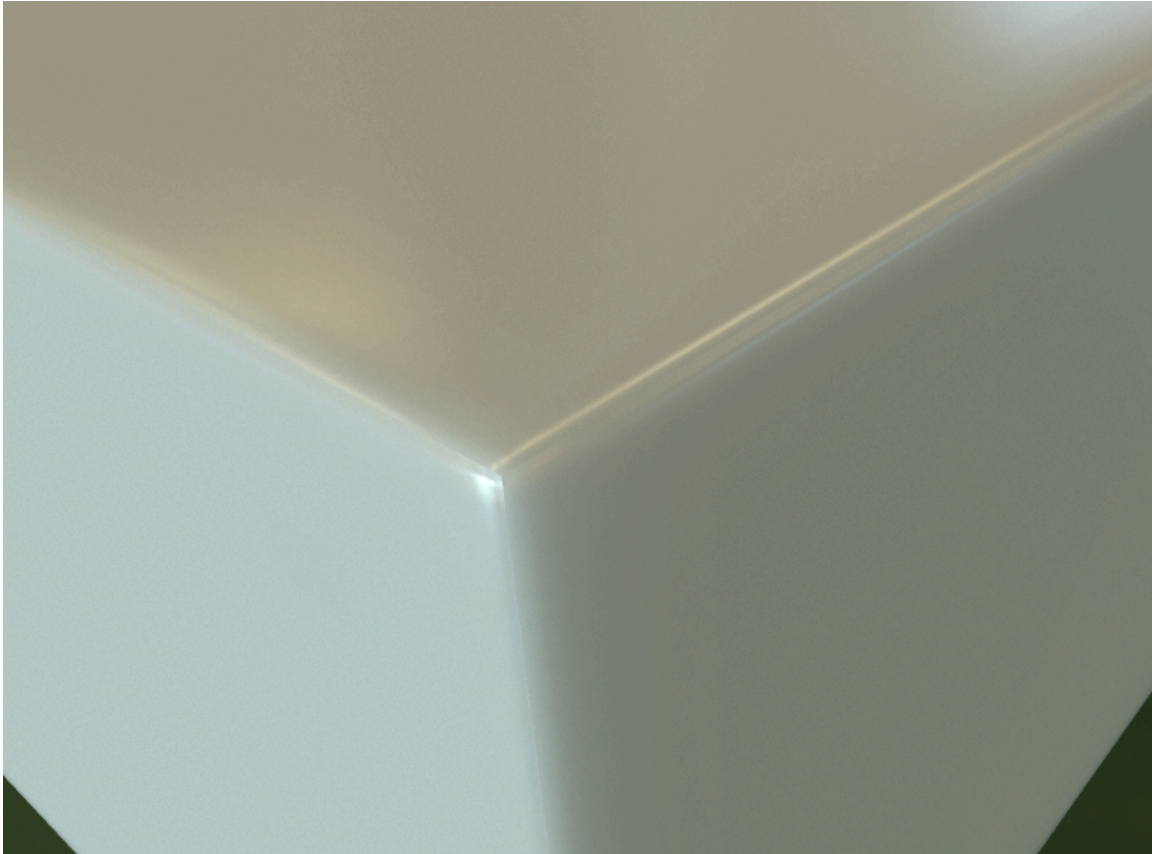
This is a perfectly baked mesh, except one little detail. When you have strange seams like that and you've done all the necessary steps correctly, you likely missed one important detail. Padding. This is a simple cube baked in

1x1k resolution, but



UV islands had only 1 pixel of edge padding in bakes. Try

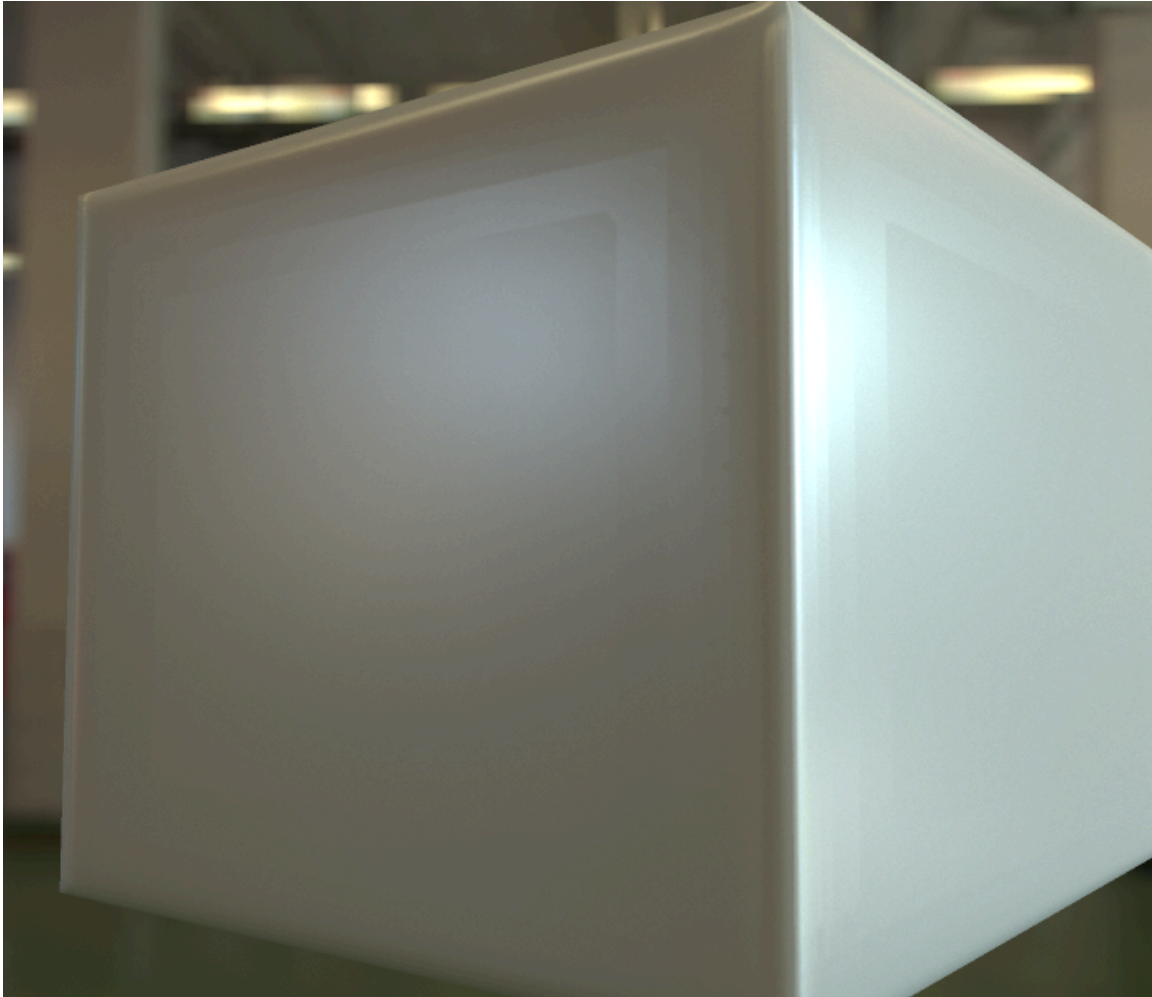
to change padding to 8-16 pixels, depending on the desired texture resolution and amount of Anisotropic Filtering in your engine.



Rebaked with 8 pixels of padding.

- 2. I have weird noise on the surface of my meshes after I applied a normal map.**





The reason you see this problem is due to lack of bit-depth precision. It's not something you have to think about most of the time, because it can be easily avoided. There are two typical ways that renderers deal with this:

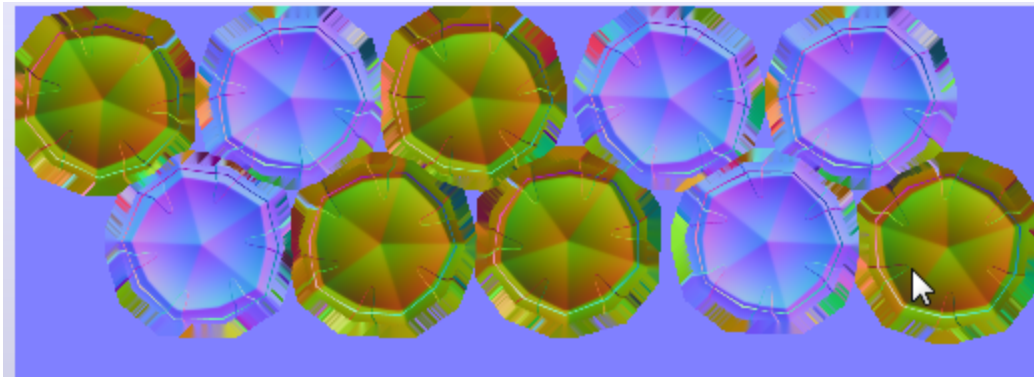
1. Adding noise or dithering, this removes the artifact that you see, but it adds noise to the texture. This is what max does and is why you don't see the same problem
2. No dithering but then you get this sort of stair-stepping artifact, this is what Maya and Xnormal do.

Do test bakes in .tga and if you encounter this issue then do the next step:

Bake your normal map in a 16 or 32 bit per channel format(.tiff for example), then import in Photoshop and

export as .psd or other 8bit formats like .tga. It will help to get rid of this issue.

### 3. My bake is yellow, should be blue! What am I supposed to do?



If you've done all the necessary steps from "Be Prepared" section you would never get this issues in the first place. But in case it slipped your mind, this problem appears when you have inverted normal direction in your high poly, fix it and rebake.

## Painting:

### ***1. Fixing normals by hand, do or don't***

Sometimes you encounter waviness in your normal maps, projection errors or gradients and you don't want to fix your low poly or create a proper cage. There is an option of fixing your mistakes in Photoshop by smudging and cleaning the normal map. But, ultimately, it's a destructive workflow. What if someone asks you to change your mesh and you'll have to do rebakes? Are you going to fix it all by hand again? If you prepare and work smart with your high poly and low poly, then you won't have to do that extra work.



## ***2. How to add details to normal map***

Okay, I baked a mesh, but there are no details. I need to add fabric definition to the normal map or other details, how do I do that?

There are a lot of ways to create normal map details. Simple way is to convert a height map to normal map with the help of Nvidia normal plugin in Photoshop or use other applications like Njob, Knald, Xnormal, NDO2, Substance Designer.

Height map is a greyscale map.





In converting height map to normal map you have to think about what values in this map mean. Everything that is darker than Middle Gray is treated like indents and everything lighter as peaks. You don't have to use Middle Gray at all, but if you don't then you can't separate indents from peaks easily.

Gray = middle layer

Using this technique you can convert any grayscale image to a normal map.

If you want to overlay your generated normals on top of baked normals, all you have to do is overlay it smartly

on top.

You can do it manually in Photoshop using this method:  
<http://vimeo.com/95516153>.

Crazybump also provides a feature to overlay normals.

You can also try this handy Photoshop script, but it might not work on your version of Photoshop:  
<http://www.polycount.com/forum/showthread.php?t=131819>

### ***3. Software for painting normals, 2d and 3d***

With the help of Quixel NDO2 and Substance Painter, generating and painting normal details became a breeze. NDO2 is a handy plugin for Photoshop that allows you to paint directly on the normal map and quickly generate normals using everything Photoshop can offer.

<http://quixel.se/dev/ndo>

Substance Painter is a 3d painting application that allows you to paint directly on your low poly mesh and affect all kinds of things from normals to diffuse texture at the same time.

<http://www.allegorithmic.com/products/substance-painter>