

Blender Bros

THE HARD-SURFACE E-BOOK

Created by Josh Gambrell and Ponte Ryuurui (aka BlenderBros)

Introduction

3D Modeling is not hard, unless it's **hard** surface. This was my first clever pun for the e-book. Give me a break will ya?

But seriously, hard surface is not hard either. At all. Unless we take the term literally.

This e-book will teach you everything you need to know to get started.

WHERE TO START?

This e-book is going to start with the basics, so if you are already familiar with the basics, feel free to skip ahead, or simply use this e-book as a reference guide when needed.

This e-book is going to assume some familiarity with Blender, and instead is meant to focus on fundamentals rather than teaching Blender from scratch. There are plenty of YouTube tutorials for that.

Table of Contents

The Default Cube 3

Auto Smooth 6

Auto Smooth Angles 8

Bevel Limit Methods - Angle 11

Bevel Limit Methods - Weight 14

Booleans 16

Normals – What are they? 20

Booleans (continued) 25

Booleans – curved surfaces 32

Ngons vs. Quads – The REAL answer 43

Non-Destructive vs. Destructive Modeling 46

Boolean Cleanup Strategies 51

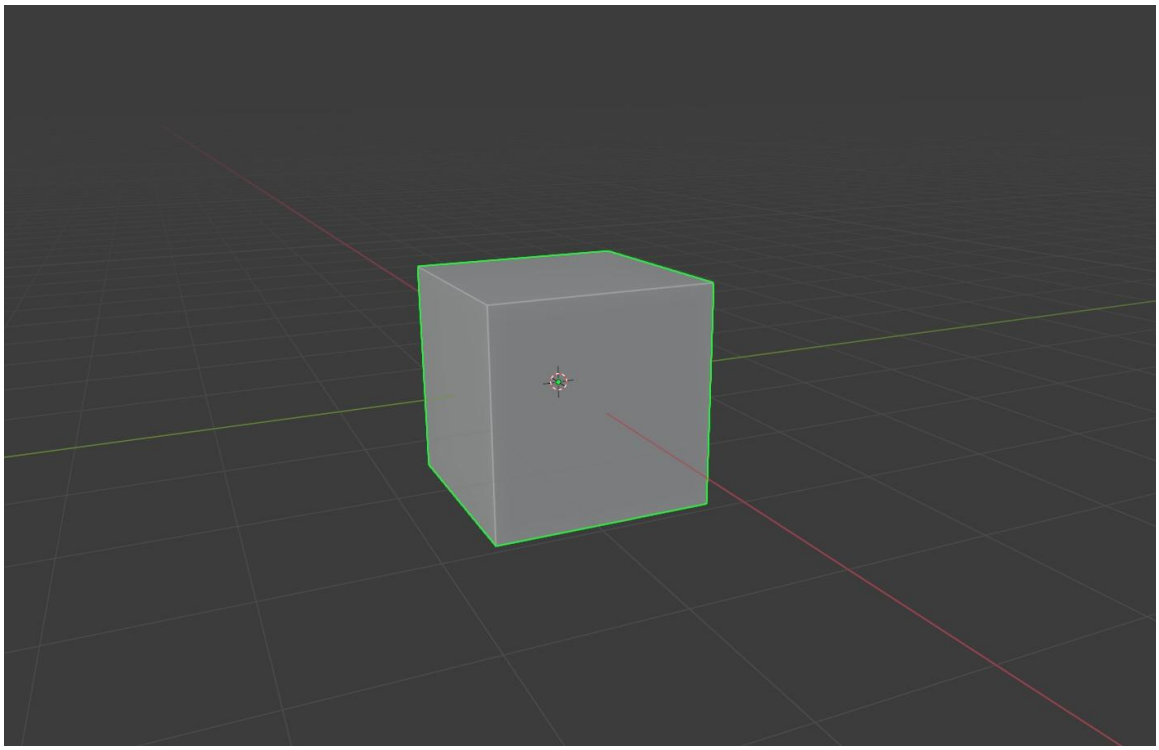
Multi-Level Bevels 59

Dicing Booleans 68

Subd with Booleans 73

Conclusion 84

The Default Cube

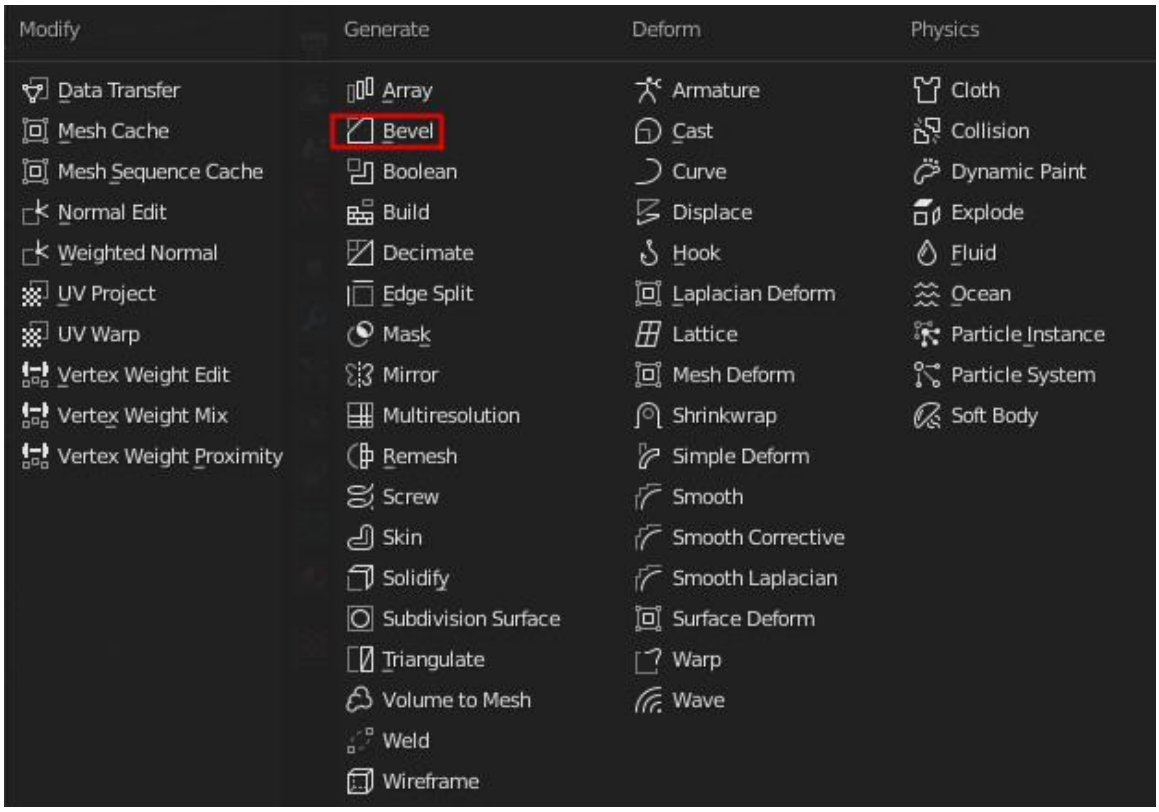


“Who are you and what do you want from me?” said the default cube, immediately before being deleted from the scene. Poor guy. He never survives more than a few seconds.

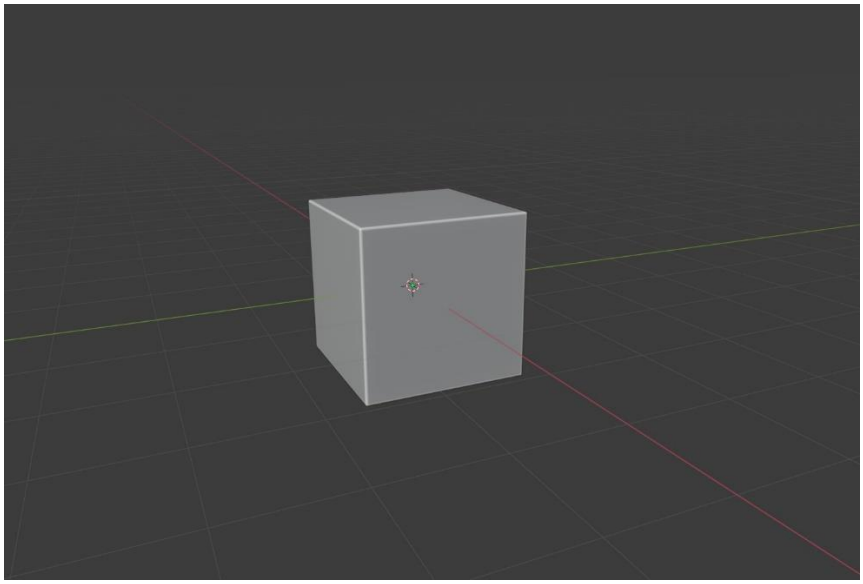
But for the sake of this part of the book, we will spare him.

In Blender, the default cube is the first object present in your scene. But most of you know that.

In Blender, we have a modifier called the **Bevel Modifier**.



What happens when we add this modifier to the default cube?



Notice the nice highlights around the edges?

This is a natural effect that occurs in real life.

Nothing is truly sharp. Even the edge of a knife blade has a microscopic bevel on it. It just looks sharp because that is all we can see with the naked eye.

Every object should have bevels, no matter how big or small.

Bevels simulate that nice, rounded edge effect, but they also reflect light very nicely. It is especially noticeable in metal materials.

It might not seem like much, but the differences it will make in your scenes are massive. Bevels are the basis to hard-surface modeling. They are what will separate an amateur artist from a professional one, or a beginner from a more experienced one.

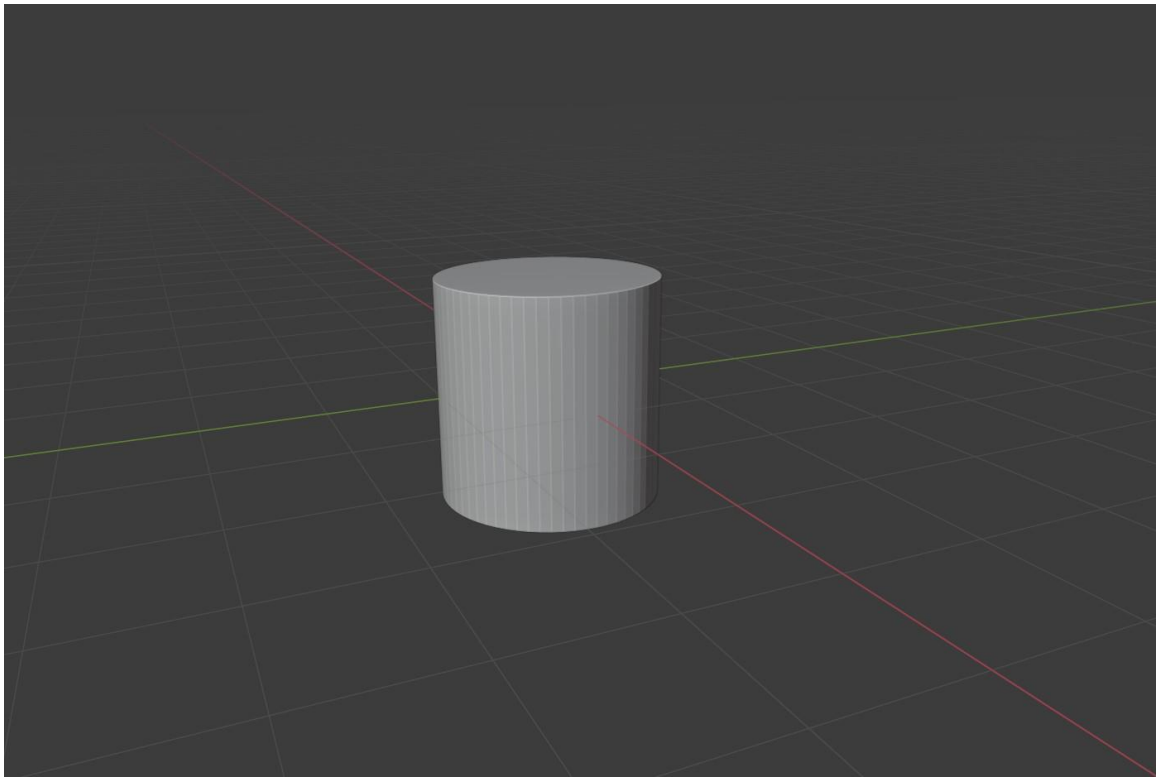
Bottom line: use bevels. Always.

Auto Smooth

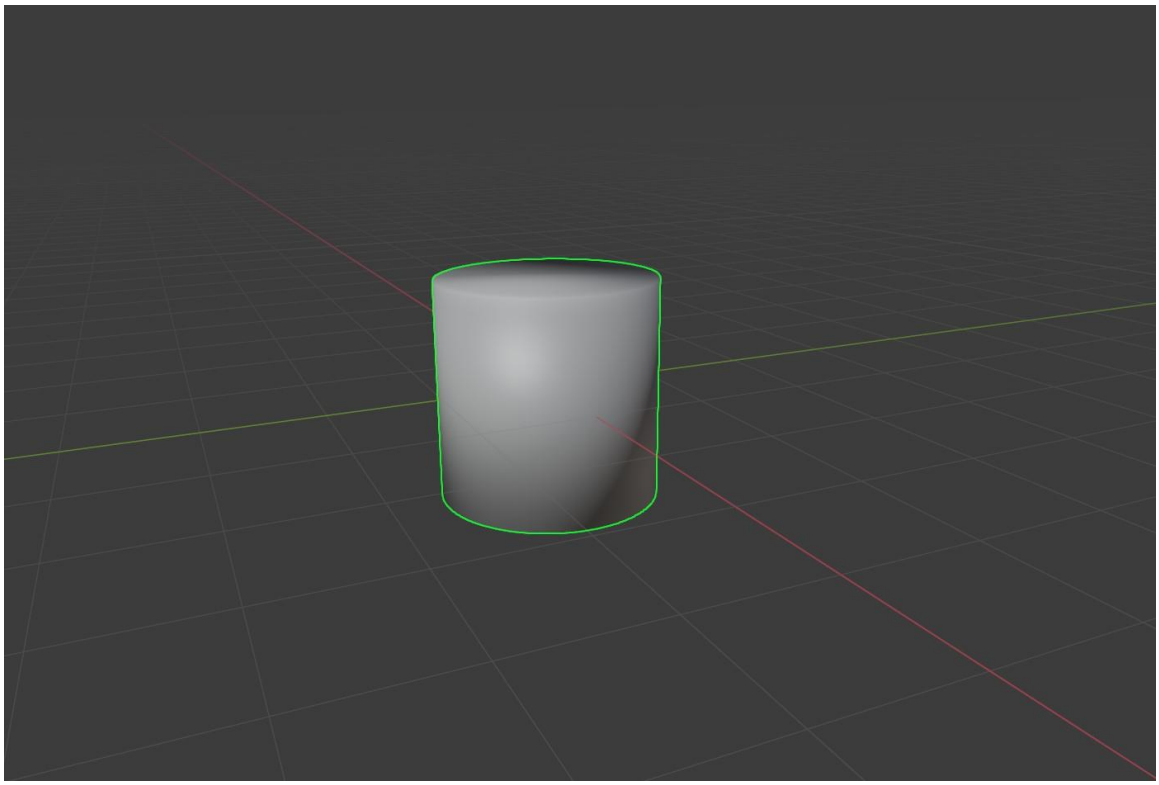
Auto smooth is easy to understand. I don't know why people like to over complicate it.

Sometimes you need to smooth out an object.

For example, the default cylinder has flat shading by default:



If we smooth it out, this happens:



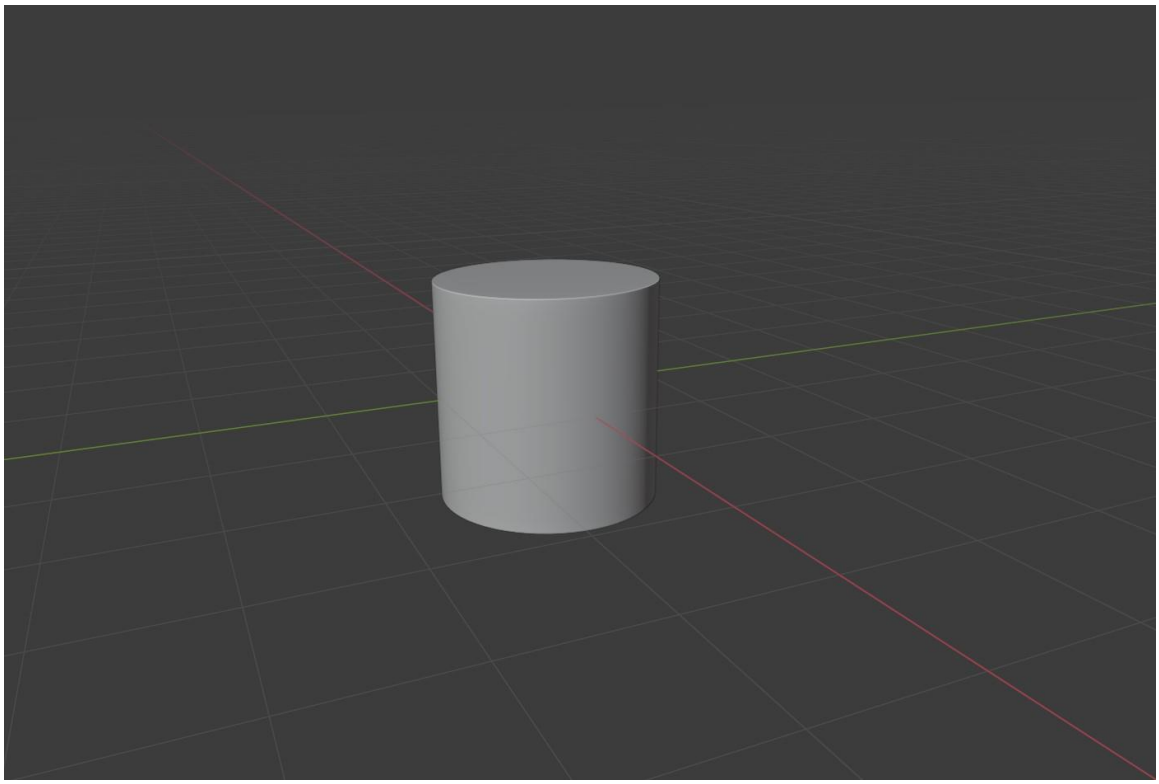
Why does it look so... weird?

This is because the smoothing algorithm in Blender smooths out over *everything*, even over harsh edges.

Spheres and organic objects smooth so nicely because these have nice, continuous surfaces with no harsh edges on it.

In hard surface, this is different. We don't want to smooth out over the hard-angled edges. We want to tell Blender to smooth out the geometry *separately* based off of the angles of the edges.

Auto Smooth Angles



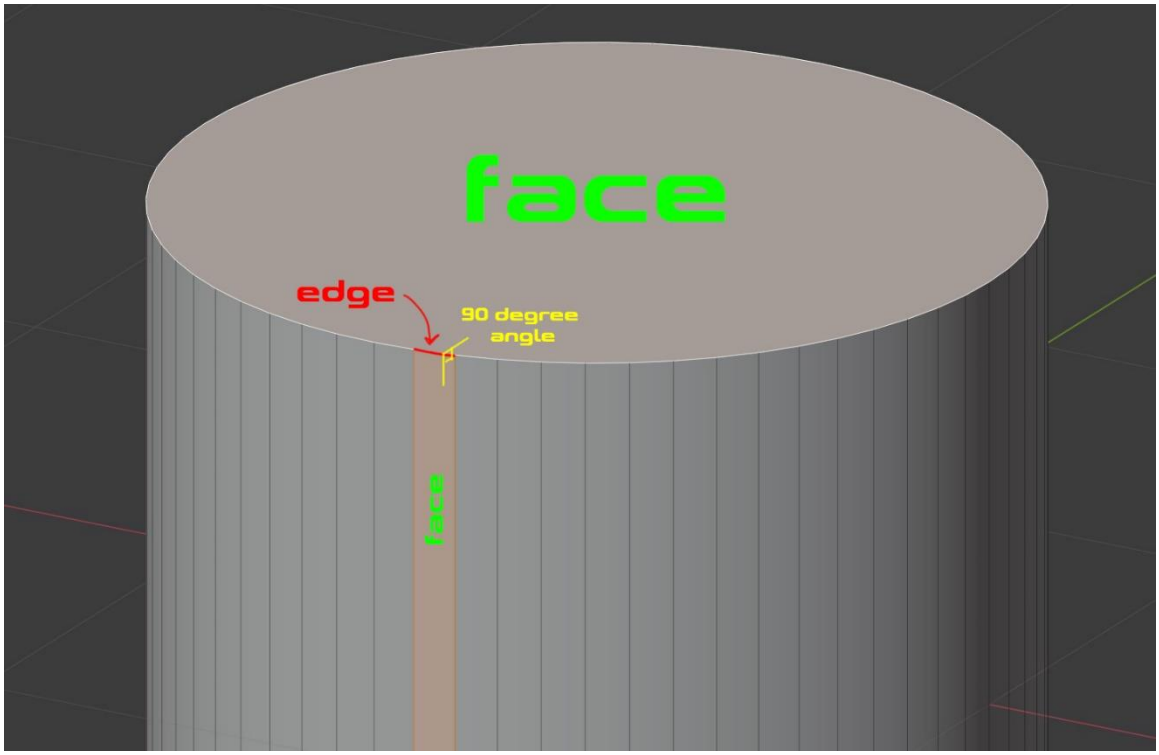
The image above is the result we'd like to achieve.

It smooths out very nicely, with no weird shading problems.

This is where auto smooth angles come in handy.

We can simply tell Blender "hey Blender, if the angle between two faces is *below* this value, then don't smooth out over that edge."

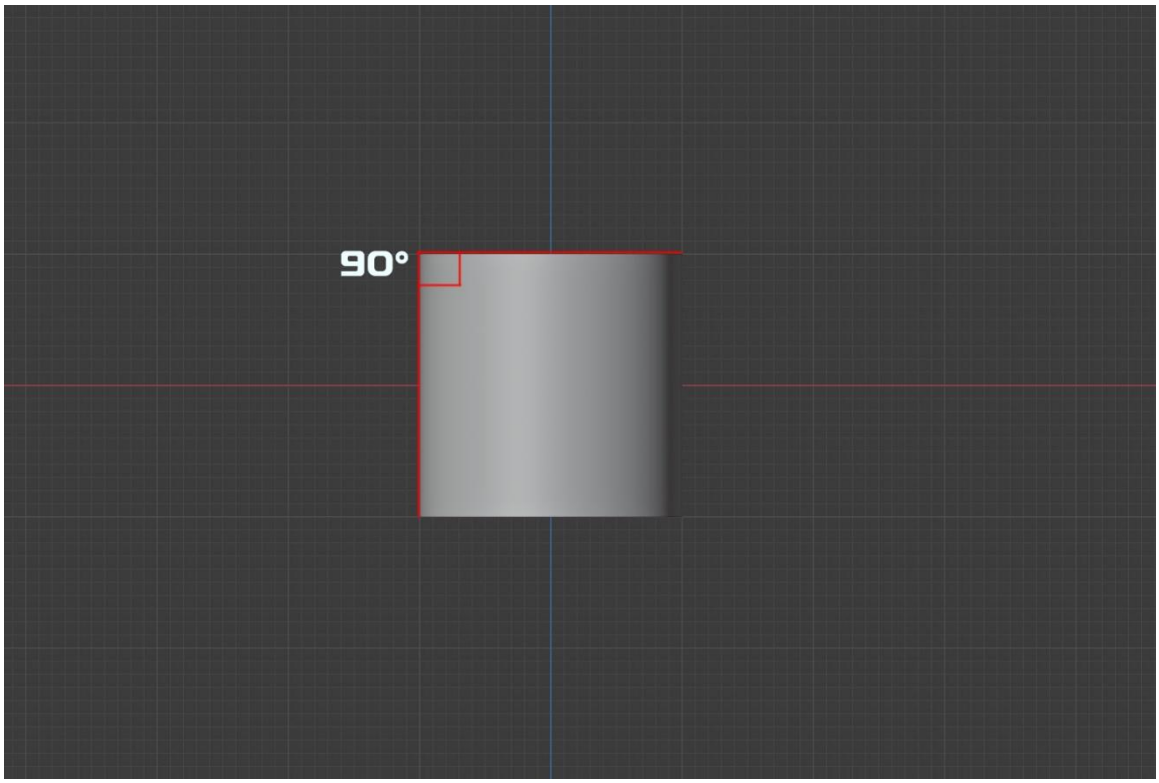
By the way, two faces are connected via an edge, so we aren't necessarily concerned with the "angle" of an edge. That isn't really a thing in 3 dimensions. We are technically talking about the angle between two faces.



Simple enough? If the last paragraph was confusing, just ignore it.

In the image above, the angle of the top set of edges is a clear 90-degree angle.

Here is a view of the cylinder from the side – the angle is more apparent this way:

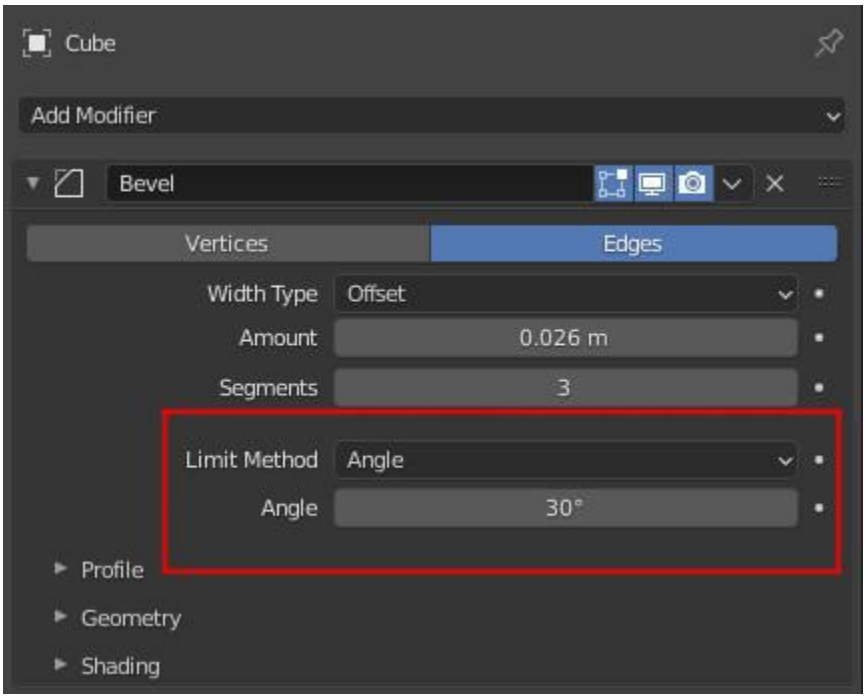


So, to avoid those weird shading issues, we need to tell Blender "hey Blender, if the angle between two faces is at 90 degrees or *above*, smooth *everything* out (bad shading in this case!).

If the angle between two faces is *below* 90 degrees, don't smooth over those edges. Smooth them separately".

This is how we get that nice shading in the pictures above.

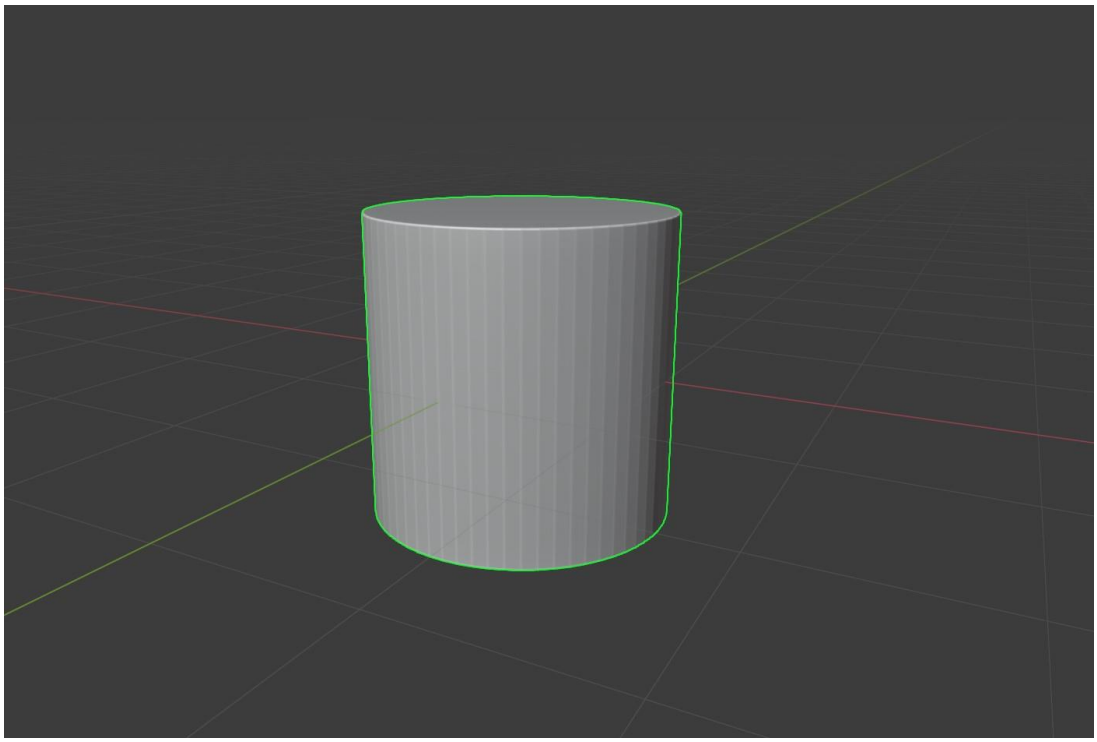
Bevel Limit Methods - Angle



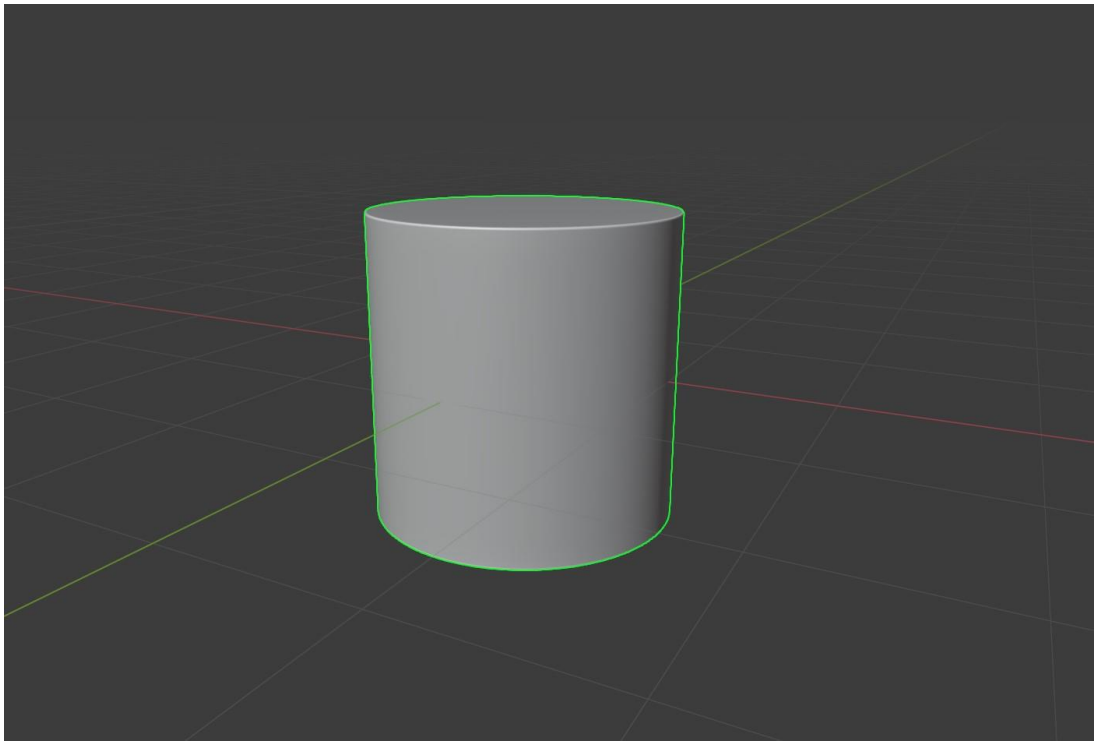
In the bevel modifier, you'll have something called a "Limit Method".

By default, it's set to "None", meaning *every single edge* will be beveled. Rarely do we want this. We want to bevel only the harsher edges, like shown above!

Here is the difference between no limit method and the limit method set to Angle:



Limit Method set to "None"



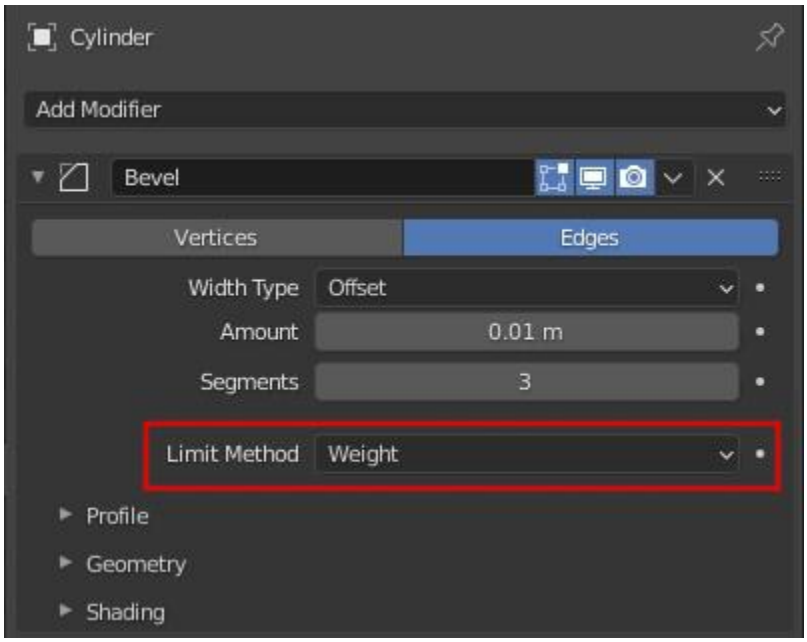
Limit Method set to "Angle"

In the above photos, the top one has every single edge being beveled, and the bottom one only bevels the top edge.

Which one looks nicer? I think you know the answer.

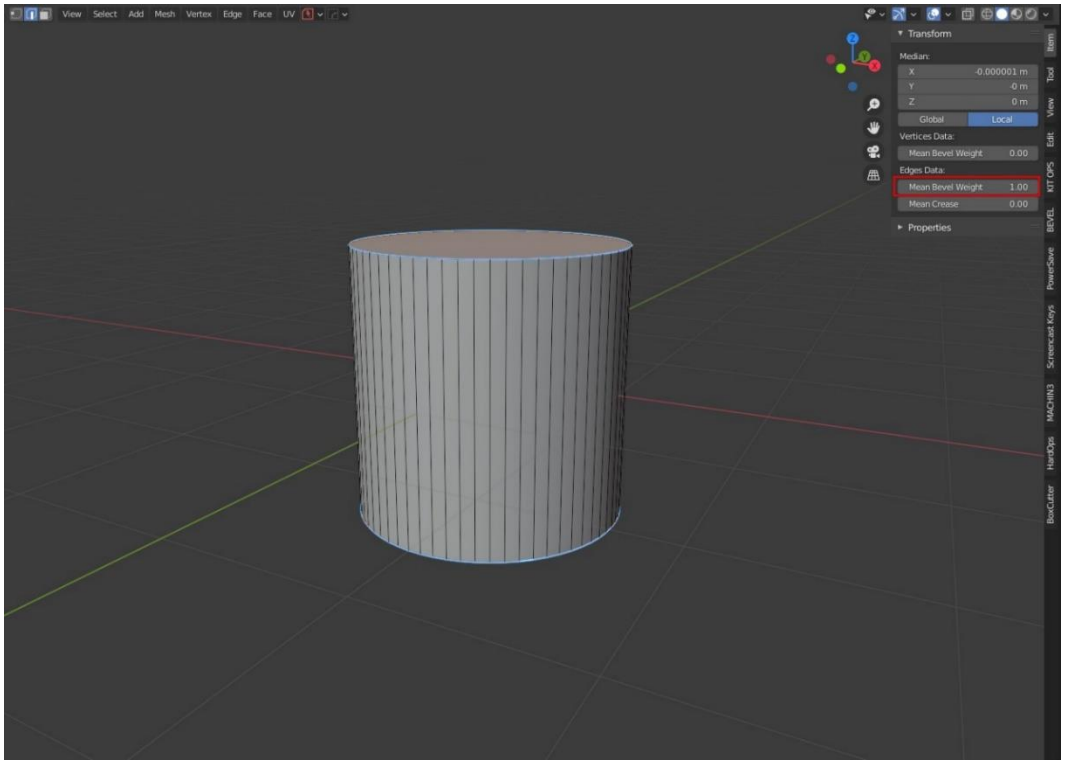
The “Angle” Limit Method in the bevel modifier works *exactly* like the auto smooth angles: it bevels edges based off of angles.

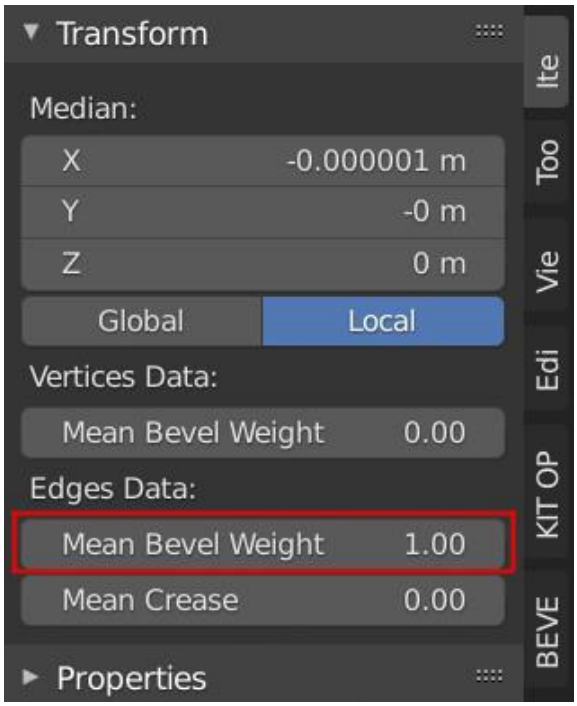
Bevel Limit Methods – Weight



Weight is a nice one because you have full control over where the bevels go.

All you have to do is select the edges you wish to bevel, and apply a mean bevel weight of 1:





Any selected edges will now have a mean bevel weight of 1, basically meaning those edges will be the ones getting beveled, so long as your limit method is set to "Weight".

This is useful because with the Angle Limit Method, sometimes angles will catch areas that you don't want to catch in order to bevel another portion of the object.

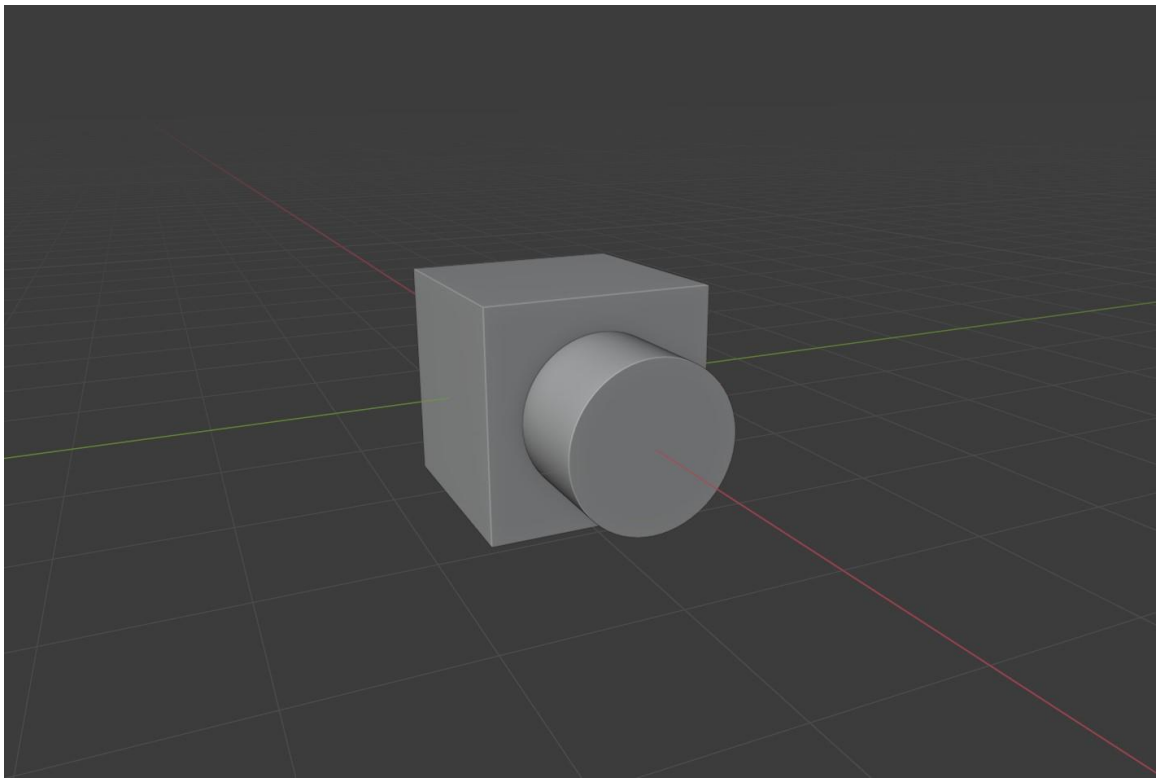
Angle Limit Method is not as versatile, but it is immediate (you don't have to make any manual selections).

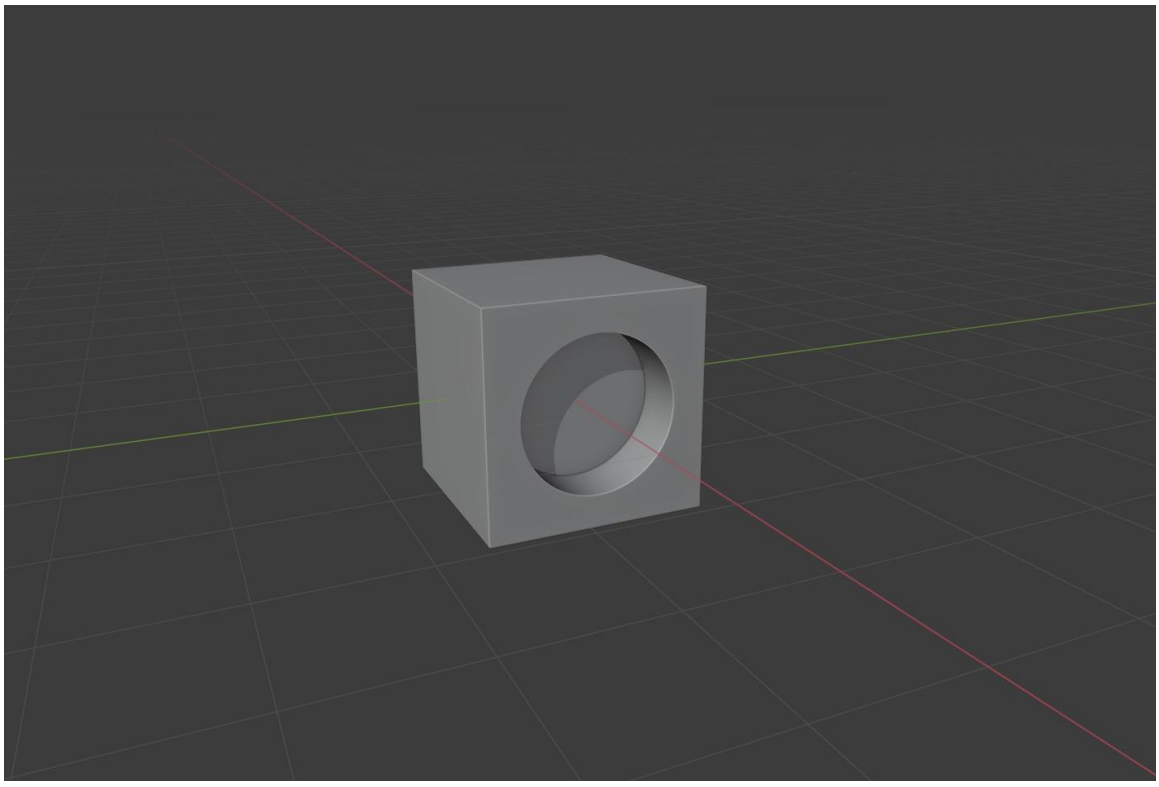
Most of the time, you'll be using the Angle Limit Method, but sometimes you'll encounter situations in which Weight is a better option.

Booleans

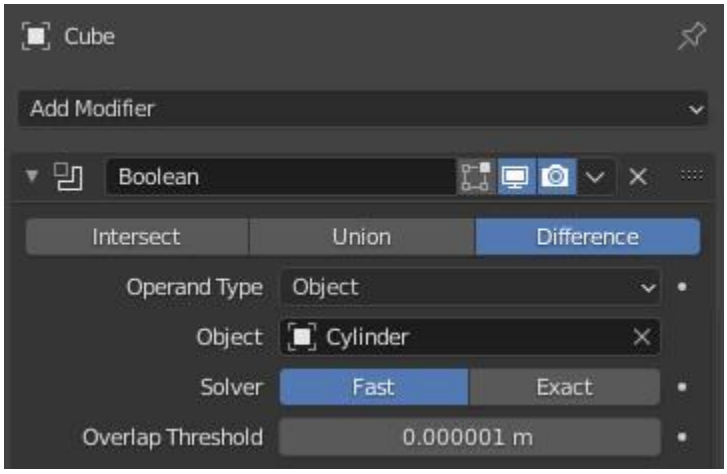
Booleans simply allow us to cut into objects based off another object.

For example, in the images below, I've used a cylinder to cut a hole into a cube:

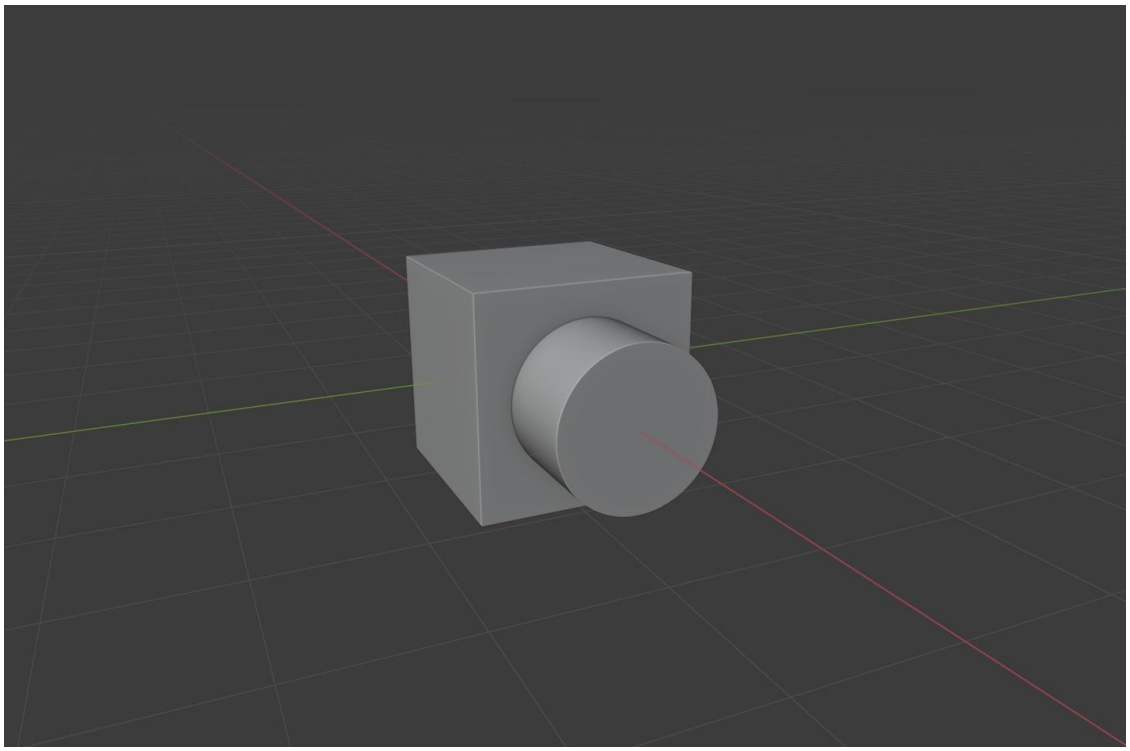




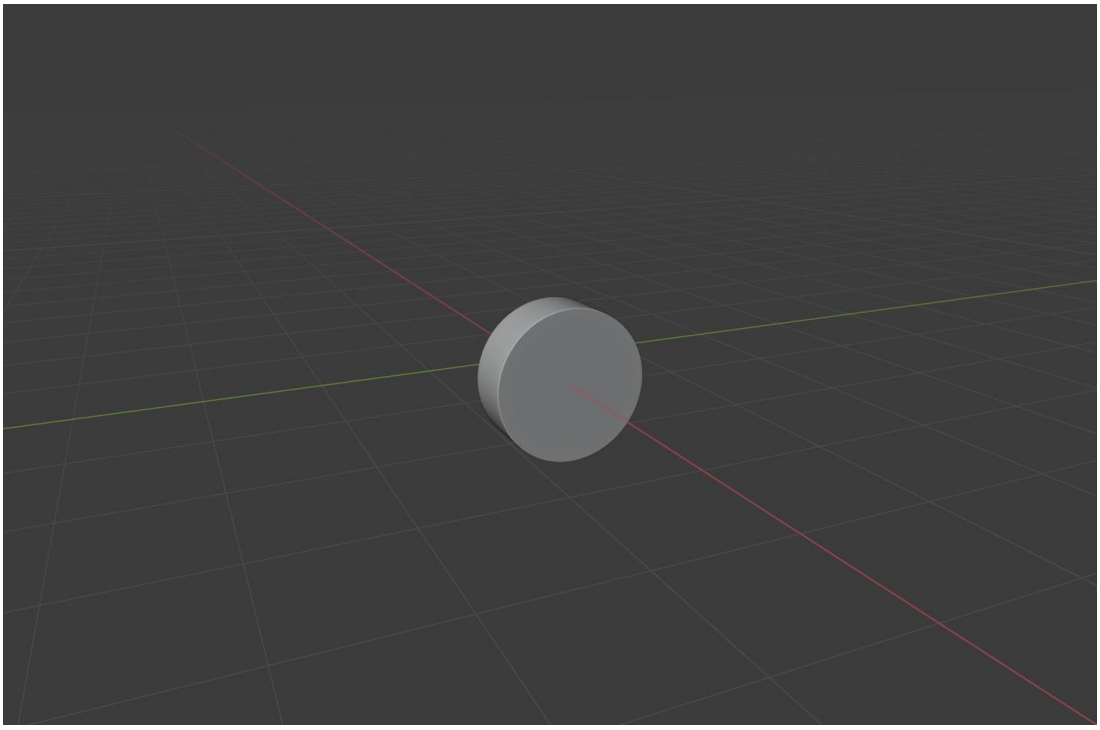
As many of you know, this is simply done using the “Difference” operation in the Boolean modifier:



Union will simply fuse the two objects together, and intersect will take the common areas between the two:



Union

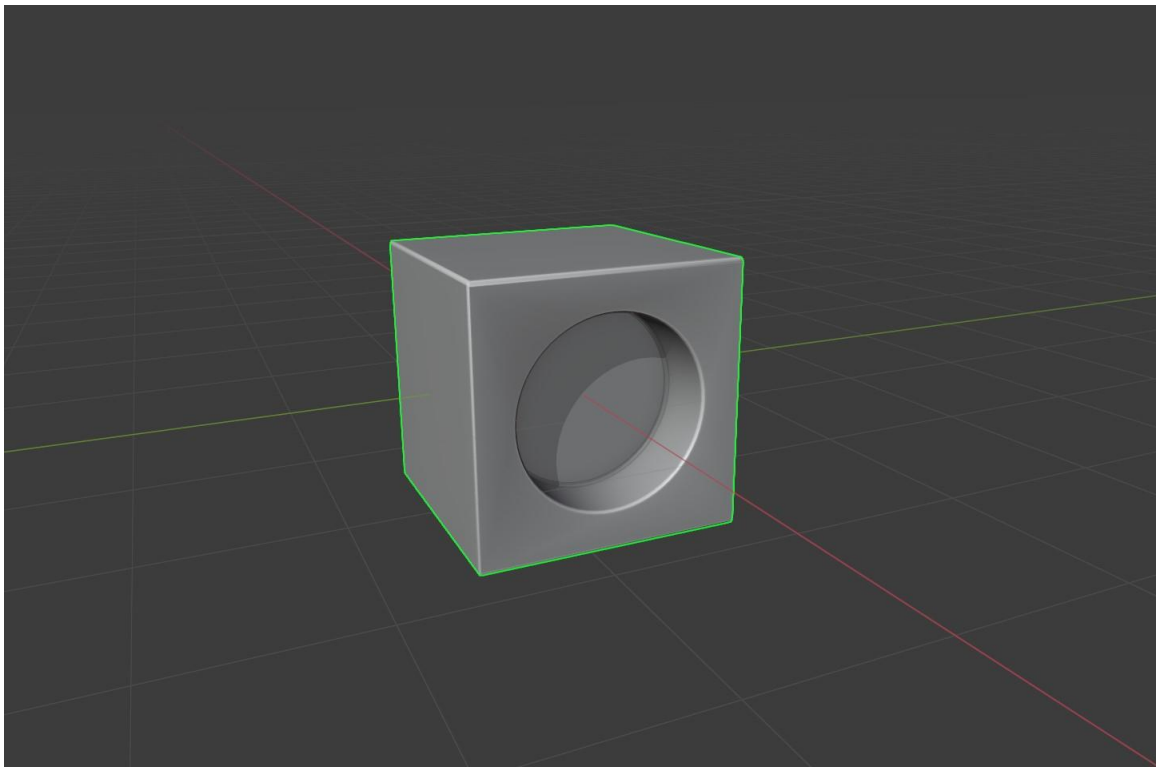


Intersect

I rarely use intersect. Sometimes it comes in handy, but not often.

I use difference Booleans probably 70% of the time, unions 29% of the time and intersect 1% of the time.

What happens if we add a bevel to this object? Well, as you probably would have guessed, it runs a bevel around the edges as usual.



The shading looks a bit off though.

First, make sure your auto smooth angle is appropriately set. 30 degrees is generally a good default, and in this case works fine.

But the shading is STILL off. Why is that?

Let's take a quick break and discuss normals so you can understand why this happens.

Normals – what are they?

If you've taken an elementary calculus course at any point, you are probably familiar with the idea of a tangent to a curve.

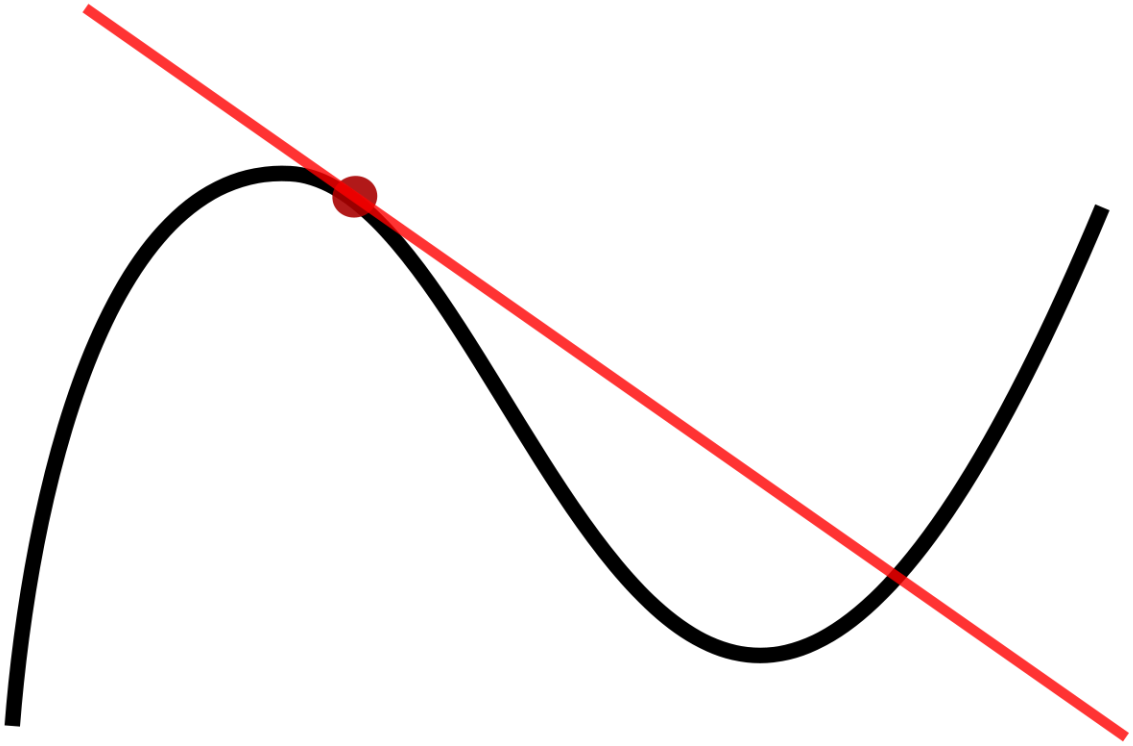
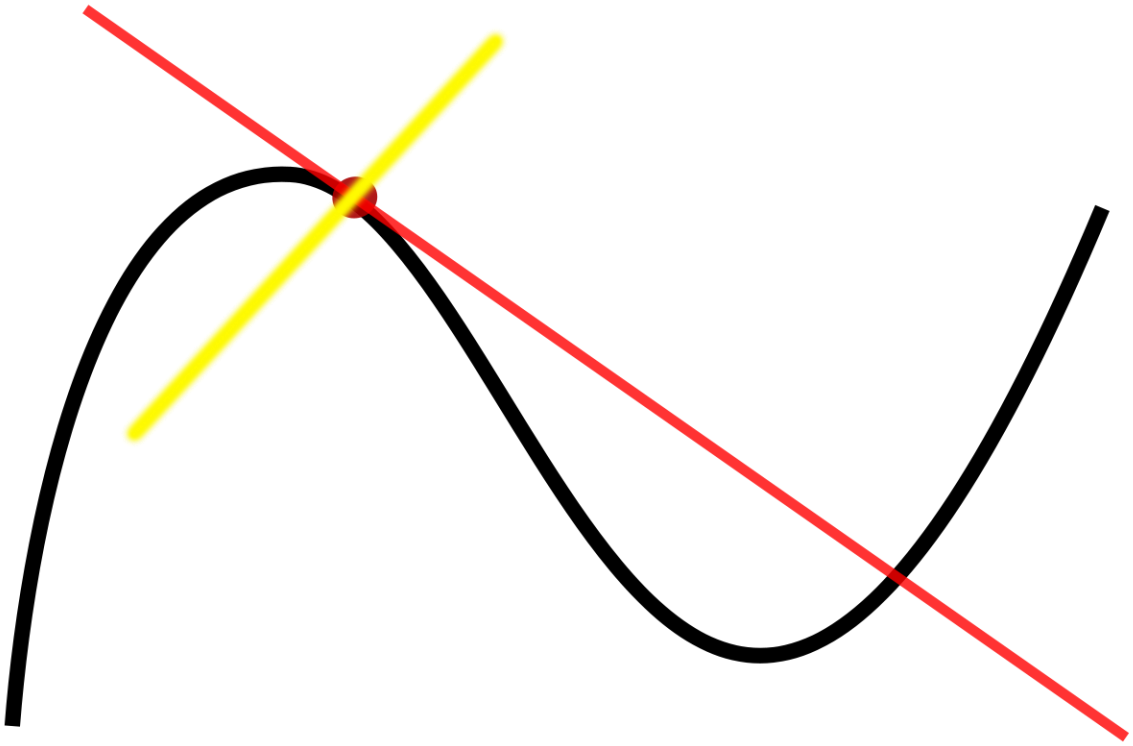


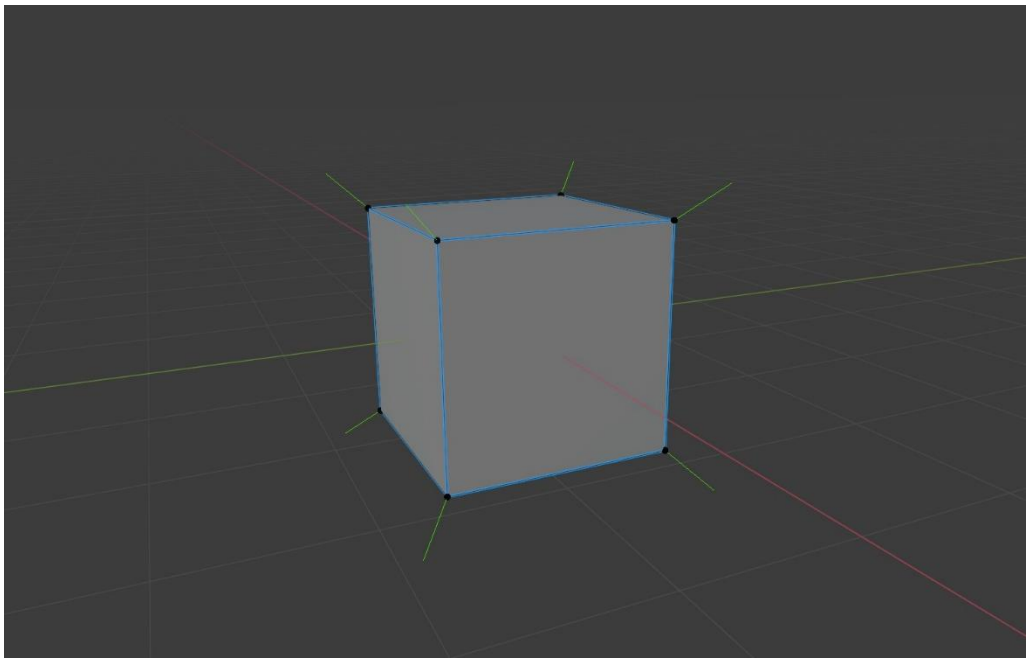
Image from Wikipedia

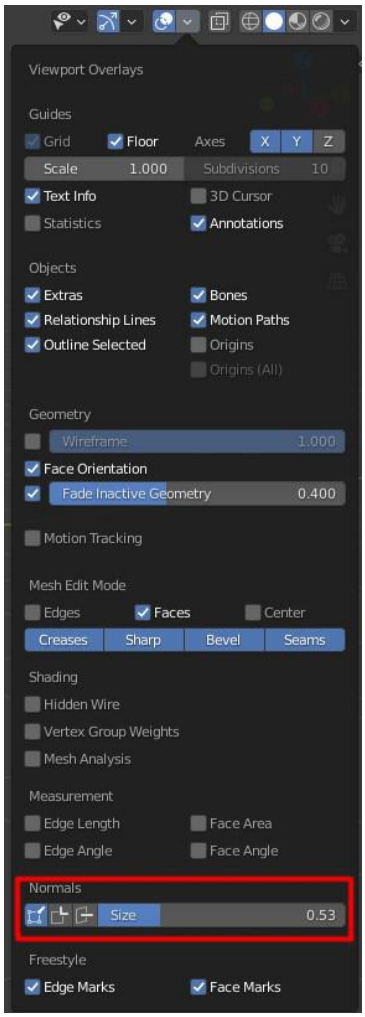
As you can see, this line intersects the top of the curve at a single point.

A **normal line** would be a line perpendicular to this point (as shown below in yellow):



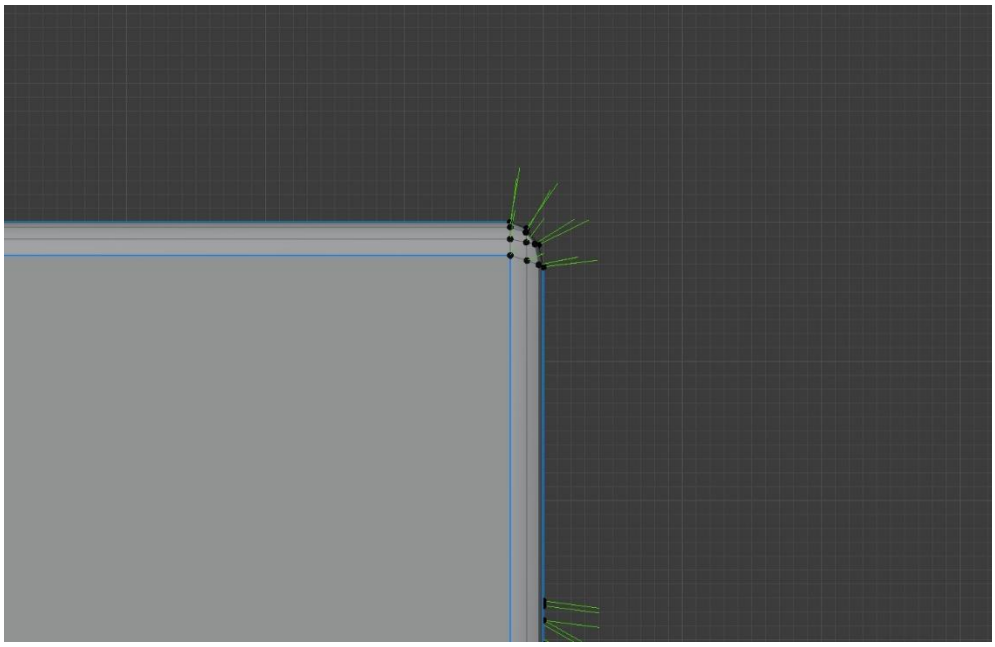
The same idea applies to 3D. A normal line would be any line *perpendicular* to a vertex. You can enable normal lines in the overlays panel in Blender, then view it in Edit mode:



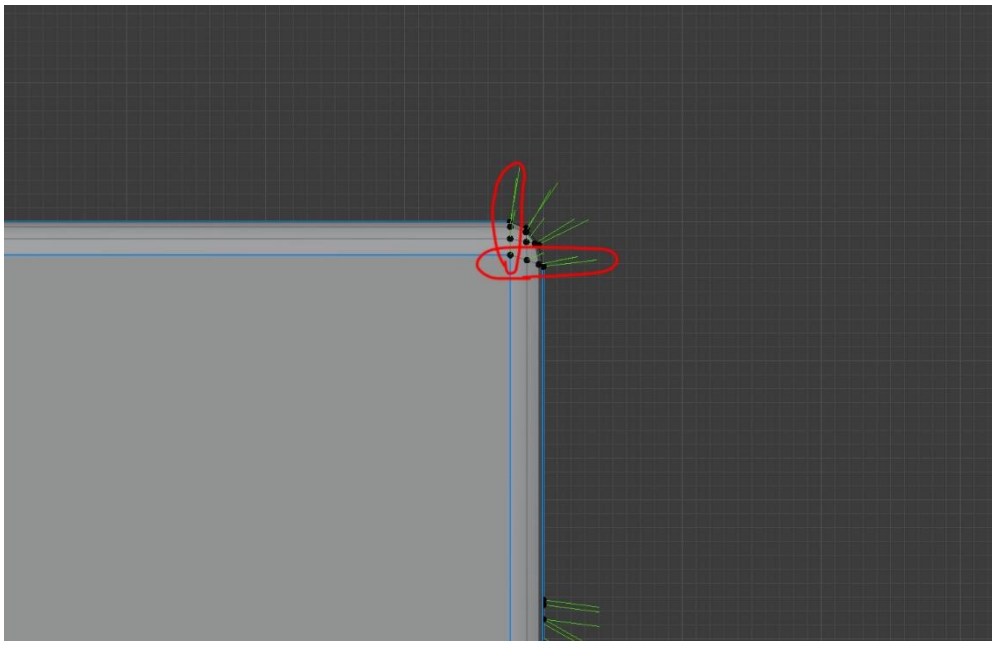

Viewport Overlays
Guides
Grid
Floor
Axes
X
Y
Z
Scale
1.000
Subdivisions
10
Text Info
3D Cursor
Statistics
Annotations
Objects
Extras
Bones
Relationship Lines
Motion Paths
Outline Selected
Origins
Origins (All)
Geometry
Wireframe
1.000
Face Orientation
Fade Inactive Geometry
0.400
Motion Tracking
Mesh Edit Mode
Edges
Faces
Center
Creases
Sharp
Bevel
Seams
Shading
Hidden Wire
Vertex Group Weights
Mesh Analysis
Measurement
Edge Length
Face Area
Edge Angle
Face Angle
Normals
Size
0.53
Freestyle
Edge Marks
Face Marks

Normal lines have a LOT to do with shading. The reason the shading goes bad on bevels is because a bevel has multiple *angled* normal lines.

If we apply the bevel modifier, we see what is *really* happening behind the scenes:



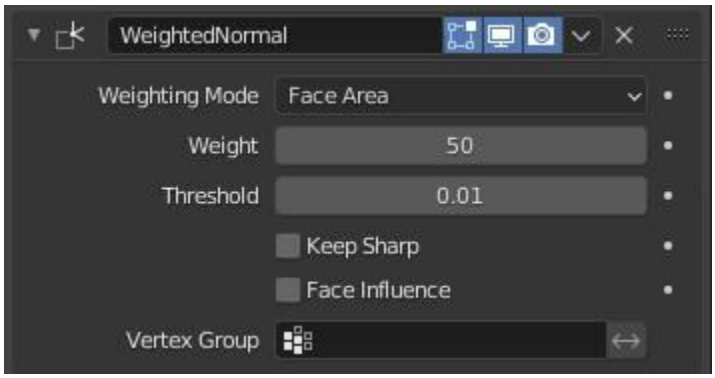
Look at the upper most normal lines. See how they are *slightly* slanted, and not completely perpendicular to grid space?

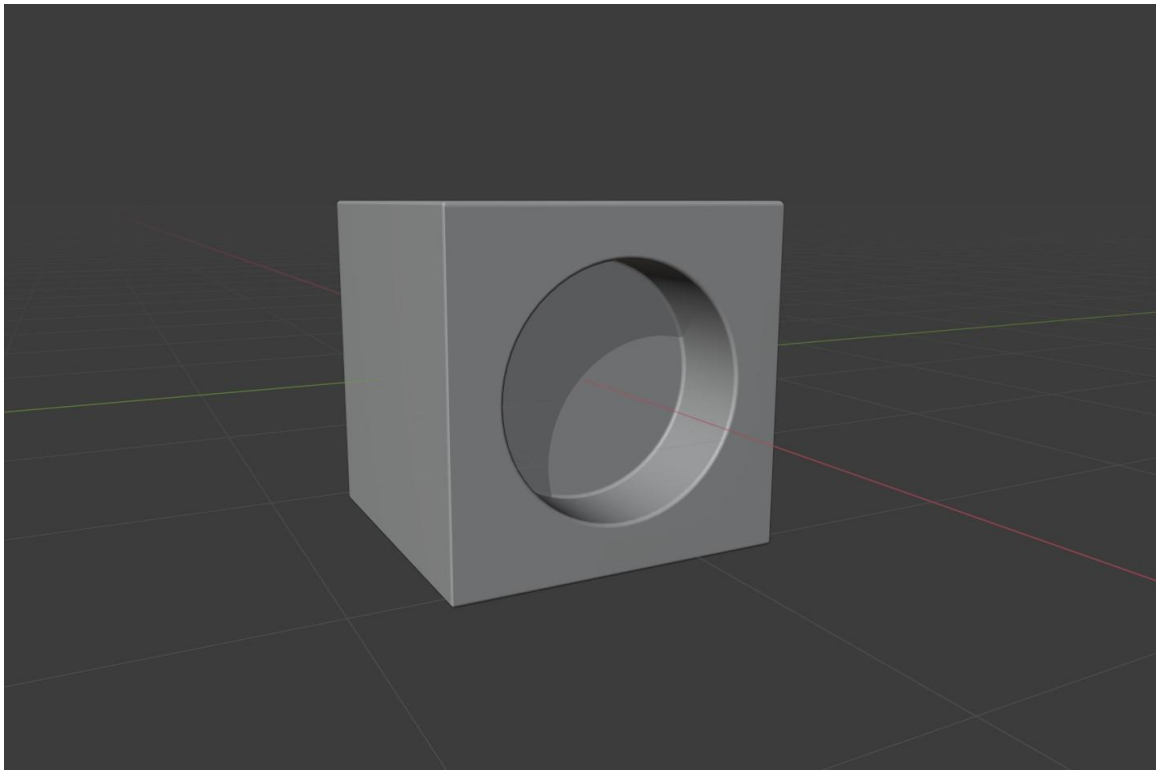


This means the shading is slightly smoothing over the faces and causing shading warps.

What we *really* want is for those normal lines in the picture above to be straight up and down (top) and straight side to side (right).

We can apply a weighted normal modifier to do exactly this, which will make the shading perfect!





Booleans (continued)

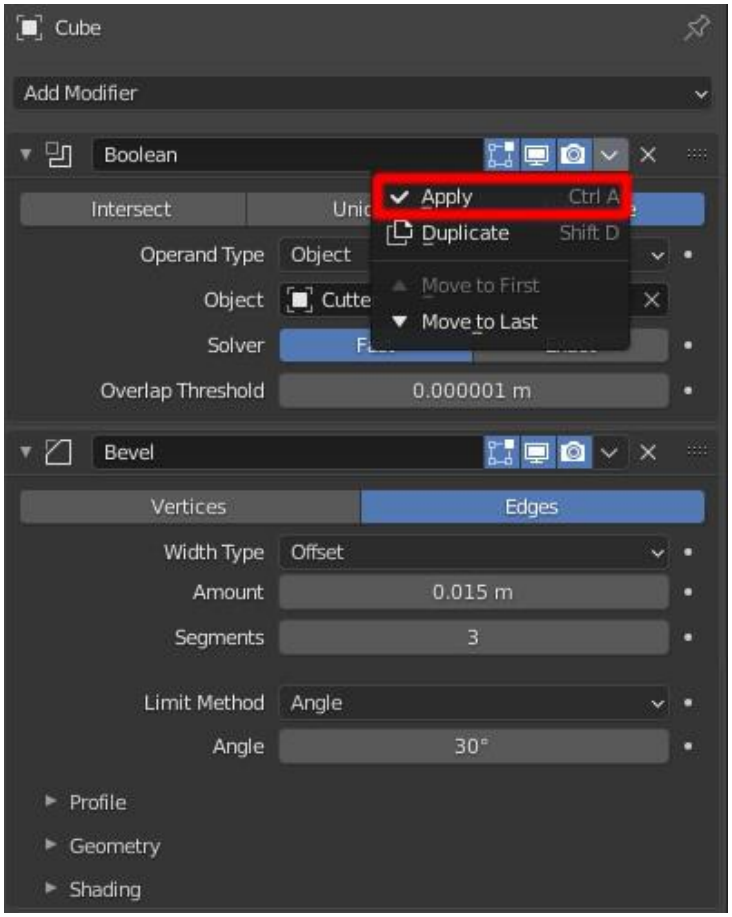
Ok, now that we understand normals and shading, let's continue.

So we've fixed the shading, applied a Boolean and successfully ran a bevel.

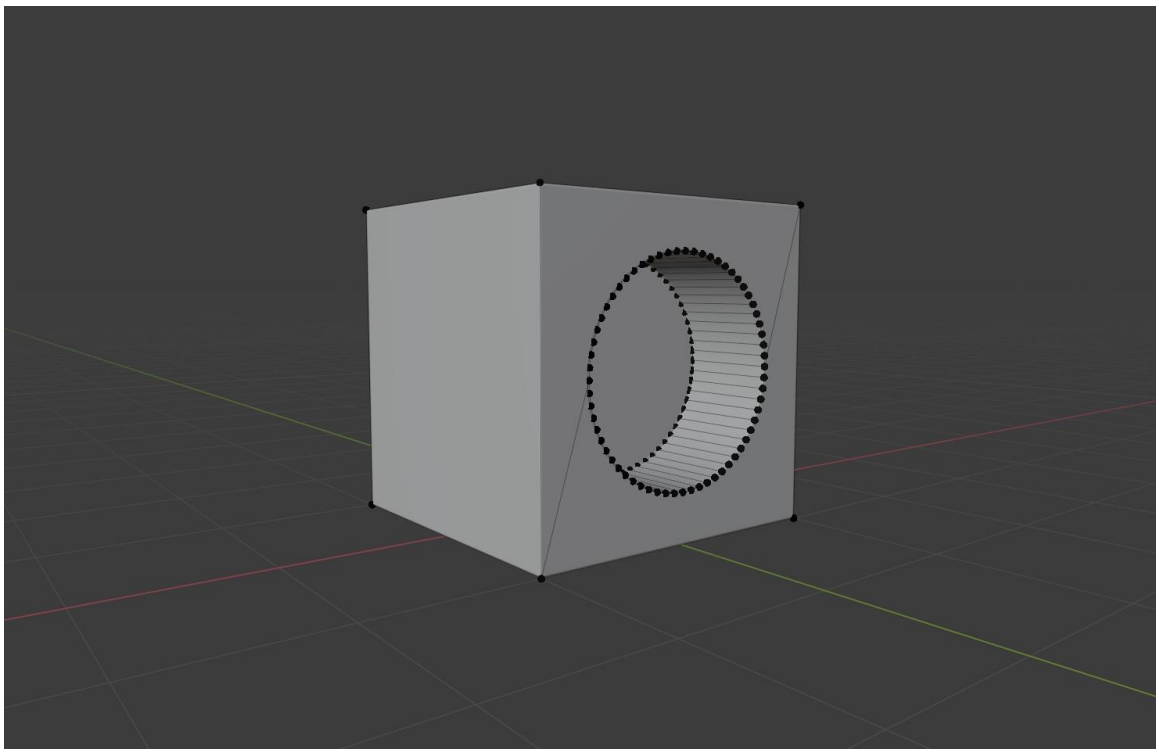
The shading is clean, and the bevel has a nice light reflection around the edges. What next? Let's talk about topology.

I'm going to go ahead and *apply* the Boolean modifier. If we don't apply it, then the changes won't take effect on a physical level.

Just click the dropdown arrow in the Boolean modifier to apply it.



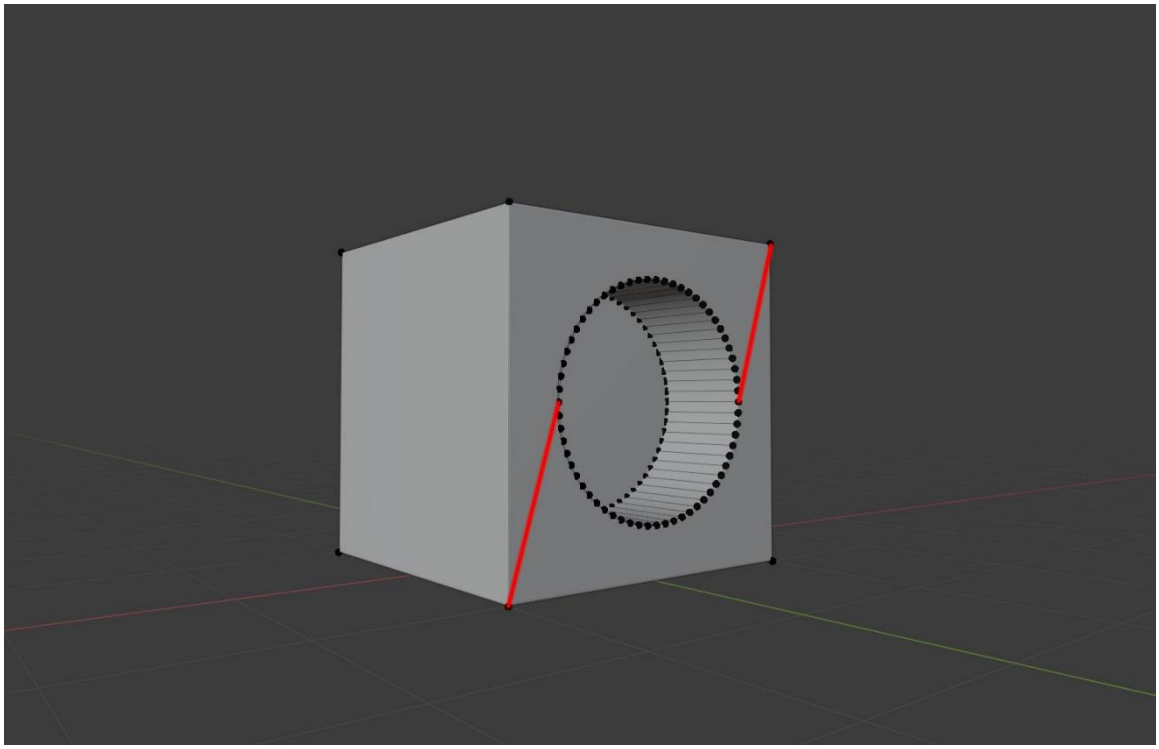
Once the Boolean modifier is applied, you will have access to the physical geometry:



There is a very simple rule when it comes to Booleans: there must be 2 connection points in the Boolean at all times.

You don't really need to remember this, per se, because it won't physically work without 2 connection points, so it is a given, but it is still good information so you know what is happening behind-the-scenes and can diagnose potential problems.

I want you to look at how this Boolean is connected to the mesh (for any newbie readers out there, *mesh* simply refers to the object you're working with).



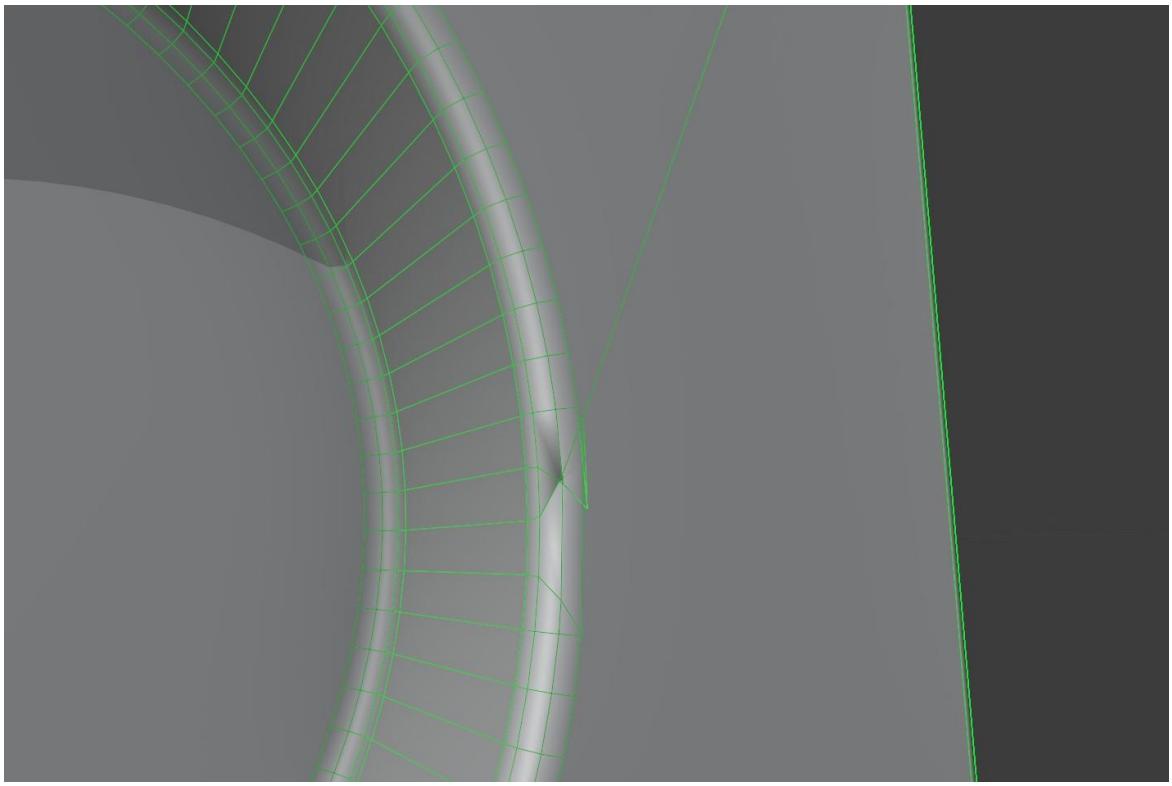
I've highlighted the connection points in red for emphasis on that area.

Blender, by default, connects Booleans automatically, and doesn't do the best job at it.

Why are these connection points so terrible, you might ask?

This is because it can hurt the versatility of our bevel.

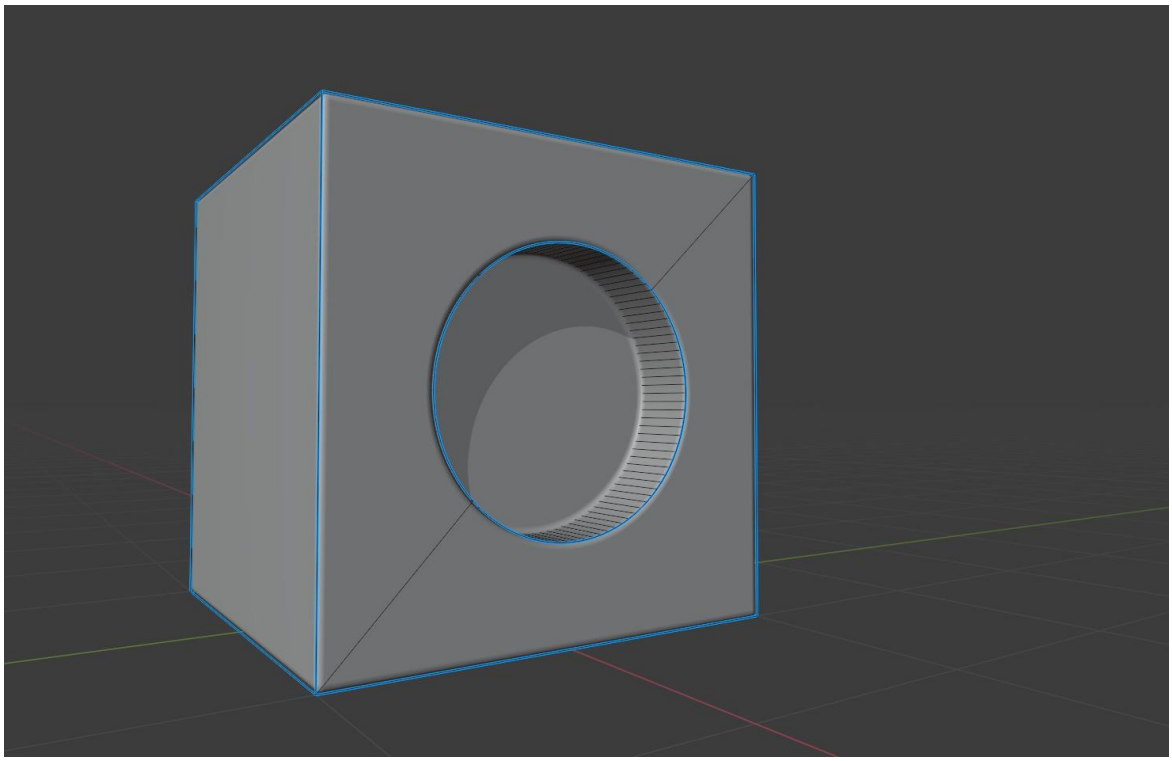
Look at the image below. Notice how after the bevel is increased too much, it kind of "runs into" that connection point? (I displayed the wireframe view as well so you can see what's actually happening).



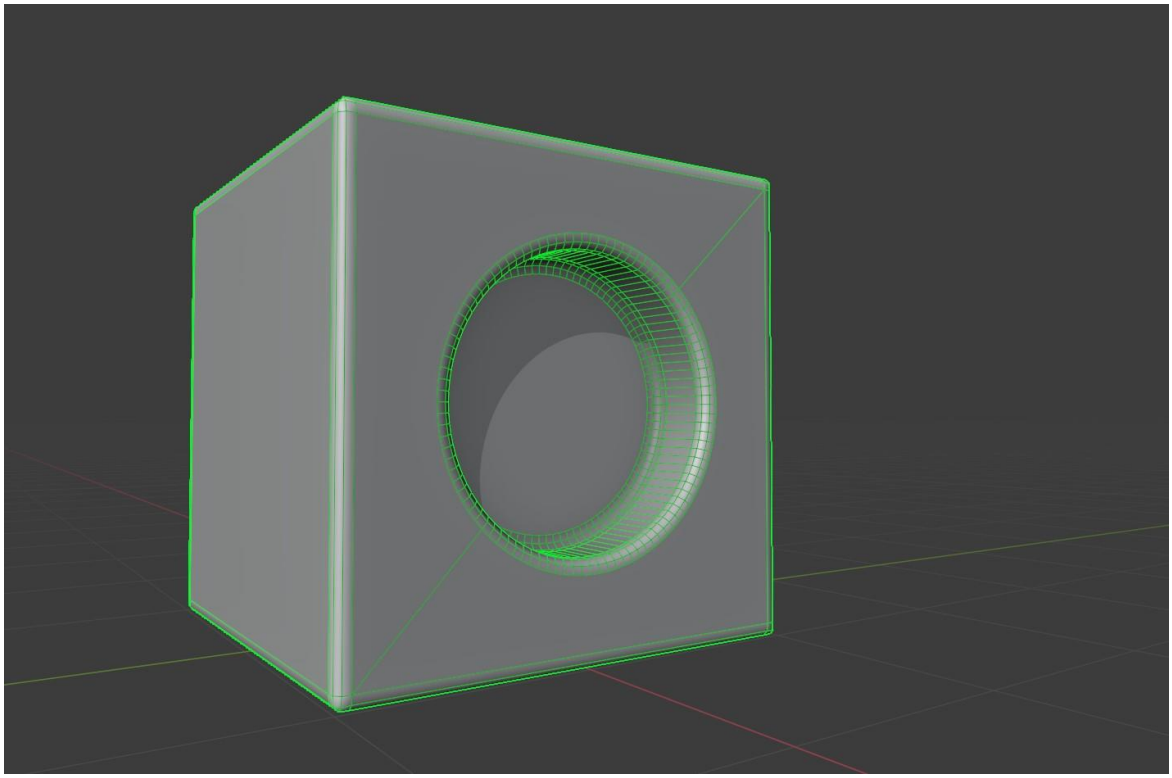
This is because that connection point is running *very* steep.

The bevel would have a lot more buffer if it had more time to bevel *before* it runs into that connection point.

For this reason, we always want to aim to have our Boolean connections as perpendicular as possible.



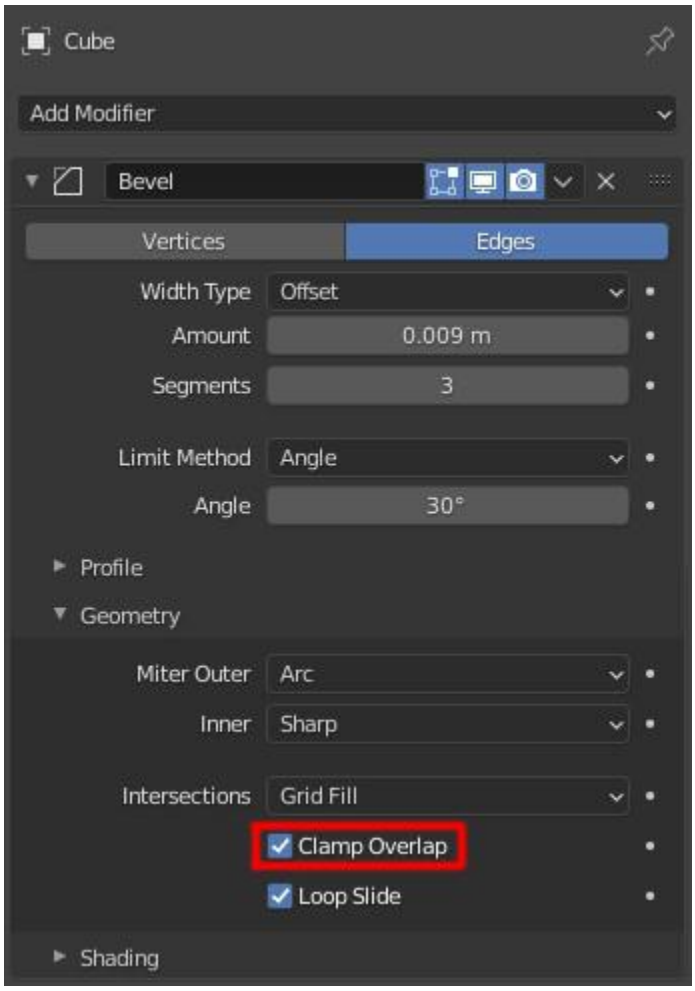
What I've done here is simply used the knife tool (or join tool, J key) to join two vertices together at a more perpendicular location.



Notice how we can now bevel as big as we want *without* running into the connection point? It continues along that connection point perfectly!

This means we have more room to increase the size of our bevel, which we sometimes want.

*As a side note, Blender by default will not allow you to bevel past the point where the bevel "runs into" the connection point. It will just... stop beveling. You can disable this by turning off the "Clamp Overlap" option in the Geometry dropdown menu in the Bevel modifier.



I almost always keep this option off unless I am diagnosing problems. Sometimes the overlaps in more complex meshes can happen almost instantly, causing the clamp overlap option to kick in before the bevel can even execute.

Usually, these overlaps are so minor and unnoticeable that I'd rather just have the option off and fix the problems later if they are severe enough to worry about.

This is kind of confusing for beginners, so let's not stress about it now.

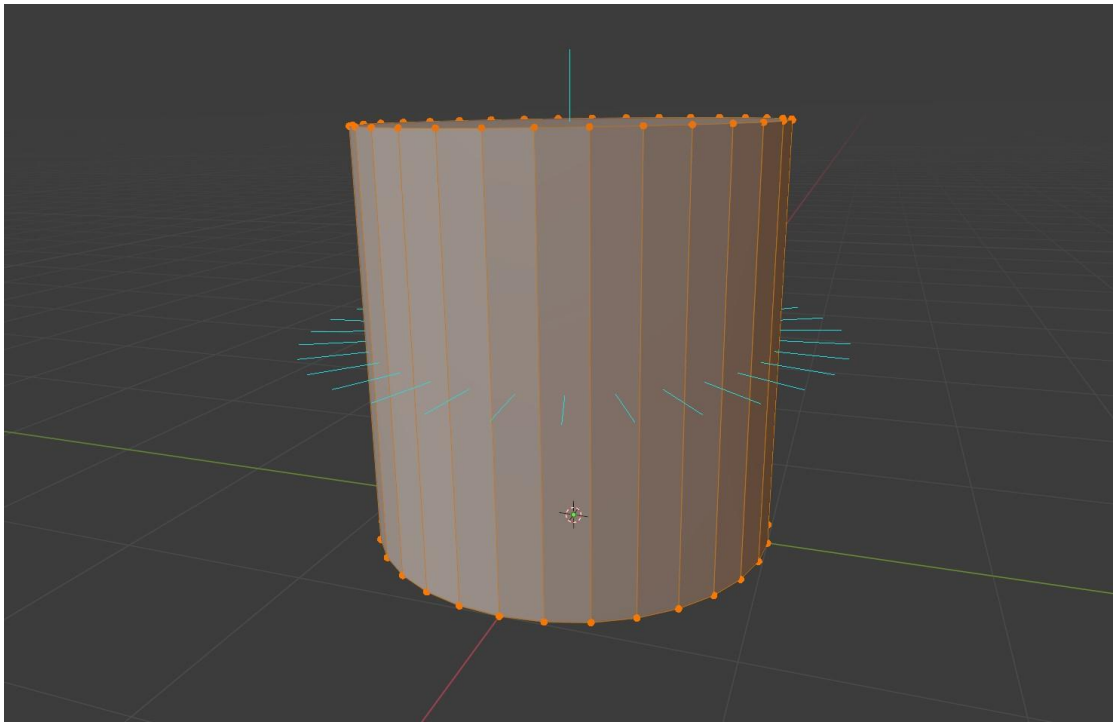
Booleans (curved surfaces)

This is where hard surface modeling really gets tricky. Not difficult, but tricky.

Booleans in curved surfaces don't behave as nicely as Booleans on flat surfaces.

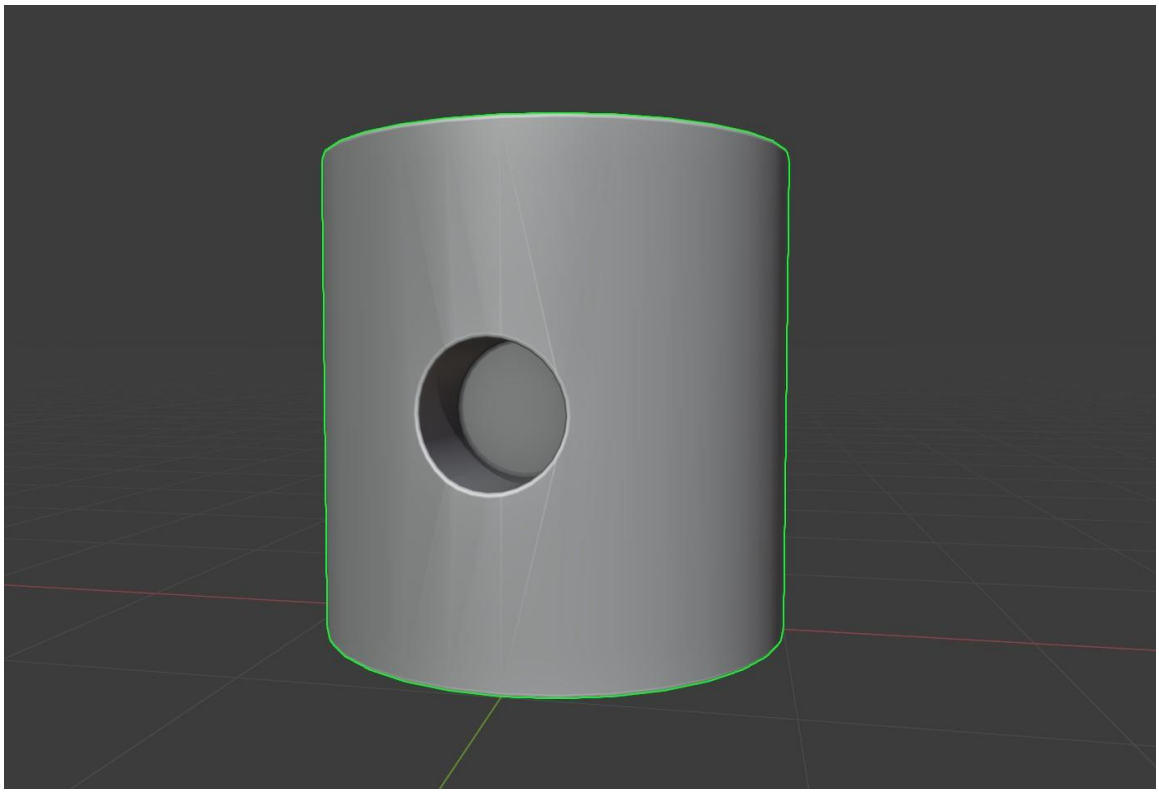
Reason being is because curved surfaces consist of multiple faces at slightly different angles making up that curvature.

Can you guess what this means? (hint: it has to do with normals!)



If you guessed that the differently angled normals throughout the curved object might be the culprit, then you are on the right track!

Let's try cutting a cylinder into a curved surface (perhaps another cylinder!) and then run a bevel.



The shading is terrible.

Unlike with flat surfaces, we can't just slap a weighted normal modifier on a curved surface and hope that the shading gets cleaned. Since curved surfaces consist of *many* different orientations, and not just a static, flat one, weighted normal is generally out of the equation in terms of actual use.

You can still add it (I usually do), but results will be very limited.

So what do we do? How can we get clean shading?

Technically speaking, the only way to get *truly perfect* shading is to have a mesh consisting of all quads.

In hard surface, though, this generally isn't very practical or efficient. We are usually more concerned with *almost* perfect shading as opposed to *truly perfect* shading. This allows us to save time and get the result without having to worry about quad-based topology everywhere.

This is an ongoing and debated discussion amongst the 3D community, which we won't get into here.

Generally speaking, though, ngons are completely fine depending on what you are trying to accomplish.

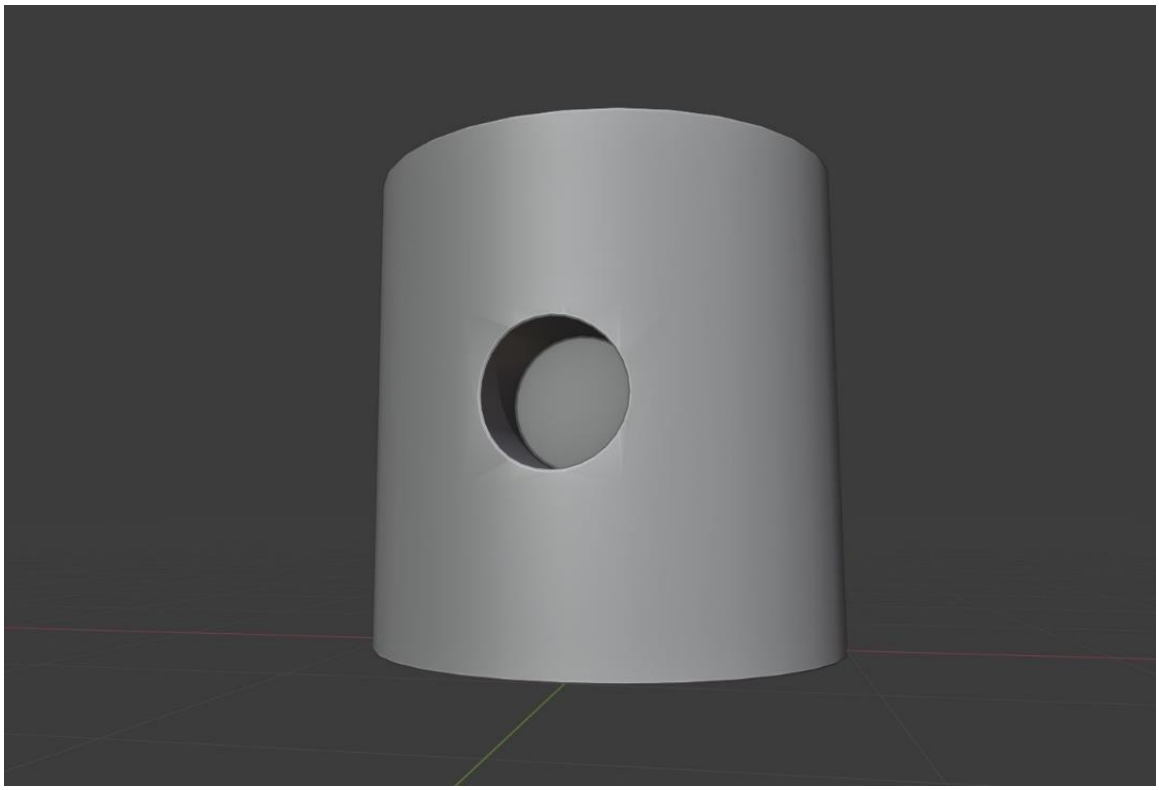
If you are an organic modeler, rigger, animator, or deal with deformations of any sort, quad-based geometry is almost always needed to get a working result. There is no way around it. Attempting to deform ngons is asking for a mess. It never works well unless the geo is incredibly dense and ngons are scarce.

Subdivision surfaces with ngons are also tricky, and we'll discuss that later.

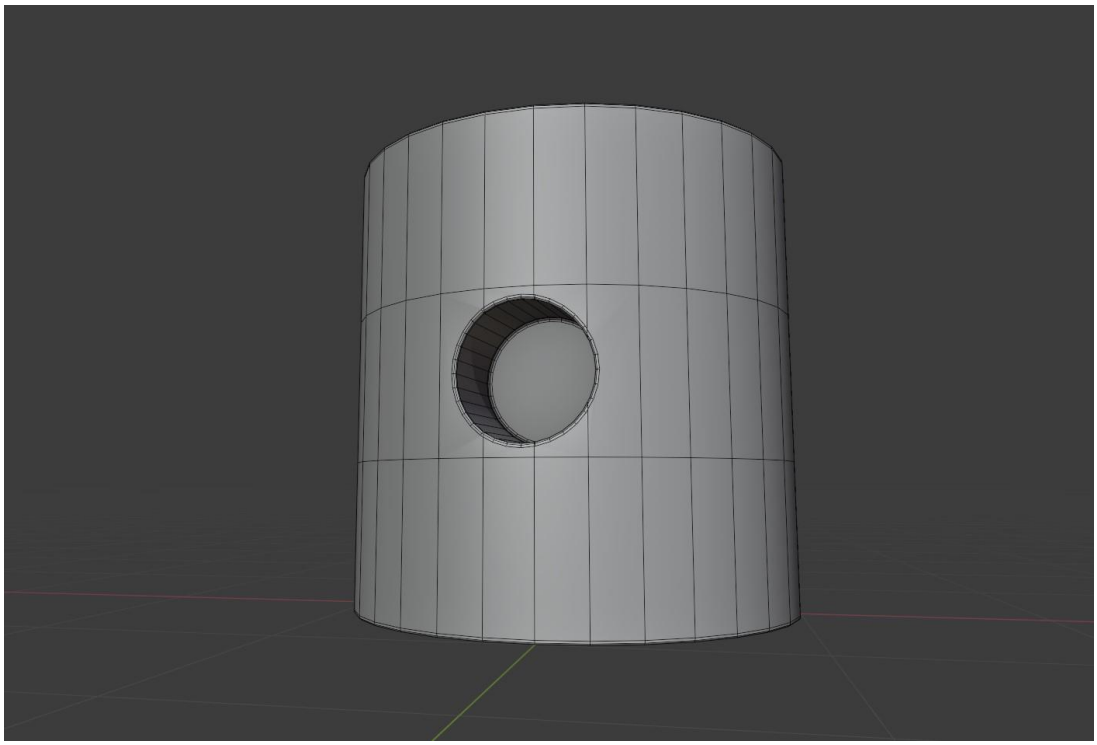
However, when working hard surface for things like concept renders, 3D printing, and even game assets if you approach it properly, ngons and Booleans are perfectly okay to use.

There can be complications when working with other software or game assets, but the subject of quad-based topology is so massive that I could discuss it for hours. For now, I'll spare you the hassle. Just keep in mind the things mentioned above and you'll be a-okay to use ngons and Booleans in your workflow.

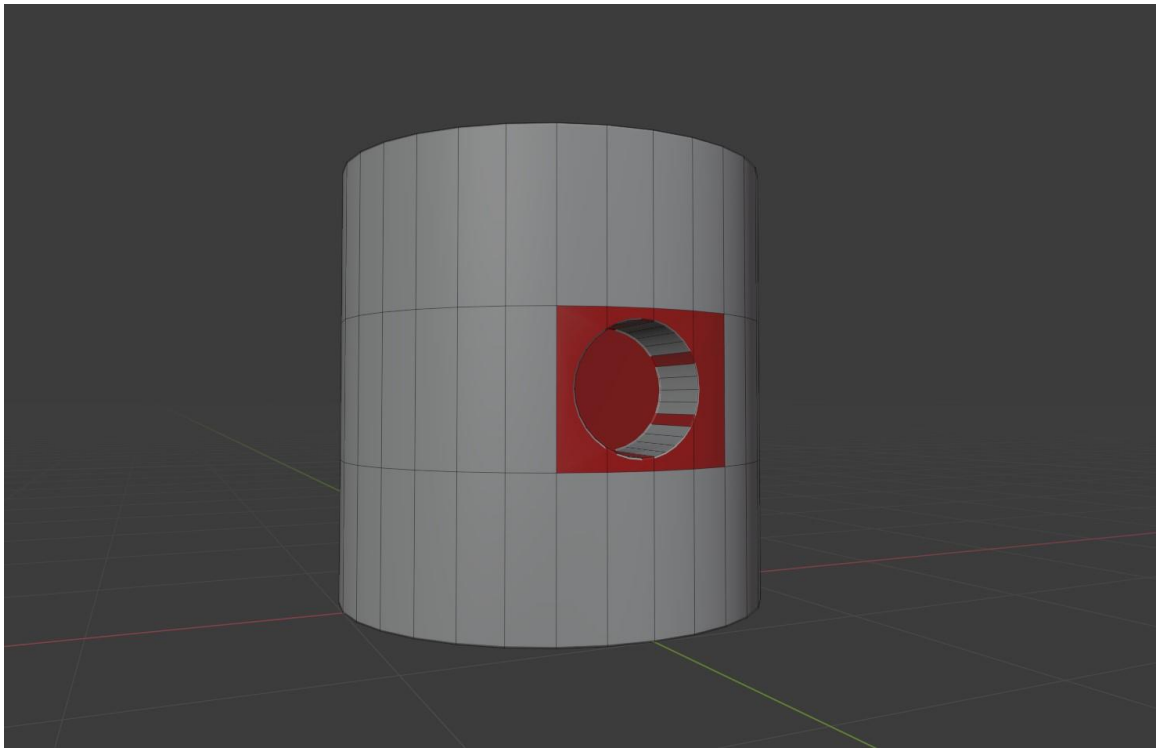
Anyways, back to the question at hand – how can we get clean shading when we run Booleans on curved surfaces? The trick is to get the Boolean isolated so tightly that the shading artifacts are barely noticeable, if at all.



This is a little bit better. What I've done here is simply added a horizontal loop cut above and below the Boolean.



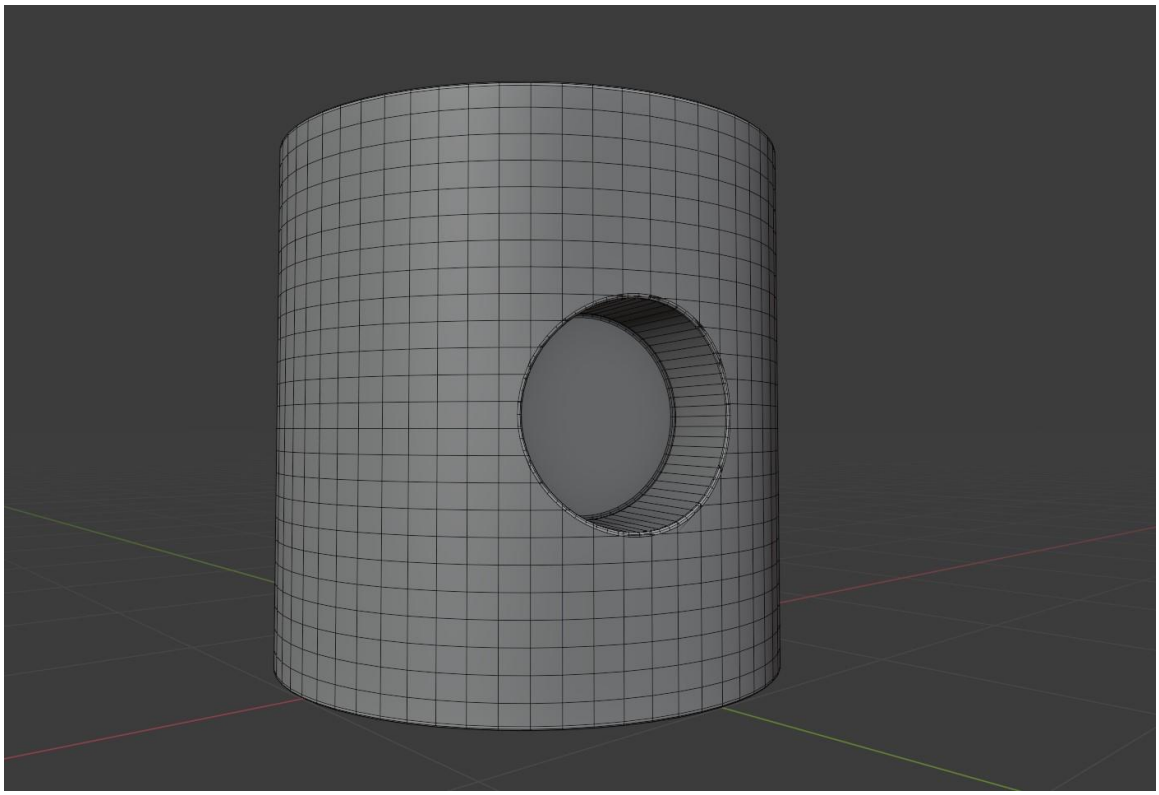
This means that the shading can only stretch up to those two horizontal loops, beyond that the shading is perfect because it is all quads!

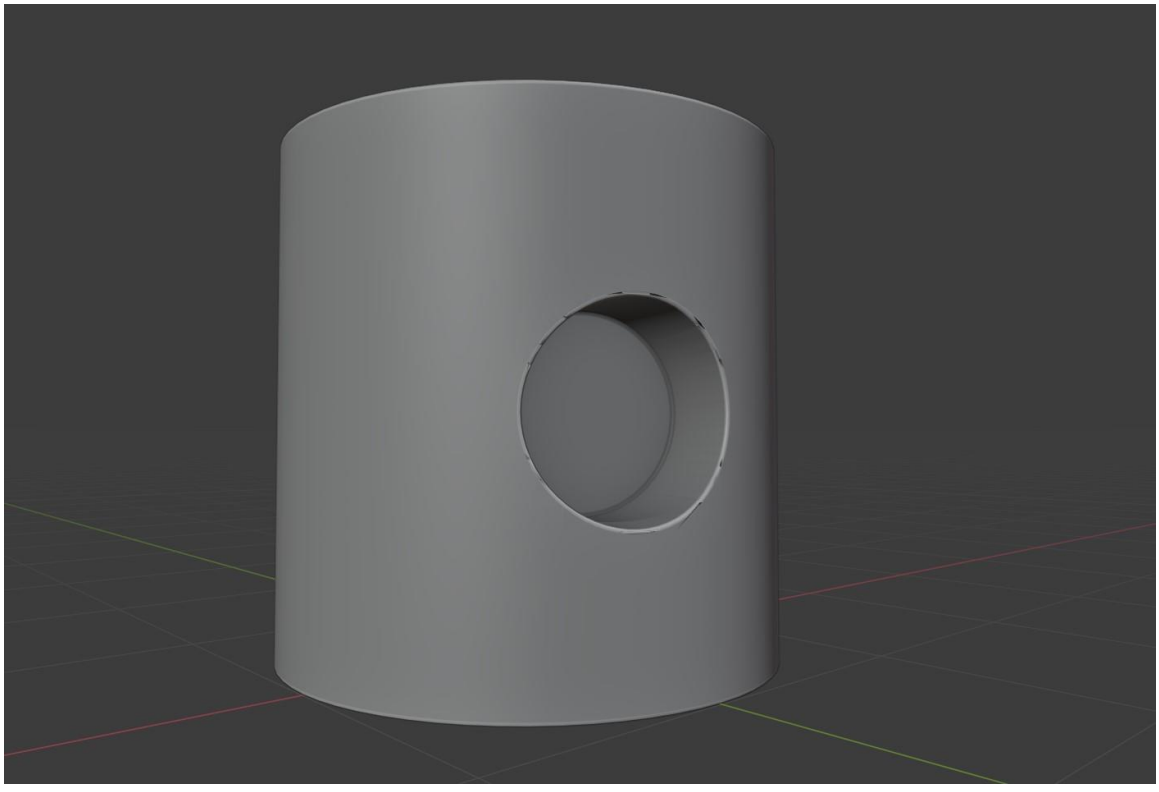


All ngons are displayed in red here, exactly the area where the bad shading is occurring! Everything else has perfect shading because the geometry is perfect (quad-based) which doesn't allow for shading error.

We can do even better though. The denser the geometry, the more isolated the shading artifacts are.

What if we used a 64-face cylinder (the ones above were only 32) and add in a good amount of loop cuts through the middle? Let's see what happens.



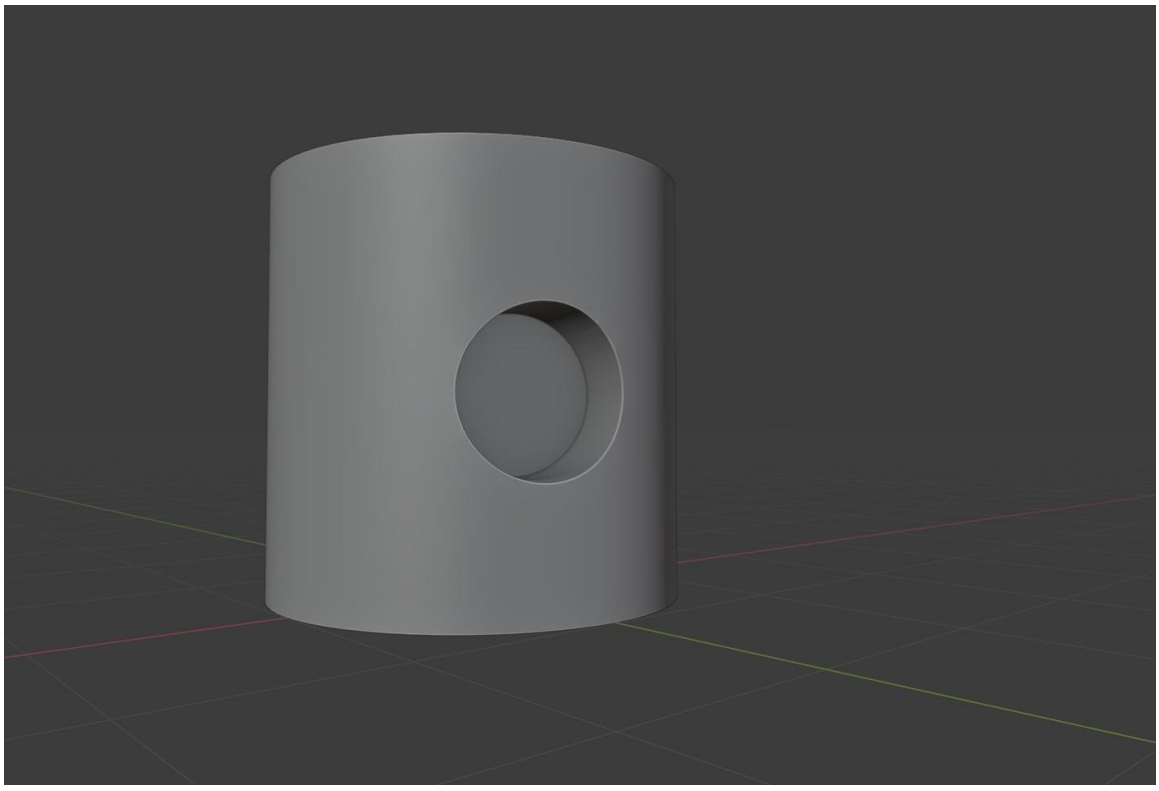


The geometry is even denser (top), and the shading is *much* cleaner (bottom).

Although the shading looks good, we have these little black spectacles all around the Boolean cut. What are they?

These are called artifacts. They occur when bevels overlap with other parts of the geometry (you briefly saw this when demonstrating the Boolean connection point issue).

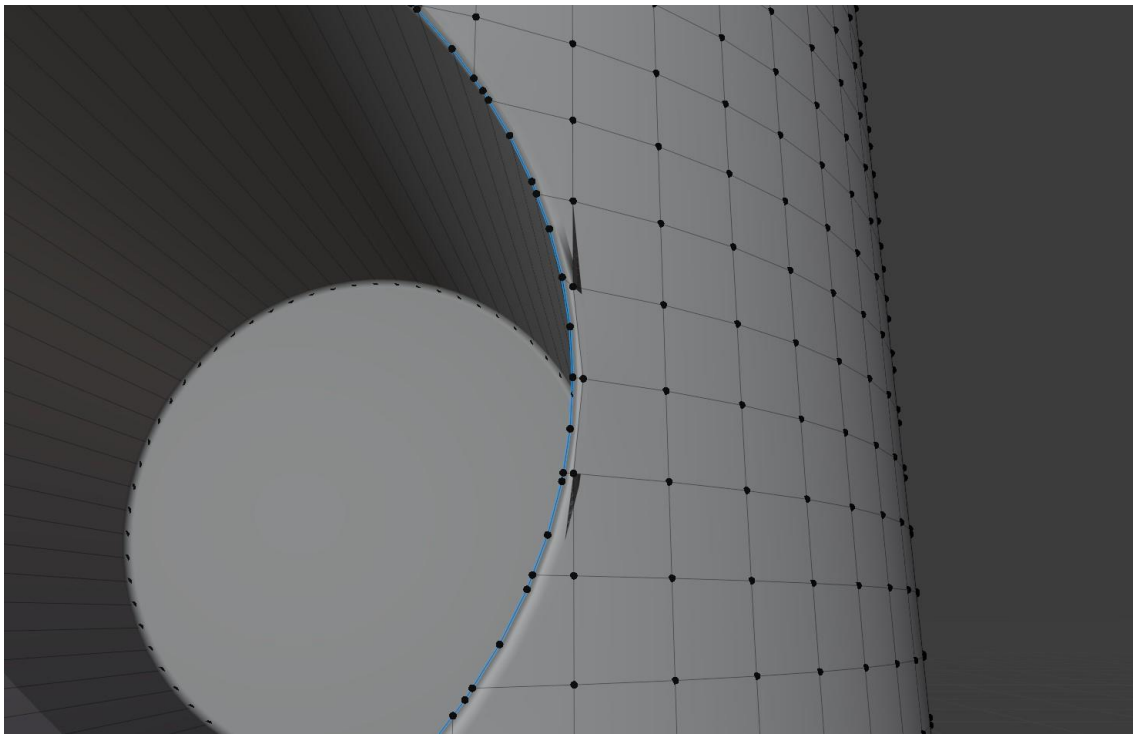
There are a few ways to fix them. The cheap and easy way (which rarely works perfectly) is to make the bevel tinier so the overlaps don't happen as often.

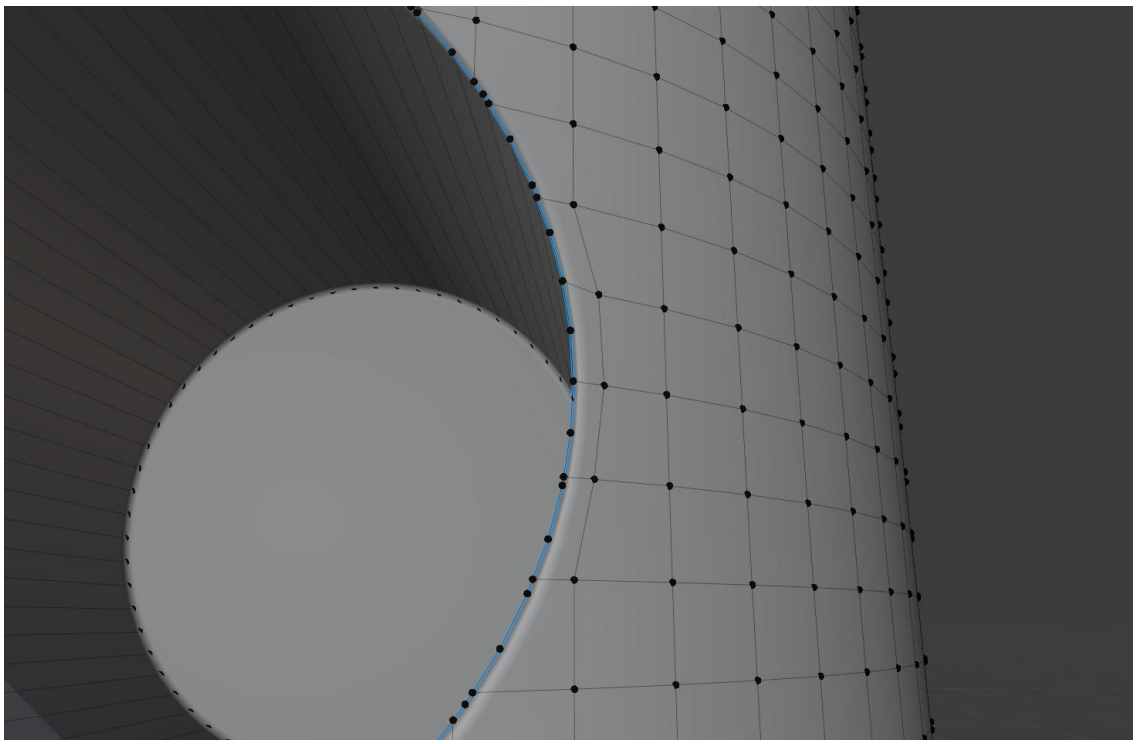


In the picture above, I made the bevel smaller, and it removed a lot of the artifacts. However, in the actual blend file, there are still quite a few artifacts that you can't really see here.

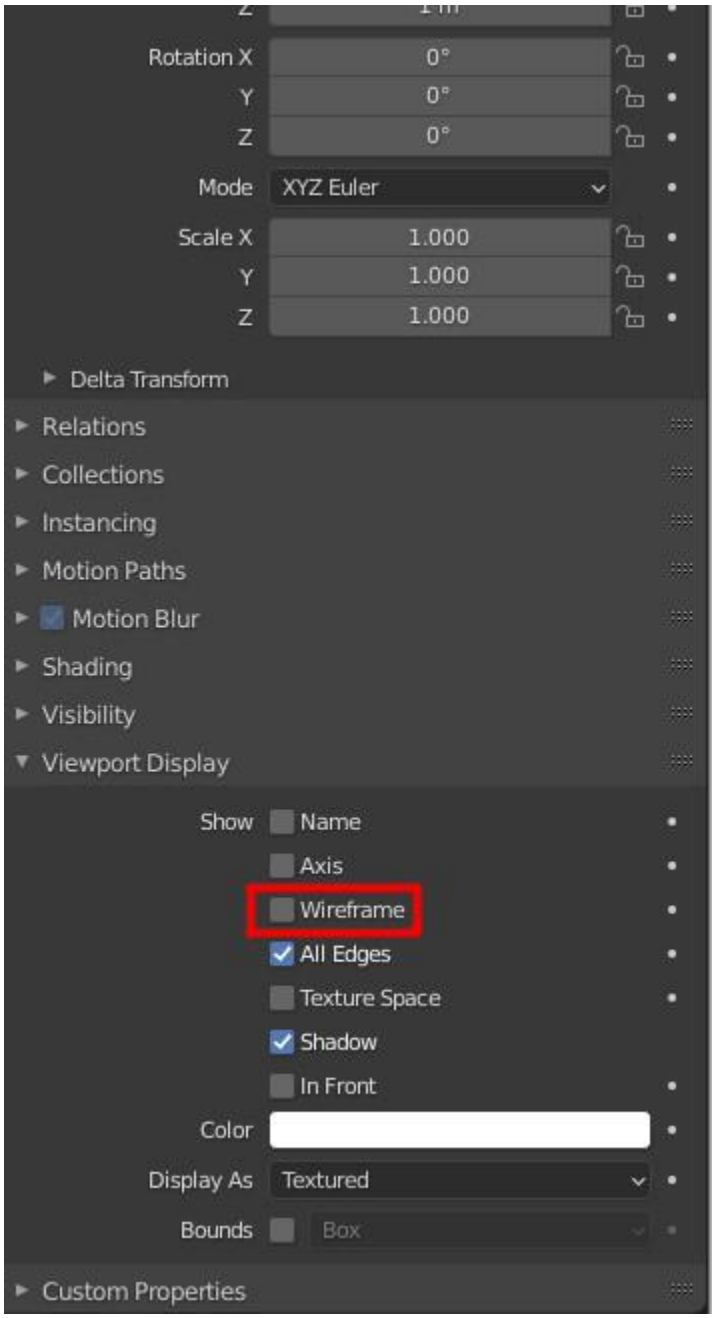
I would suggest fixing these types of issues manually. It's really easy to do.

Simply apply your Boolean modifier and move any overlapping geometry away from each other.

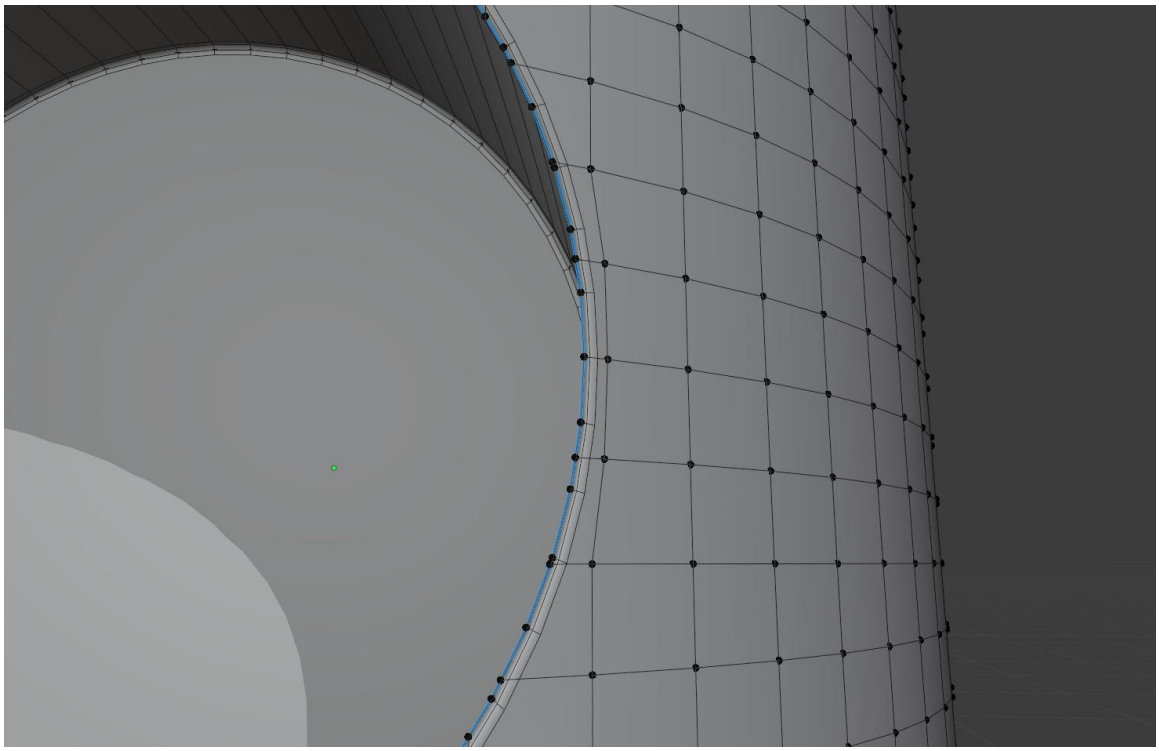




You can turn on wireframe mode if it helps you to visualize where to move things. It can be found in this menu (near the modifiers panel you'll see a small orange box, it's in there):



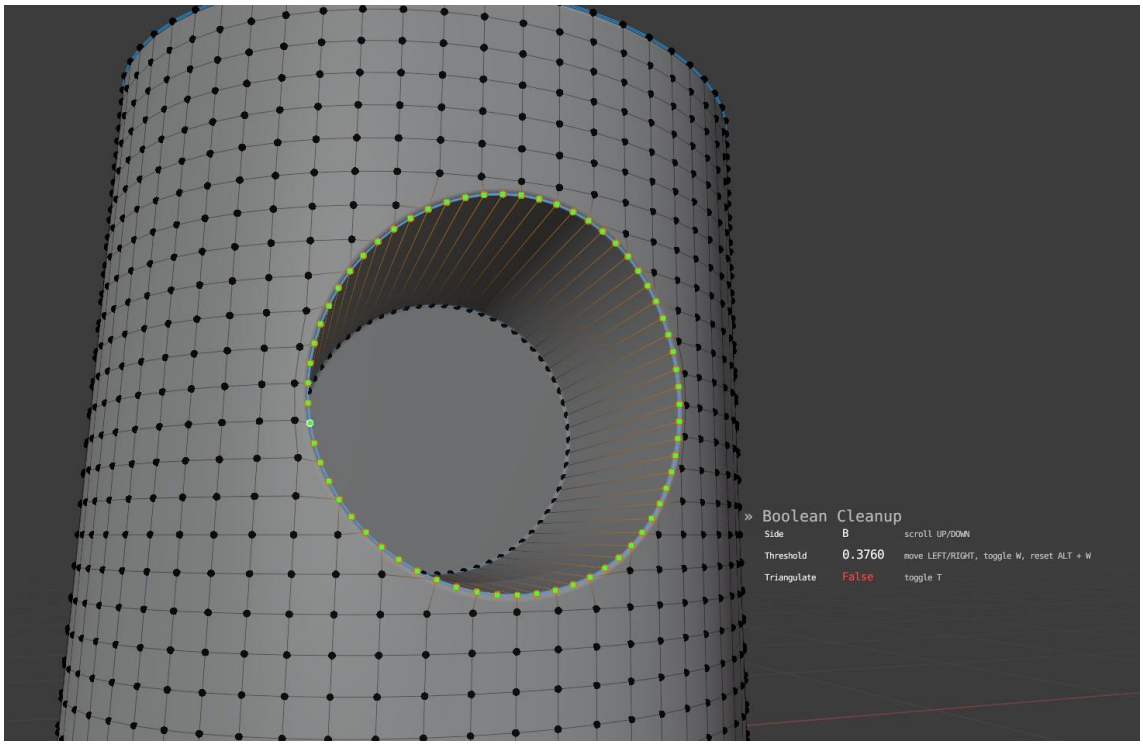
*Note – you have to enable this in object mode.



Now it's easier to see where exactly that bevel is on the mesh, and how far out you must slide your verts.

Now, as you can imagine, moving these vertices one by one all the way around each Boolean cut can be pretty tedious. For this reason, you can use an addon called Mesh Machine to expedite the process.

You basically select around the area that needs cleaned and use the "Boolean Cleanup" tool to clean (most) of it up in one click.



You might have to pick up some of the slack, but usually it's only a few leftover vertices - much better than the many you would've had to deal with before!

As a conclusion, the denser the mesh and tighter the ngons are pushed in, the less shading errors will appear. On the downside, you'll probably have more artifacts to clean up as a result of closer geometry. Choose that balance wisely.

Ngons vs. Quads – The REAL answer

I was having a chat with Chipp Walters, creator of the popular Kit Ops kitbashing addon for Blender. He is an experienced industrial designer with decades of experience, and I am always interested in picking the brains of someone who is a veteran in those industries (although I am 22, I'm not ignorant to the fact that wisdom simply comes with more life experience, so I look up to older folks for input with both life and business in general).

I remember him saying something along the lines of "the whole ngon debate nowadays is ridiculous" and this piqued my interest because it was refreshing to hear an industry veteran discuss this quad pretense, even though most of these new Boolean/ngon workflow techniques are fairly modern.

Just earlier today I asked him if he'd like to input a testimonial into this book here, and he happily obliged.

Here is what he said, verbatim:

"There are 4 reasons to use quads instead of Ngons, let's examine them.

1. "Because that's just the way it's done." This is perhaps the most often used reason and literally makes no sense. Legacy organizations may prefer quads, and many times don't even know why. More on this later.

2. "They can be subdivided to create higher resolution models." Perhaps. Many quad based meshes aren't necessarily setup for Subdivision Surfaces processing, or they are not created with proper polyflow or topology. Also, there are many other considerations when using quads for SubD, like what type of UV mapping and including various resolutions for uvmaps.

3. "They are good for character design, amorphous forms, and even hard surface models that are to be 'wiggled', 'deformed', and otherwise animated." This is the best reason for using quad topology and Subdivision Surfaces.

4. "They give less problems with shading artifacts." There is partial truth to this. High subdivision-surfaced quad meshes tend to render well with little to no normal smoothing adjustment. And before advanced weighted normal editing, this was probably the reason for the "just because" notion explained earlier. But now, with proper hard surface modeling techniques, this issue is mostly insignificant.

Then what are the advantages of Ngons?

1. Meshes constructed with ngons are much faster to make and much easier to edit compared to all quads.

2. Because of this, Ngons provide for an iterative design workflow, when the artist can try out different concepts and many times in a non-destructive way. All 3D concept artists use an Ngon workflow.

3. You can use Booleans on Ngons meshes. And you can even animate the Booleans as well-- in fact much easier than with a quad based Subdivision Surfaces mesh, for which using Booleans is just about impossible.

4. CAD files always import with non-proper quad meshes. In most cases, there is no need for retopology.

5. Ngon workflows are also better at creating parts for 3D printing because of modeling speed and the use of Booleans and are easily tested for non-manifold surfaces."

This is directly from Chipp, and essentially reiterates what we briefly touched on earlier with ngons vs. quads.

I believe the most compelling reason for using quads is for organic or deformed surfaces, or perhaps if you're in a AAA industry where it is a specific technical requirement as part of the job.

It is also quite easier to UV Unwrap quad surfaces, but it isn't a requirement.

Trim sheets are also tricky to use without a nice quad-based poly flow, but you can still get away with it if you attack the problem correctly.

Other than that, Booleans/ngons all day.

Point #2 in Chipp's discussion brings us into a new topic, the topic of non-destructive vs. destructive modeling.

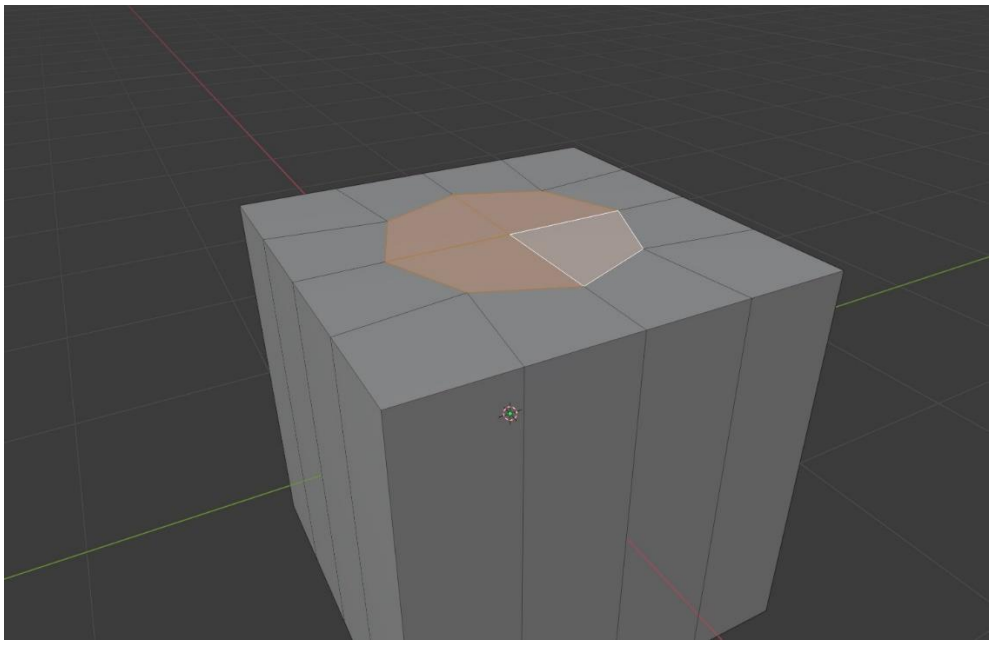
Non-Destructive vs. Destructive Modeling

There are two main workflow techniques: working destructively and working non-destructively. Let's discuss both.

Destructive workflow is what you're usually taught from the get-go: extruding, manipulating the existing geometry, adding loops cuts, etc. You probably did this when learning the subdivision surface workflow. This is called *destructive* because you are physically adjusting the mesh, and the only way to undo any changes is by using the undo button, which lacks long-term versatility.

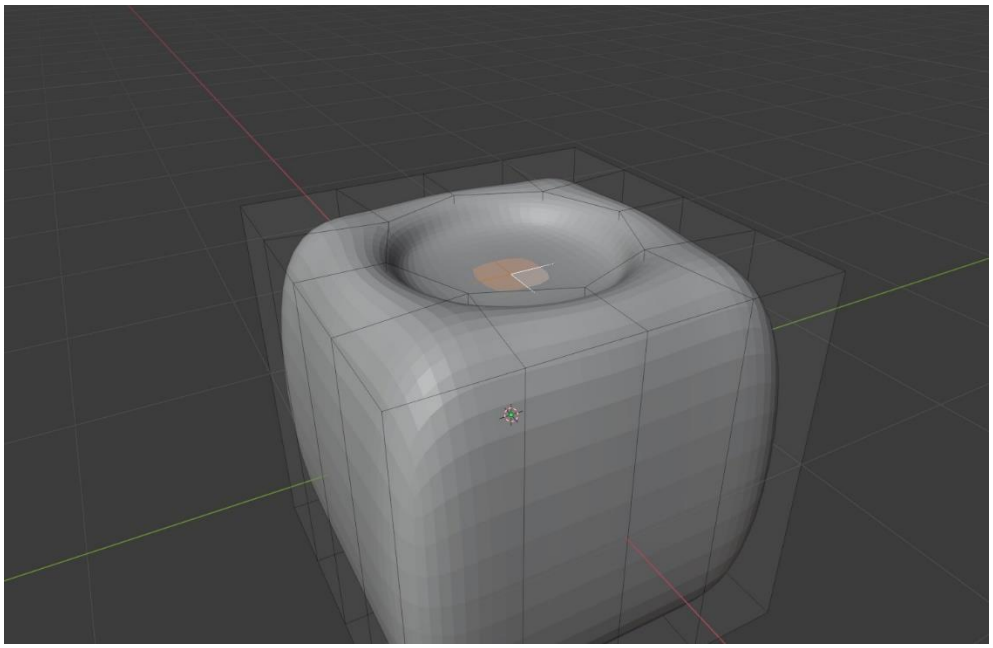
Non-Destructive workflow on the other hand allows for frequent changes whenever necessary, removing the need to have constant backups because you can adjust the geometry on the spot.

Let's take a look at a basic example:

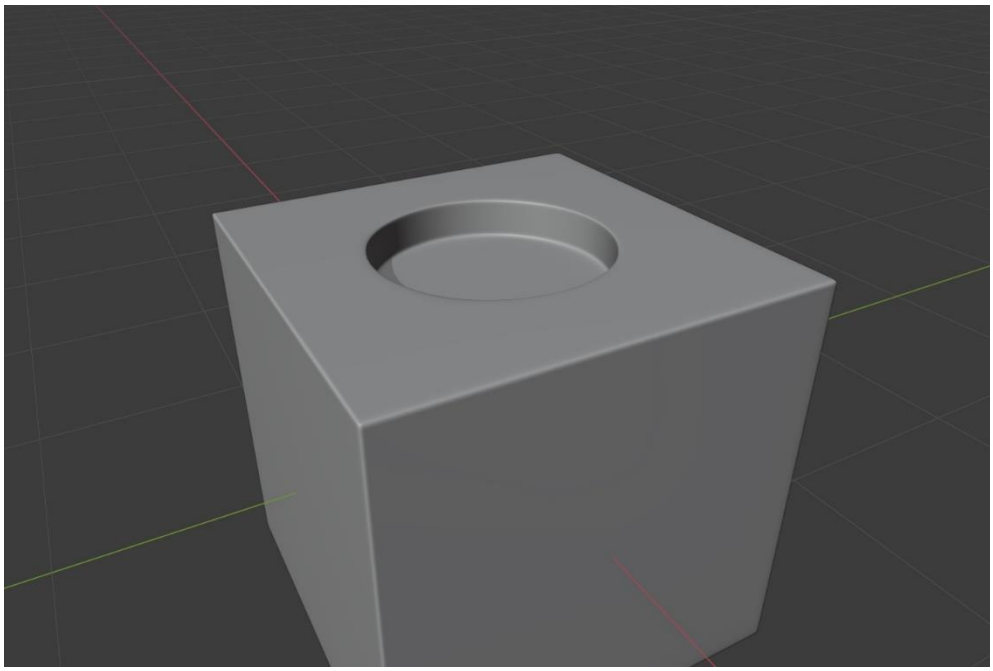


Here I have a simple set of loop cuts through a cube, and I selected the middlemost faces and used the Loop Tools addon to convert it into a circle (instead of a square shape).

Next, I'll extrude down, and add a subdivision surface modifier:

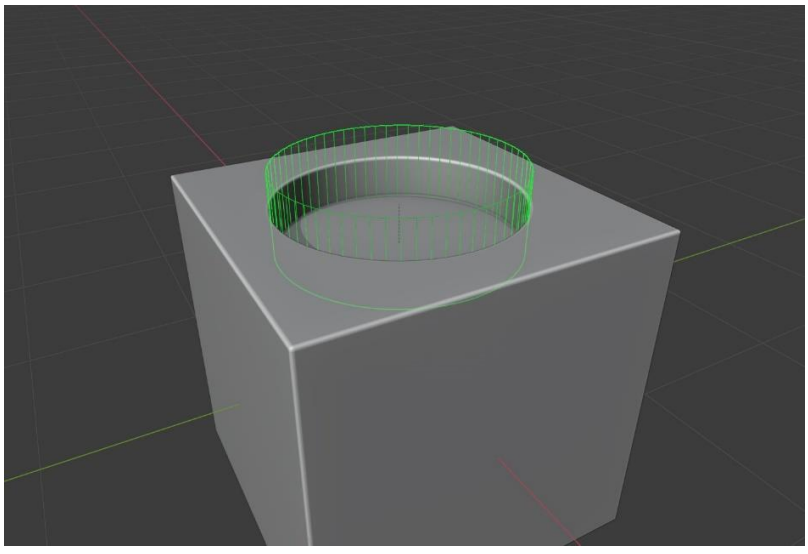


We'll tighten it up using proximity loops to get our final result:



Now what if I want to move the cut in the center somewhere else, perhaps a bit closer to a corner? Well, I can't that easily. I'd have to select all the geometry making up that circle and move it manually. This can be pretty tedious, if not impossible for more complex geometry that also affects other areas of the mesh.

What if we simply used a Boolean instead of this multi-step subdivision surface process?



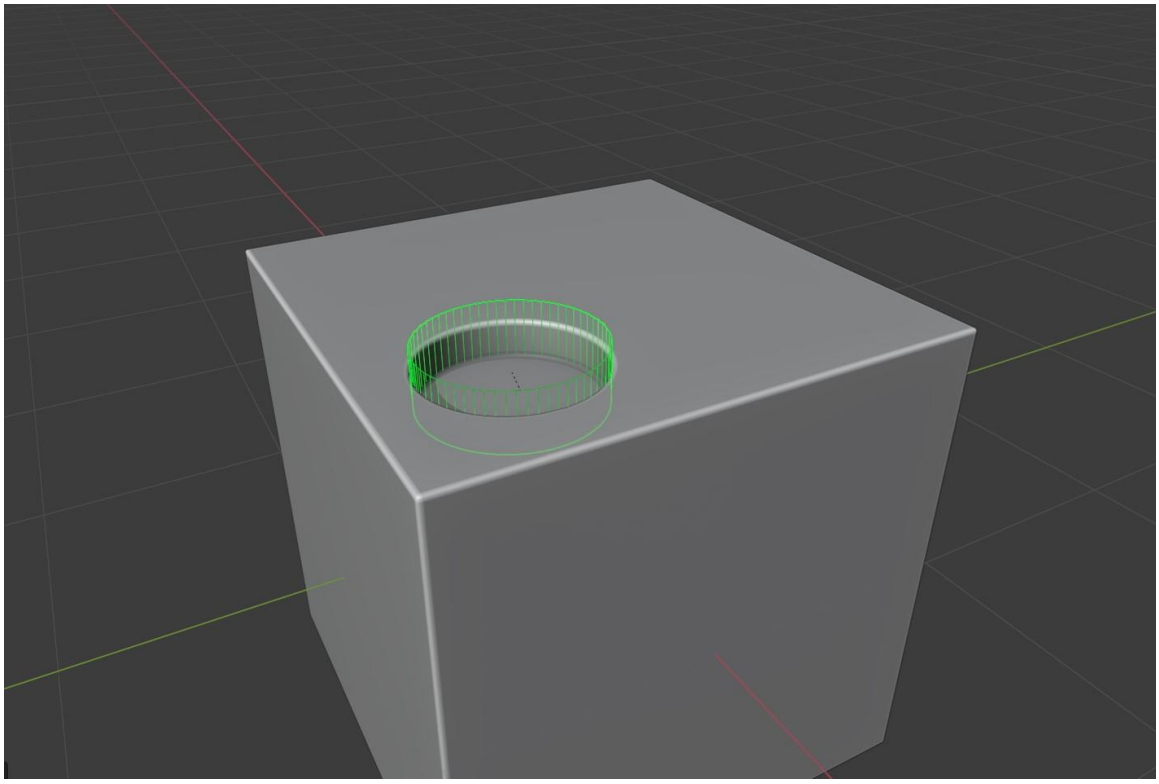
In literally just a click or two, I already have the same exact result.

Before I tell you – what is the big difference here (besides speed)?

Take a second to think about it.

No seriously, you'll feel like a pro if you know the answer instead of having it handed to you, so please think of the answer – I'll tell you on the next page!

So, the answer here – and we've already hinted at it – is that if we want to move that cut in the top, all we have to do is move the cutter object. Boom! No need to select all of the geo manually because we are working *non-destructively* – the Boolean hasn't been applied so it is still live for adjustment in the viewport!



This is the power of non-destructive workflow, and it is one we use often in hard-surface modeling.

Of course, there will be times you have to apply your Booleans because there is simply no other solution. You might have to clean up the geometry, apply bevel weights manually, or simply apply them because too many Booleans can harm performance.

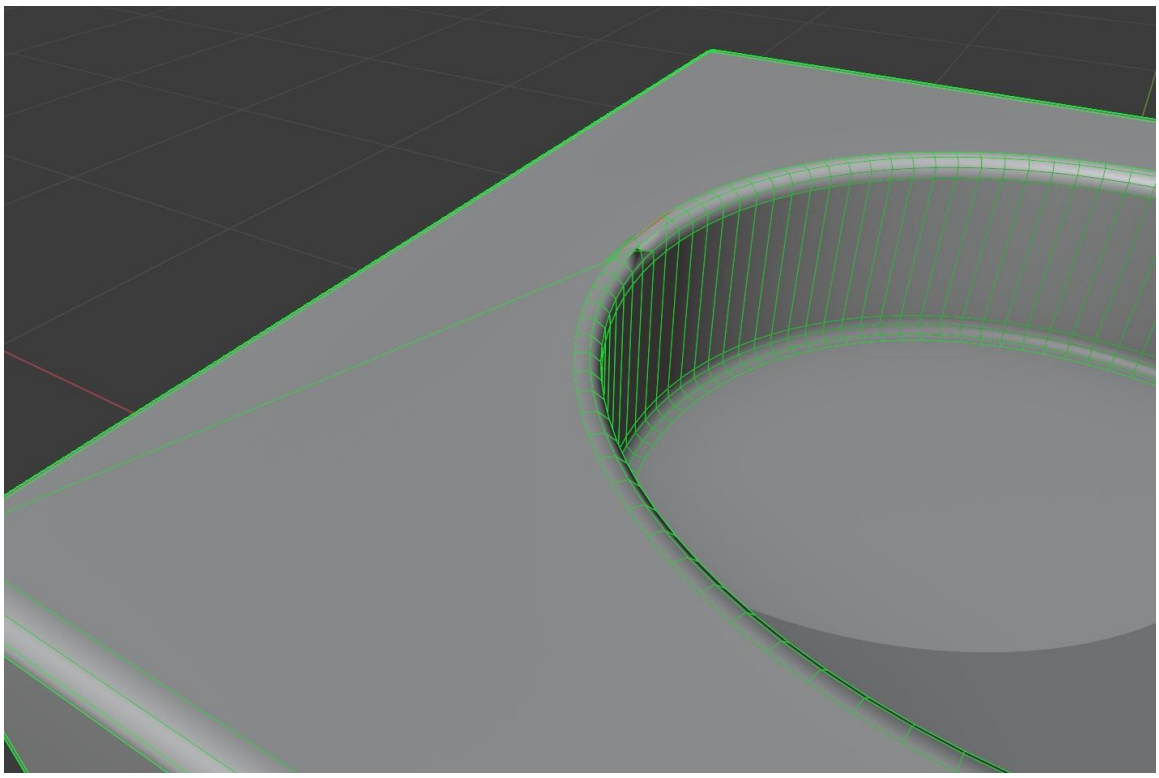
Either way, it is beneficial to work as non-destructively as possible until you absolutely need to apply the Booleans. My partner, Ponte Ryuurui, tends to work a lot more destructively than I do – it all comes down to personal preference.

Boolean Cleanup Strategies

Cleaning Booleans can be a bit… tricky, at least at first. Let's explain a few concepts and then we'll continue with the strategies.

Arguably, the *number one* issue when running Booleans is that you can encounter *overlaps* with your bevels. We've briefly touched on this already, but we will recap here.

Remember when we talked about Boolean connection points? The most obvious issue was when the bevel overlapped with them.



What was the solution to fixing this? Simply running a more practical connection point at a perpendicular angle so the overlap doesn't happen that fast (if at all).

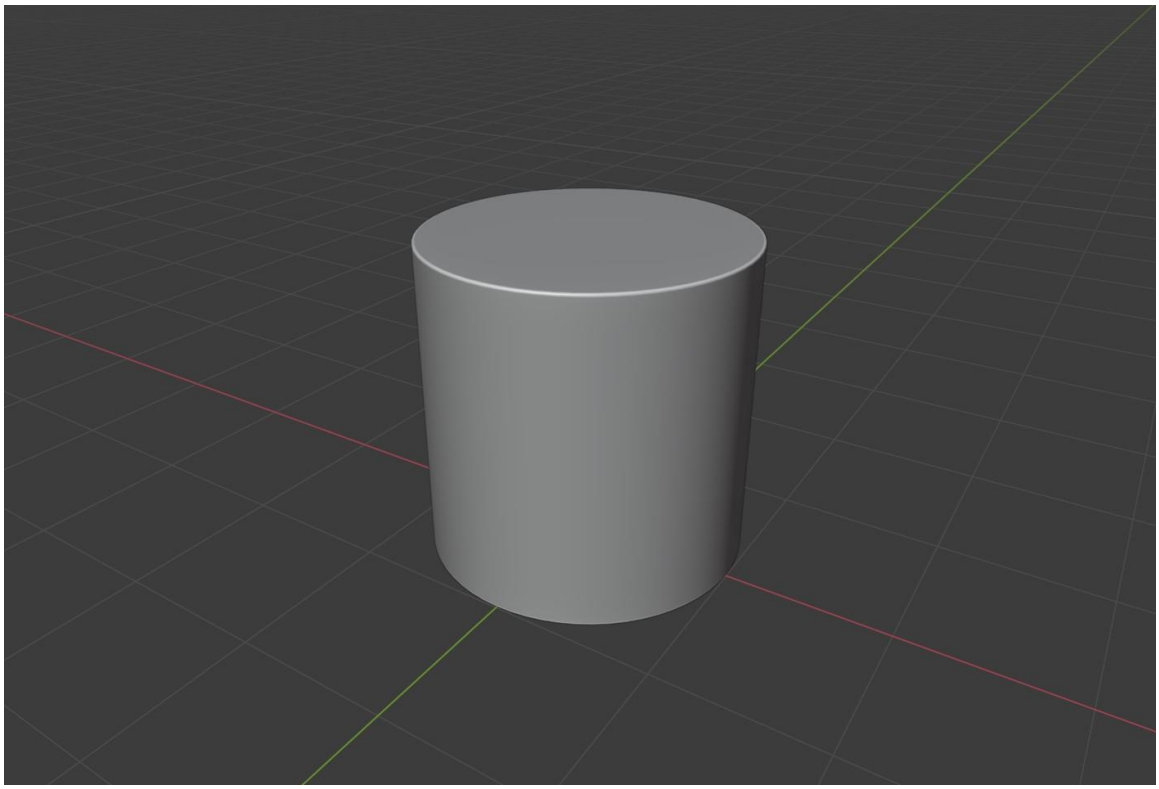
This is the concept behind *Boolean cleanup*: we want to make sure that there aren't any geometry overlaps. If there are, the mesh will look terrible – artifacts will be everywhere!

This doesn't look professional at all, especially for a portfolio piece.

If they are super small and you're doing concept renders, sometimes they are unnoticeable, so you can probably leave the small ones alone, but try not to make a habit of it. You need to first learn the rules before you can break them.

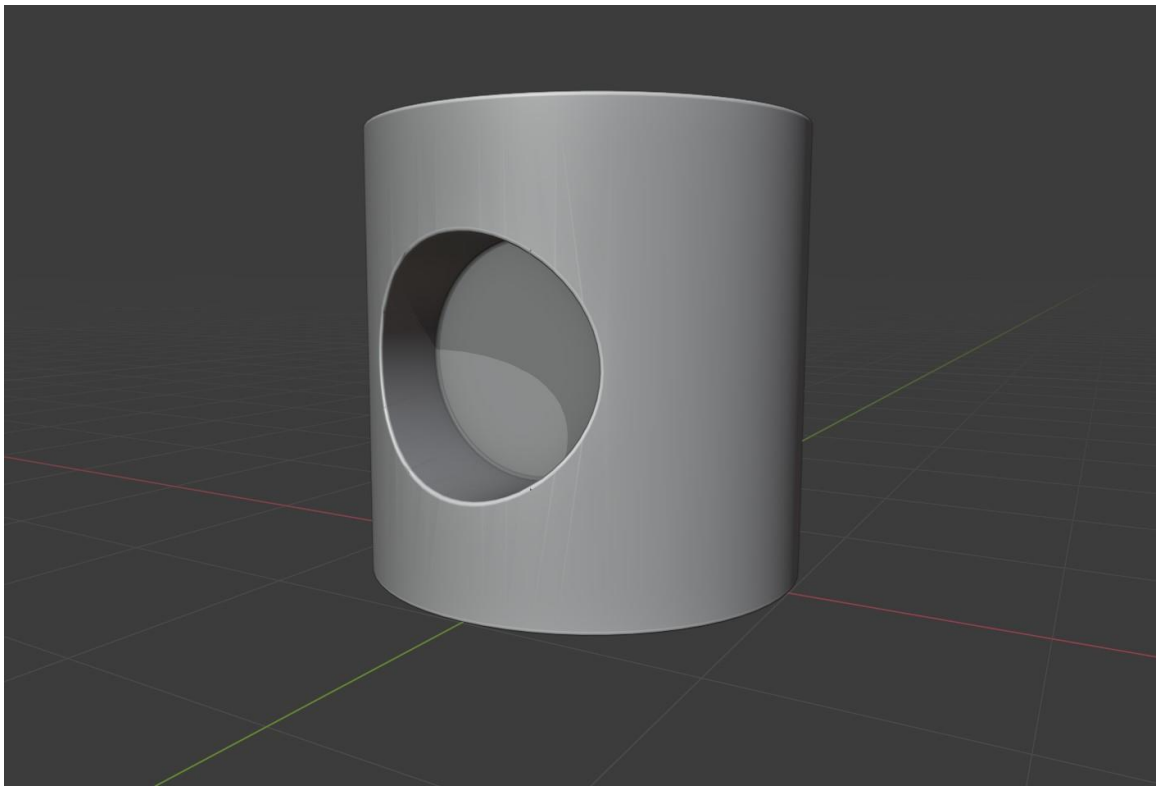
Let's look at a more complicated situation, involving a cut into a cylinder.

I'll be using a 64-vertex cylinder for this demonstration.



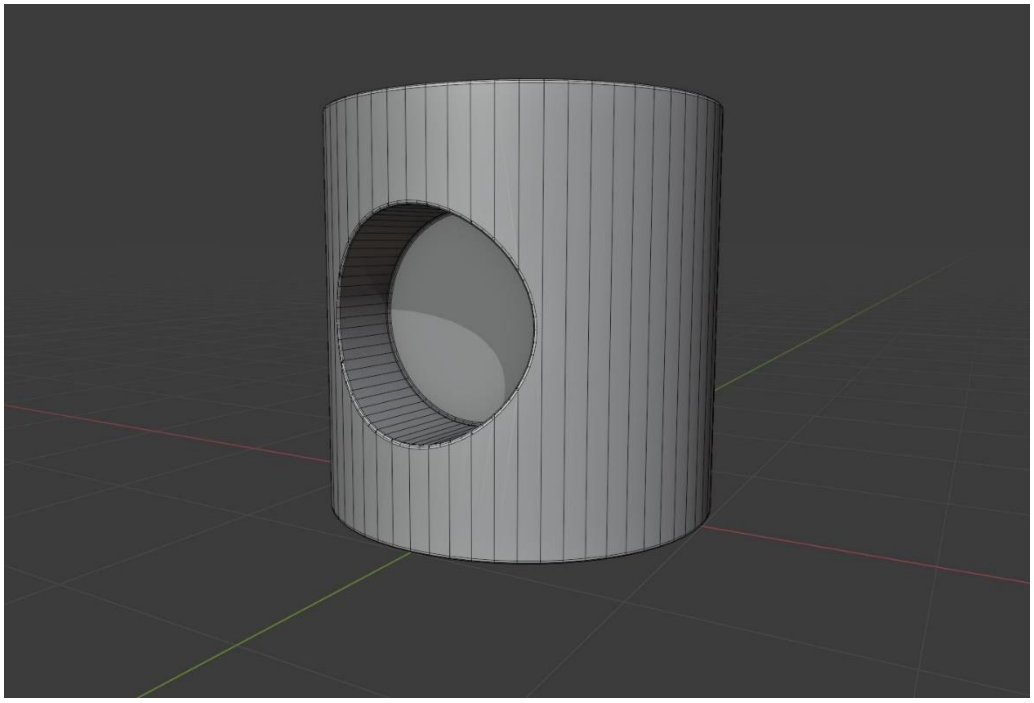
Simple cylinder, small bevel. Nothing crazy.

Let's run a cylindrical Boolean into the side, shall we?

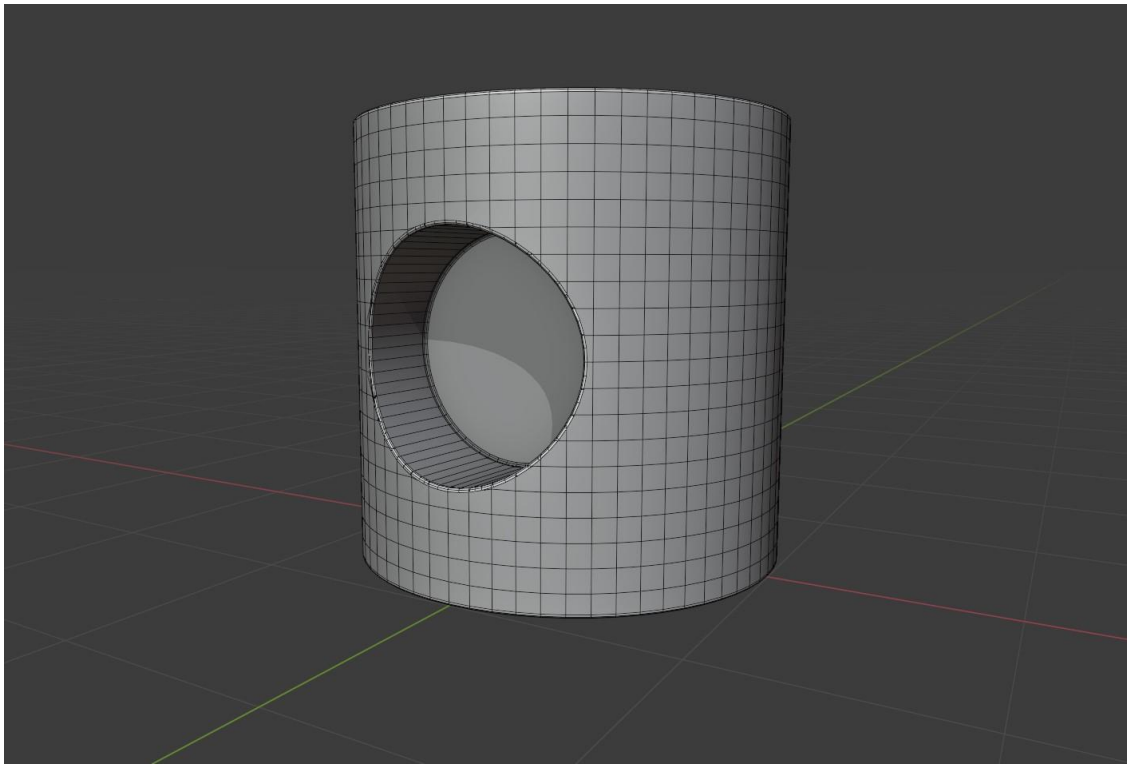


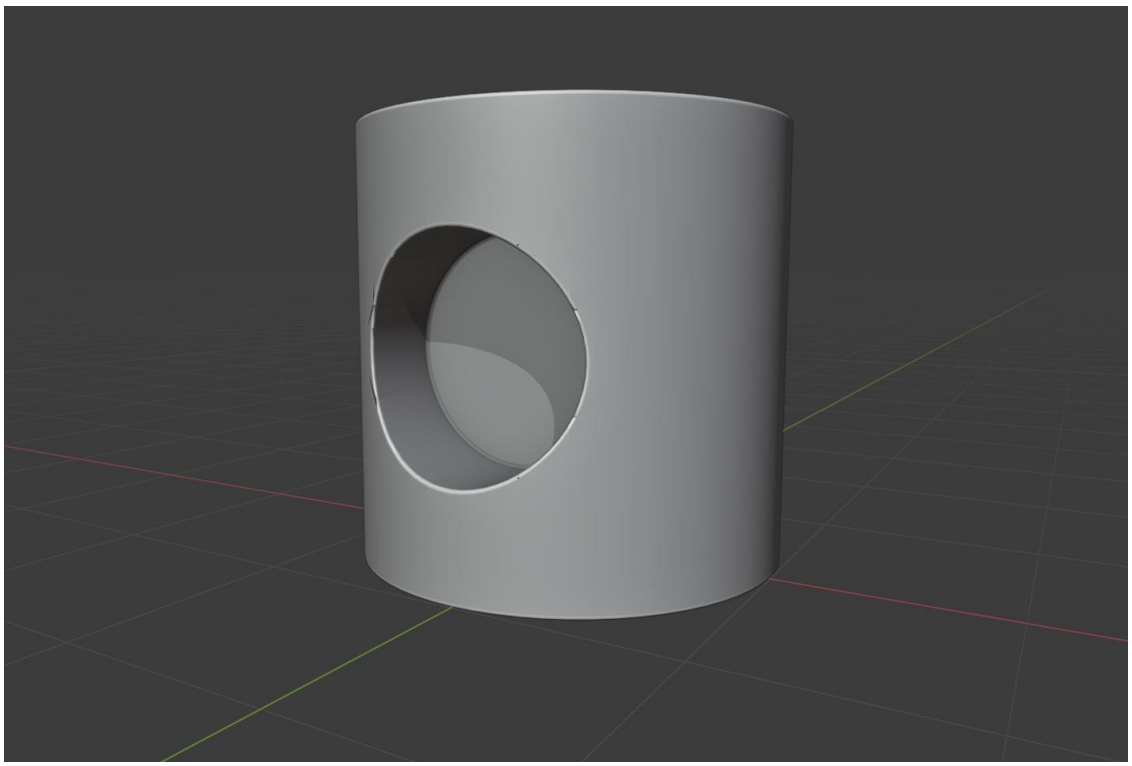
Obviously, we have some bad shading here. But you already know how to fix this. We just need the geometry to be denser so the shading can't stretch as far!

This is how it looks right now:



In the above photo, the bad shading stretches up to the top. Ew! We need to limit how far it can stretch, so of course we'll simply make the geometry denser.





As you can see, it's much cleaner this way!

But the *real* problem, which is exactly what this section is about, is about cleaning these Booleans.

The shading is near perfect, but we still have remnant artifacts around the bool. These need to be cleaned.

Let me stop you in your tracks – go to this link, right now, and grab a copy of our Topology Handbook for Blender – **it's free** and is literally what this section is all about, but it is all videos!

It will be great supplementary material for you after reading this section.

Seriously, if you don't go grab that copy it's like lighting your rent money on fire; you invested time and energy into making that money. The same applies to your studies – don't throw away valuable time when your solution is right in front of your eyes!

After you've grabbed that "handbook" (actually a series of videos), you can either watch those videos and then read this section or finish this section first. It doesn't really matter. If you need a break from reading, then go watch those videos now.

Ok, so fixing those artifacts. We need to simply slide those vertices outside of the realm of our bevel.

The reason artifacts occur is due to geometry overlaps, 100% of the time.

There is literally no other reason for them to occur if there are no geometry overlaps.

People get scared when they see artifacts – sometimes I chuckle because I know they were taught some tedious or meticulous process to fixing it. It's really easy though.

Slide the vertices away from the bevel. That's it.

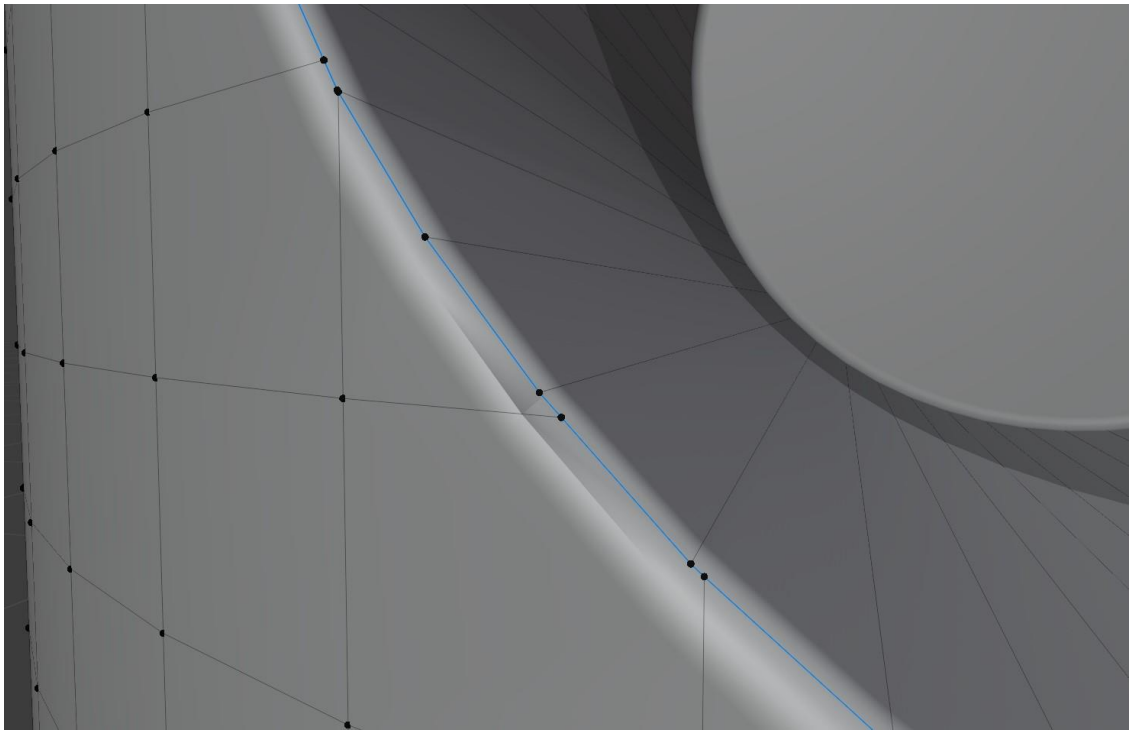
Now the vertices aren't intersecting with the bevel.

All done!

You may have some "stray" edges after sliding – just dissolve them with CTRL+X.

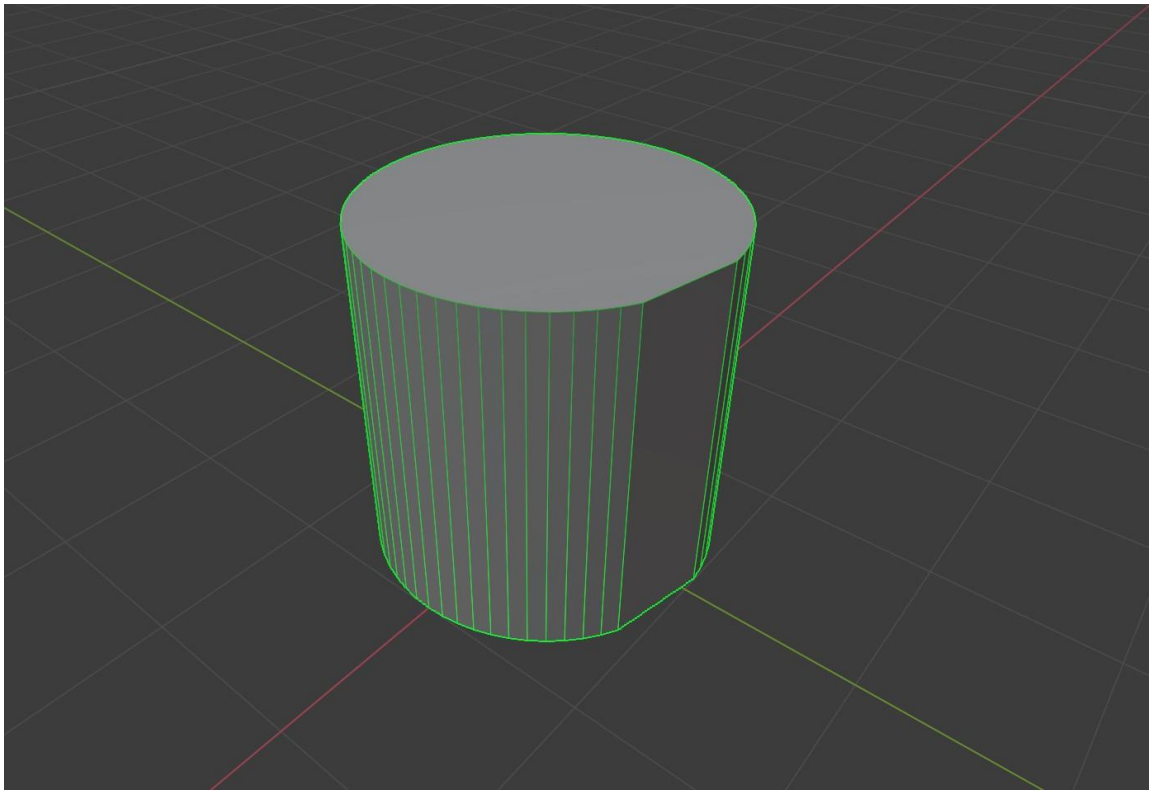
You might have two close-by vertices here and there, which you can either slide away like before, or in the following case, simply merge them together.

You can merge with the auto-merge tool, or simply select both vertices and press the ‘M’ key.



One trick that most people tend to overlook is the concept of curvature.

If you take a few edges of a cylinder and dissolve them or move them, what happens is the curvature gets disturbed. It doesn't look truly round anymore. You want to be careful to not move the vertices holding the curvature too far away. Here's an example of what I mean by the curvature getting "disturbed".



This is discussed more in-depth in the Topology Handbook I linked you. I'd *highly* suggest watching those videos to get a full understanding. Text can only do so much justice.

What if we are working with Union Booleans instead (i.e. Booleans that fuse meshes together instead of removing the difference)?

The same exact idea applies, just move/merge overlapping vertices. I'll let you test this one for yourself.

Boolean cleanup is an art in and of itself. There is no way I can explain the intuition via text – you need to watch as we work. Once again, watch those Topology Handbook videos – they are literally free, made by us and are a wealth of information.

I think we've covered the fundamentals here, there isn't too much left to discuss. The concept is easy, but proper and clean implementation takes practice. Play with some Booleans 'till you get the hang of it!

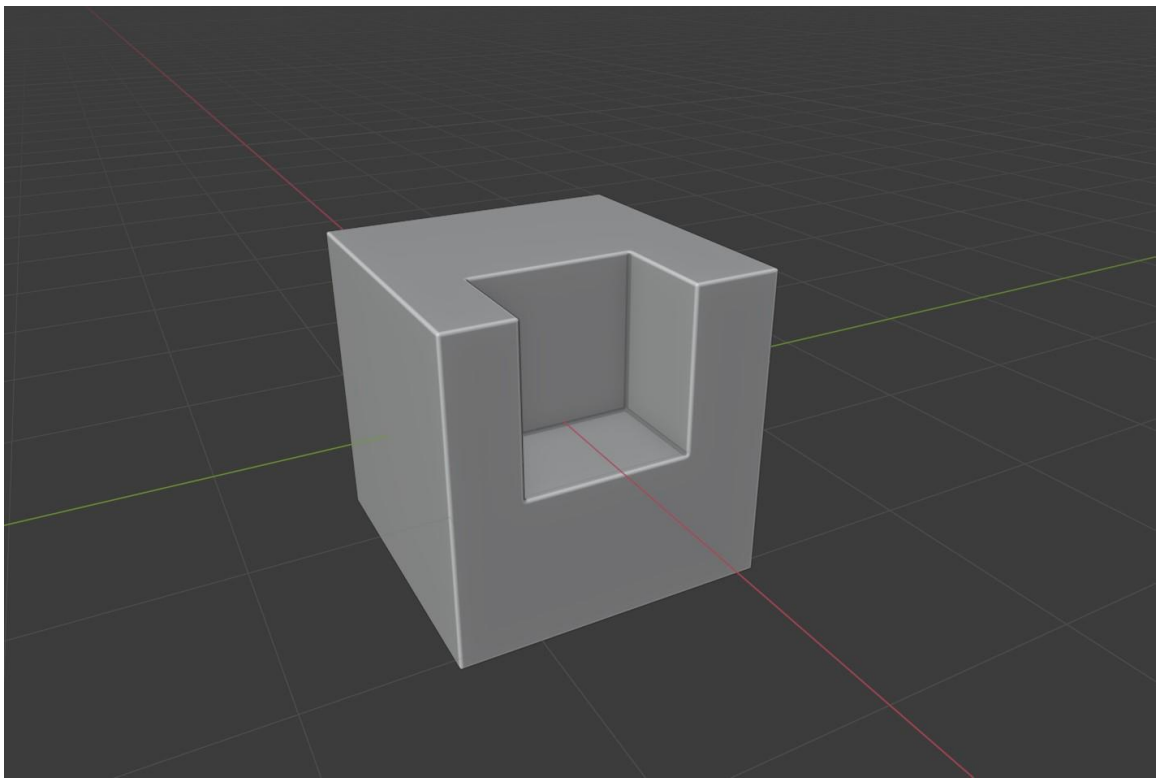
Multi-Level Bevels

Some people ask me “Josh, how can I get different sized bevels on my model?”

Usually we use one bevel modifier, which means the entire mesh gets that same size.

But we can use *multiple* stacks of bevels to get different sizes on our mesh.

Let’s take a look at a single bevel.

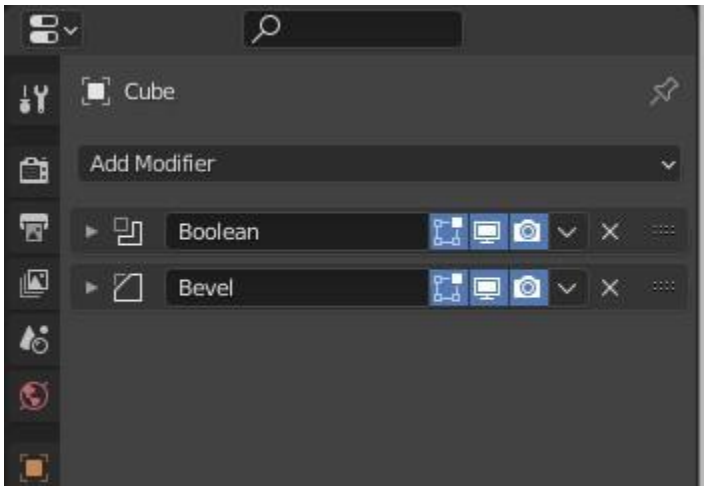


What if we wanted that cutter in the middle to have a different sized bevel?

Well, the most obvious solution would be to simply bevel it manually, but this would be a completely destructive workflow with no versatility.

Instead, what we’ll do is use a *separate* bevel modifier.

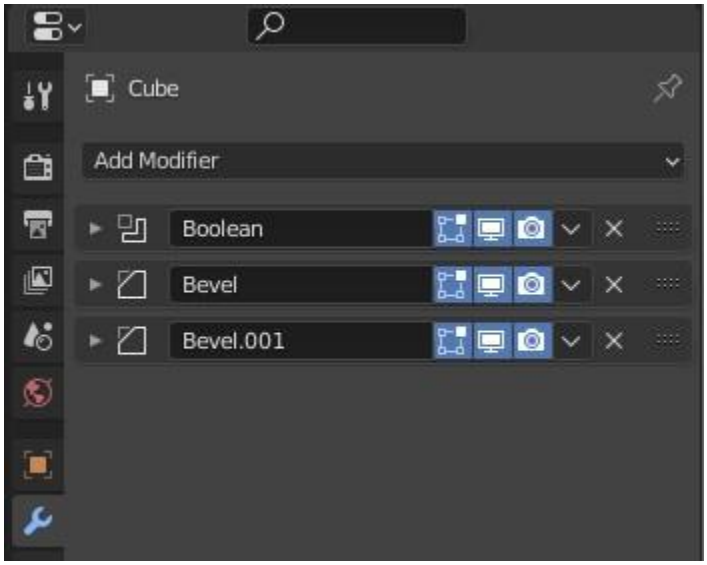
Currently, this is how our modifier stack looks:



All we have is our single boolean and bevel.

We want to use a *second* bevel modifier to make a separate bevel width.

However, if we simply add one right off the bat, notice how the modifier stack looks now:



We have two bevels back to back.

This is a problem because absolutely nothing changes in the viewport.

We need to pay careful attention to the *order* of the modifier stack.

Notice how our boolean comes first, and *then* the bevel.

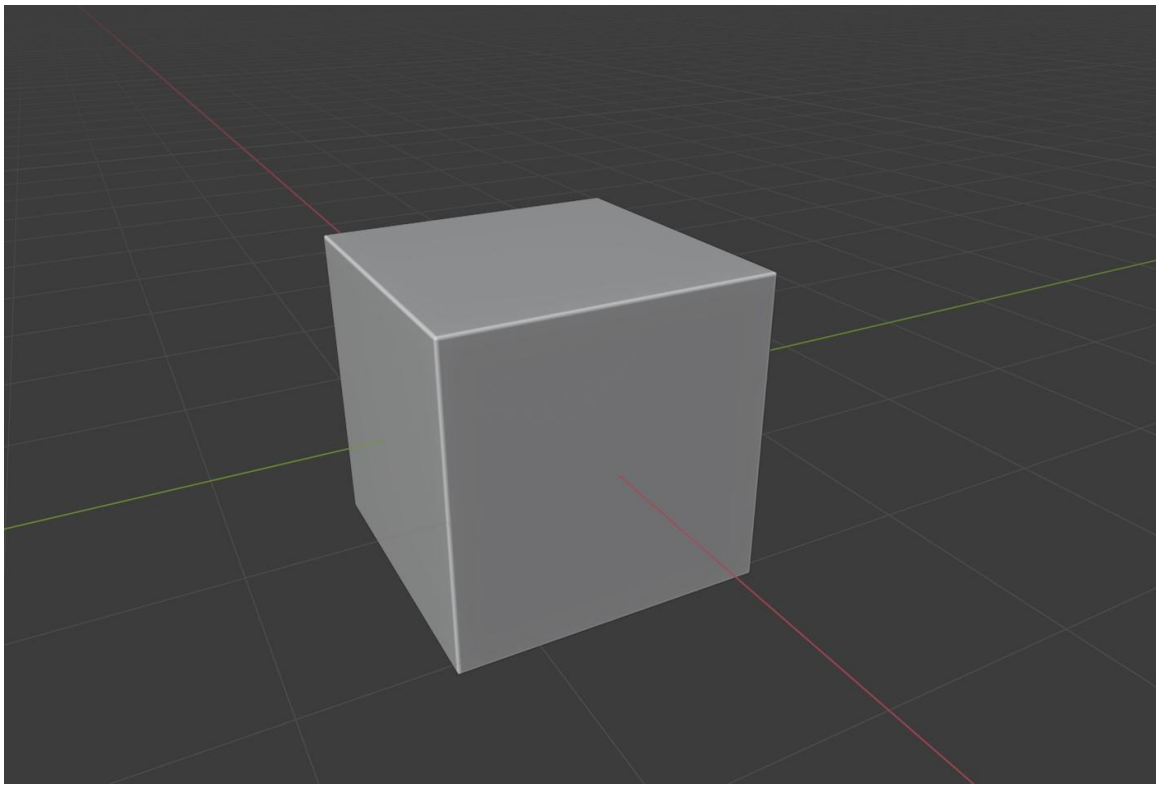
This means that the boolean operation has finished before applying the bevel.

So what happens is since Blender has already executed the boolean command, and the bevel comes *after* it, the bevel gets applied to the entire mesh.

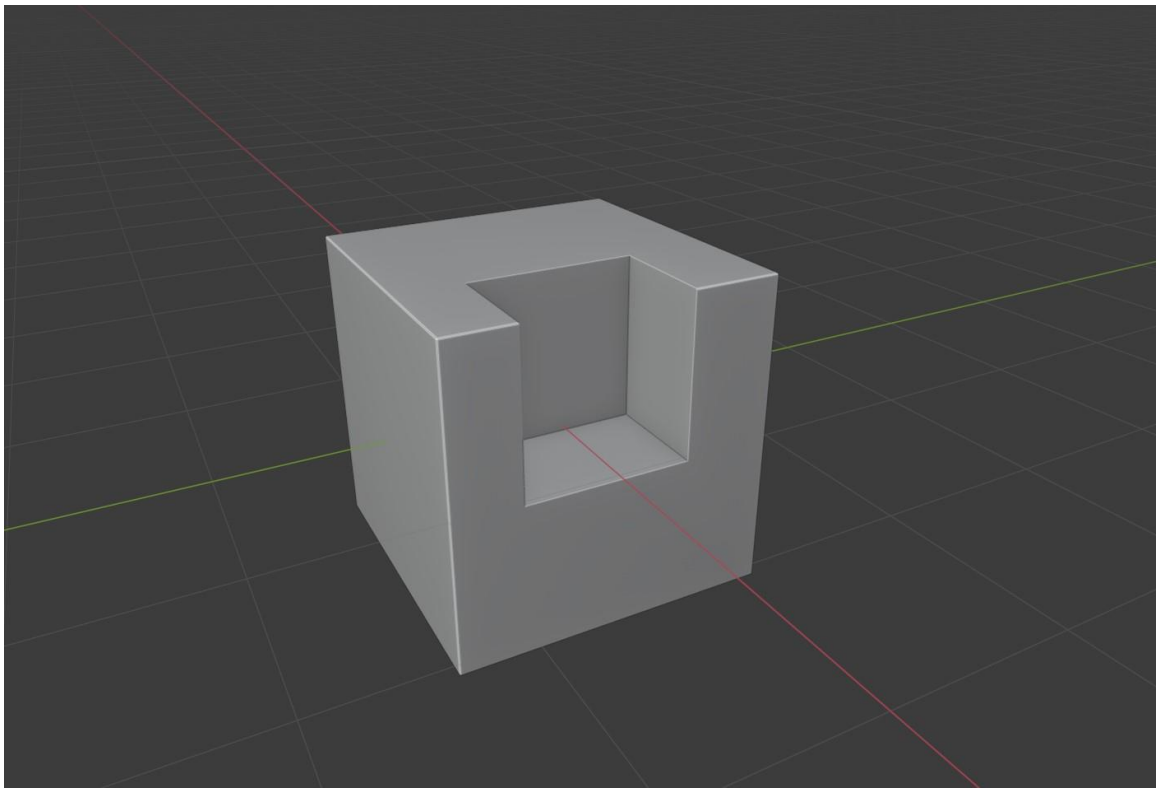
Instead, we want to swap the order.

We want to run the bevel *first*, and then the boolean.

This means that the bevel first applies to the entire cube, like this:

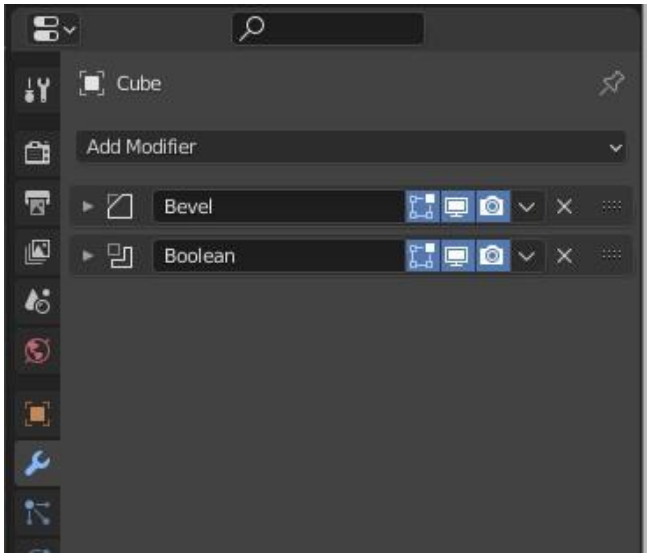


And *then* the boolean comes after it:



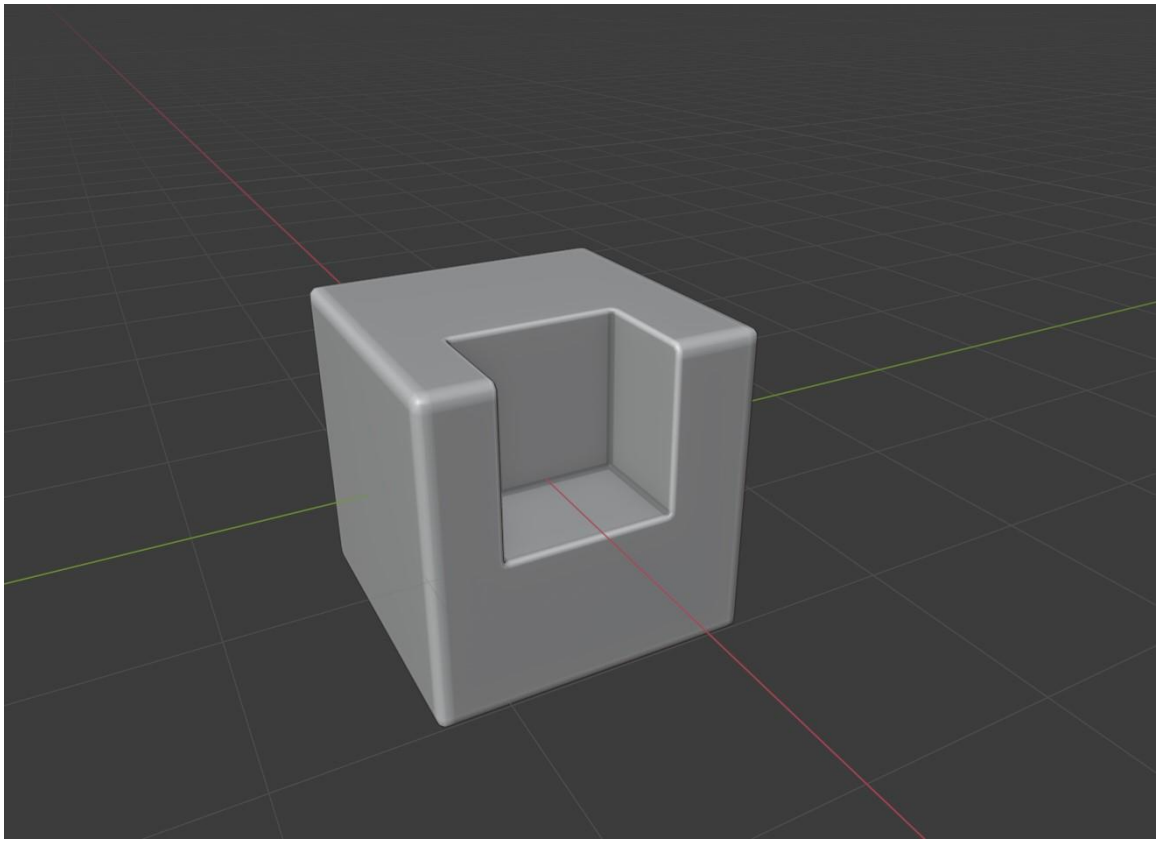
Notice how the boolean cut has no bevels around the edges, because the bevel operation was already completed *before* the cut was applied.

So now your modifier stack should look something like this:



Bevel first, boolean second.

Now we can add a second bevel modifier which will affect the boolean above it.



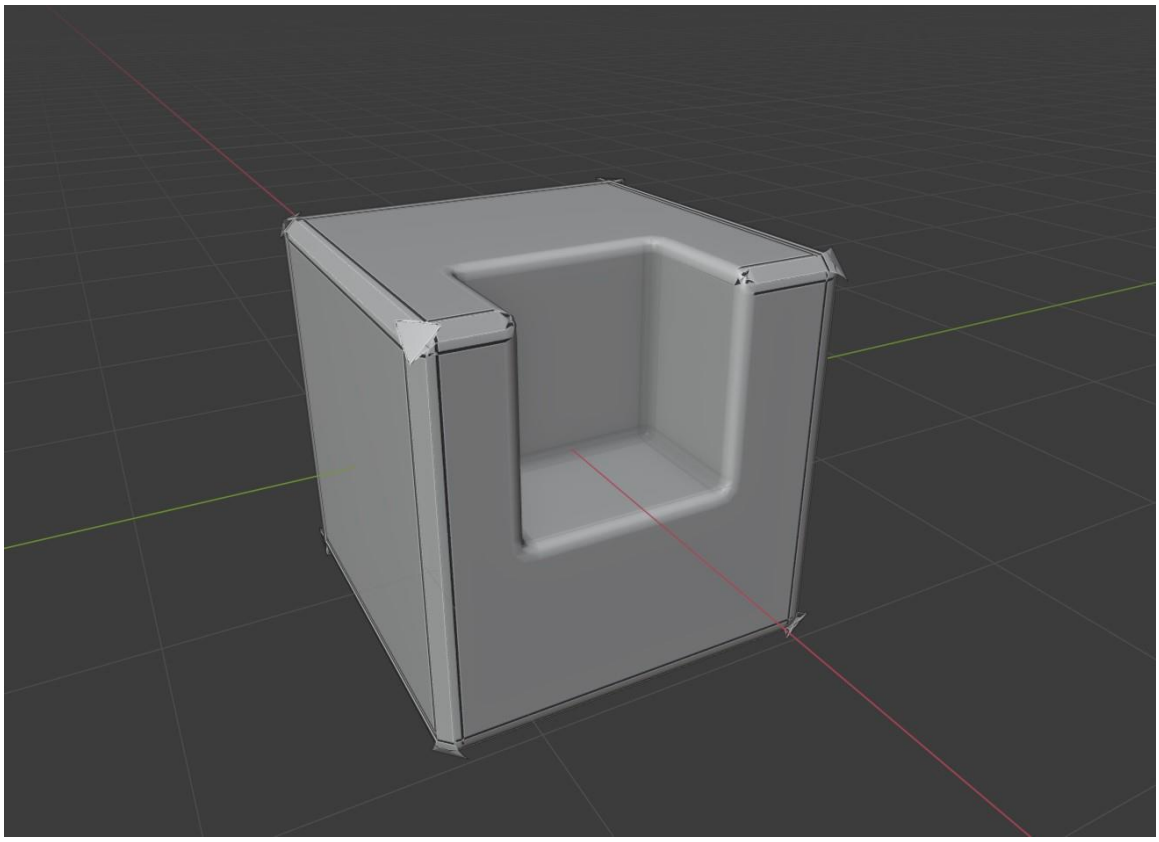
I ended up making the first bevel bigger, and the second smaller. It just worked better that way for demonstration purposes.

There is one issue with this method though. It doesn't happen here, but with more complex angles it can get tricky. Since the modifier stack works in chronological order, the second bevel is affecting *everything* above it, including the first bevel.

This means you could potentially get double bevels, causing immediate overlaps. It could get crazy.

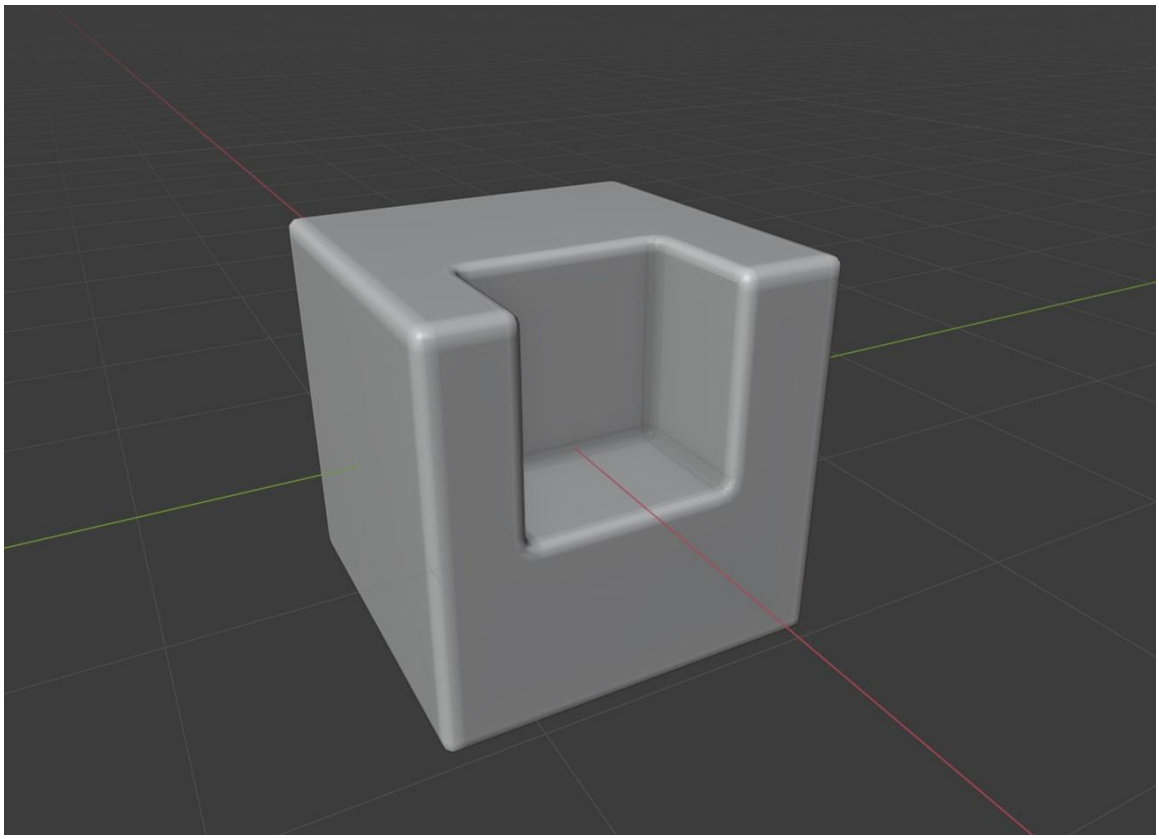
I'd encourage you to look back over the 'Bevel Limit Methods – Angle' part of the book to refresh your memory with how angles work, because this comes into play here.

Since the second bevel also affects the first bevel, this means we need to make the angle value of the second bevel *higher* than the first, otherwise the same angle values will catch simultaneously and cause immediate overlaps like this:



I made the bevel a bit higher to exaggerate the effect, but this is precisely what happens. You get two bevels running at once, immediately overlapping.

If we set the second bevel to a higher angle value, then it won't catch the original bevel and should only affect the boolean cut.



In this case, the angle value for the first bevel is 30, and the second is 60.

(The one with the overlaps were both set to 30)

This is a common pitfall and a lot of people just give up here because it seems too complex. It really just comes down to the geometrical angles between two edges and managing those values properly when running multiple bevels.

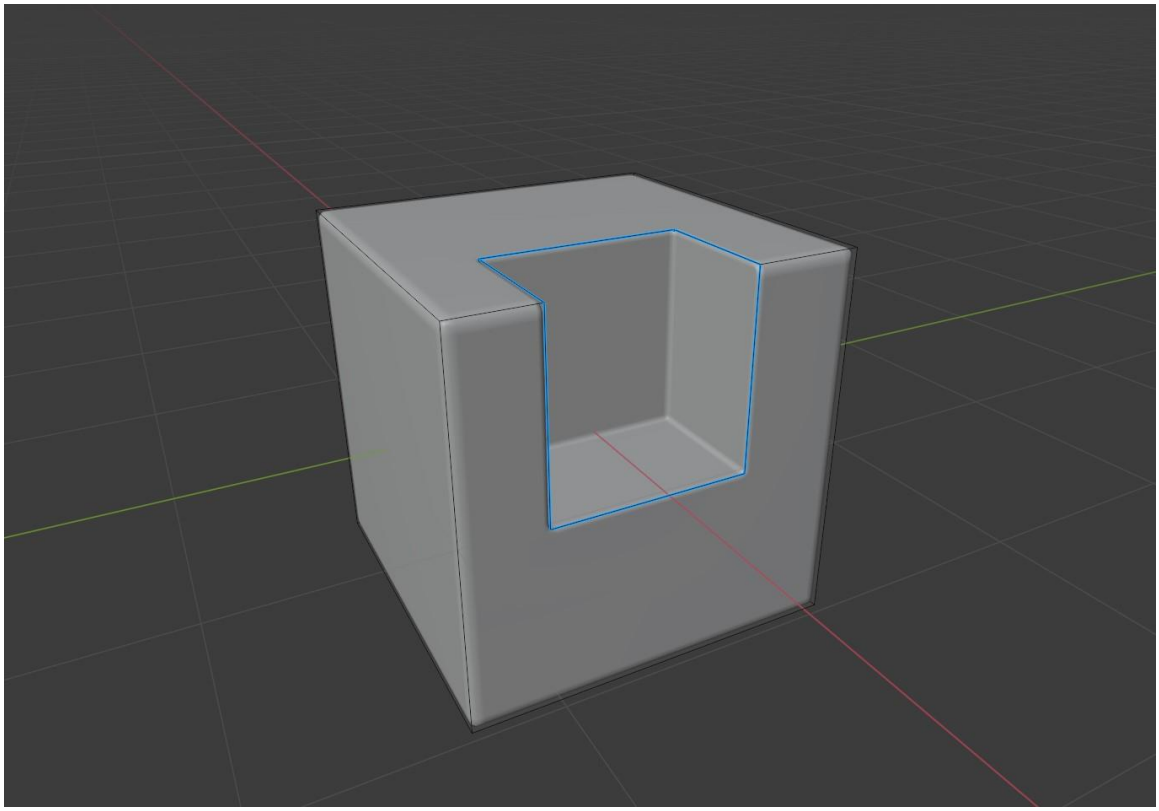
I will admit, this took me a bit to wrap my head around when I was first learning, but if you understood what I just explained here then you're probably way smarter than me and will flourish as an artist.

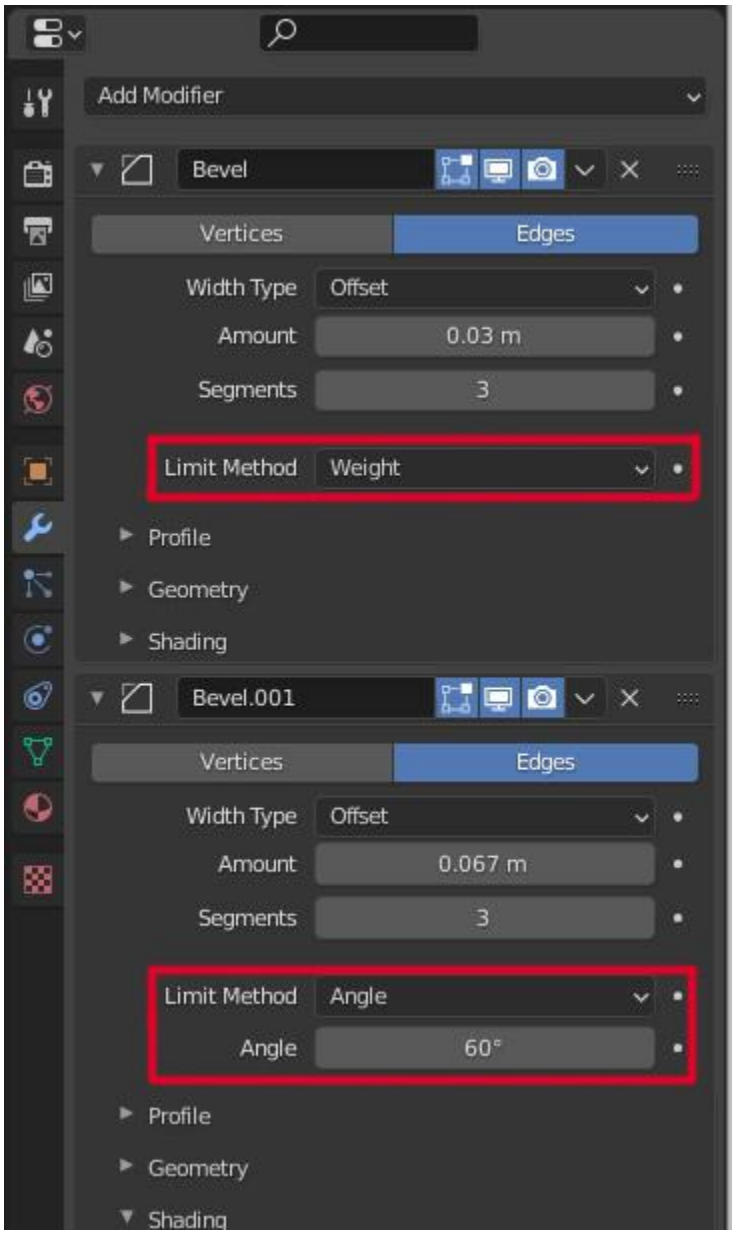
If not, welcome to the club of tricky learning, may I get you a coffee? 😉

Jokes aside though, it is a bit tricky, so don't feel bad if you don't understand it the first time. You will eventually once you play with it.

Another strategy would be to combine Bevel Angles with Bevel weights (remember, bevel weights allow us to manually select edges which are to be beveled).

Repeating the last example, I could simply add a bevel weight to the interior cutters' edges, and an additional bevel modifier set to angle.





The issue with this strategy is it's semi-destructive. Any bevel you're running bevel weight on means you have to apply the boolean in order to manually select those edges.

It comes down to a balancing act between the versatility of bevel weights and the potential obstruction of bevel angles.

Feel free to apply Booleans as needed, just keep a backup in case you want to reverse it once you go this destructive route.

Dicing Booleans

We've covered a LOT – and I mean a lot.

See why I made this book now – we're covering all the essential stuff that isn't really found in a single video anywhere? I really wish I could explain the struggle I went through when I was learning. Months of repeatedly watching videos and practicing until things started to click.

Every. Single. Day.

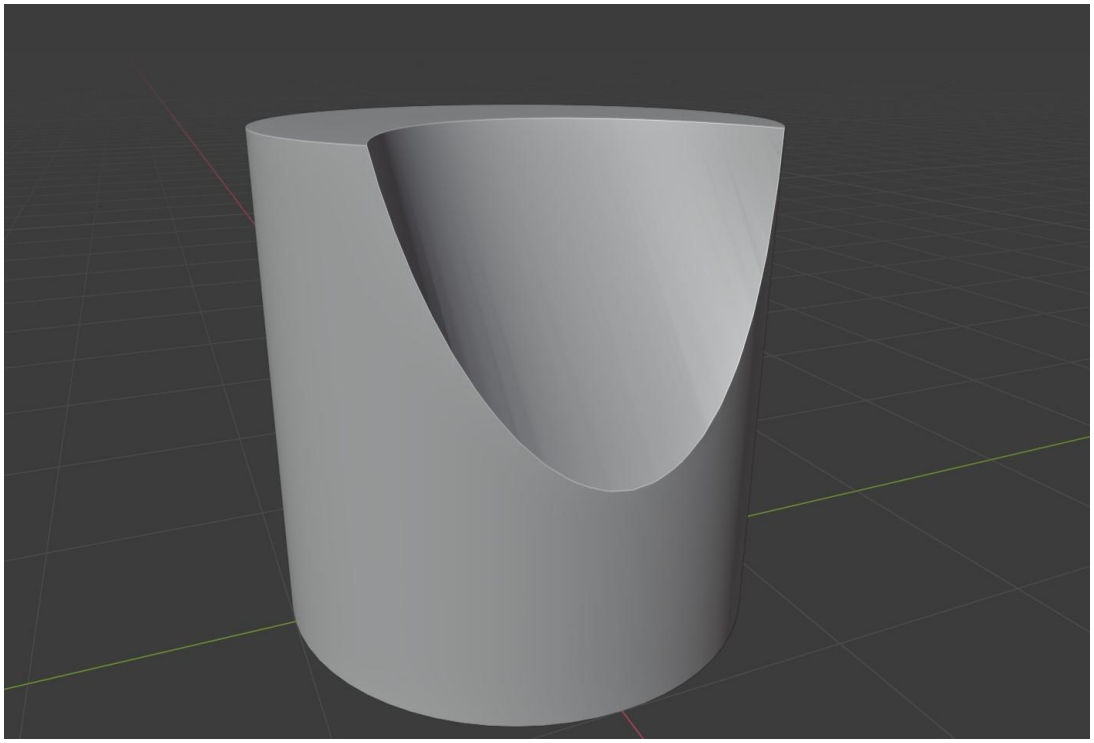
I don't want people to have this hassle because it is honestly a waste of time.

With that being said, let's get back on topic.

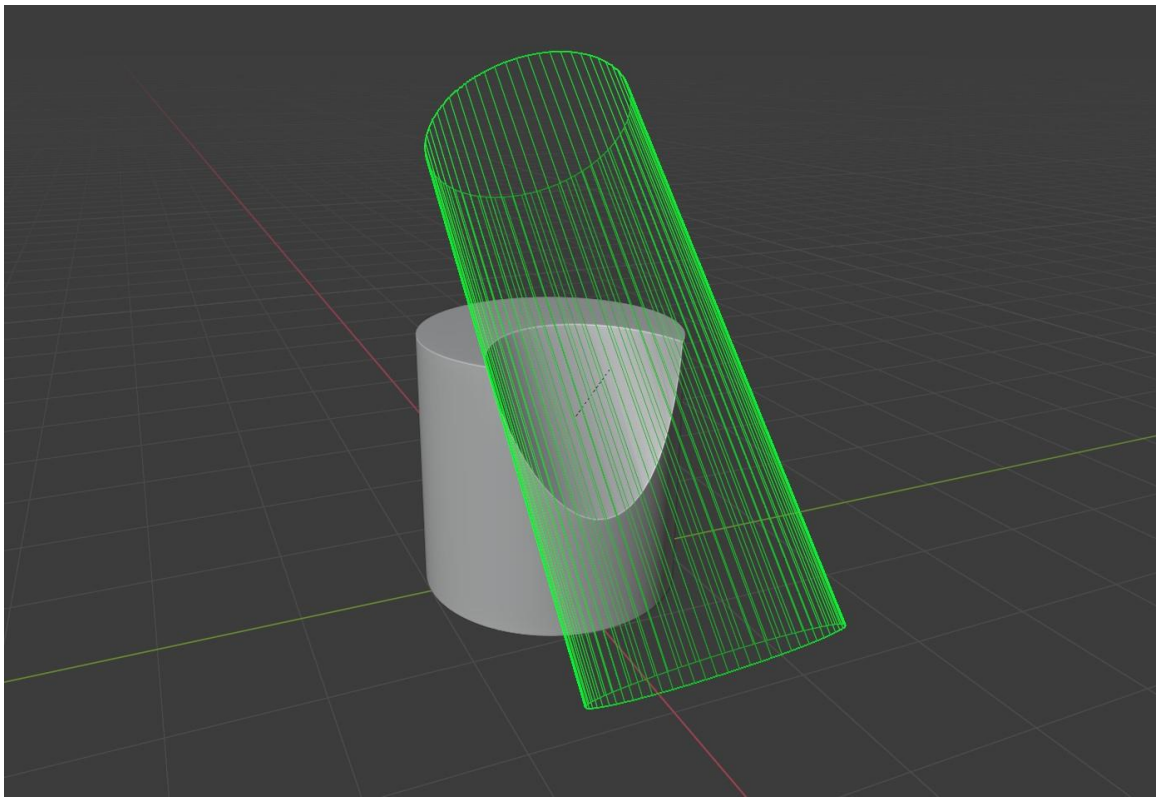
Dicing. Dicing literally just means adding loop cuts through an object.

Dicing *Booleans*, however, is a very specific technique.

Check this out:



I have a cylindrical boolean running like so:

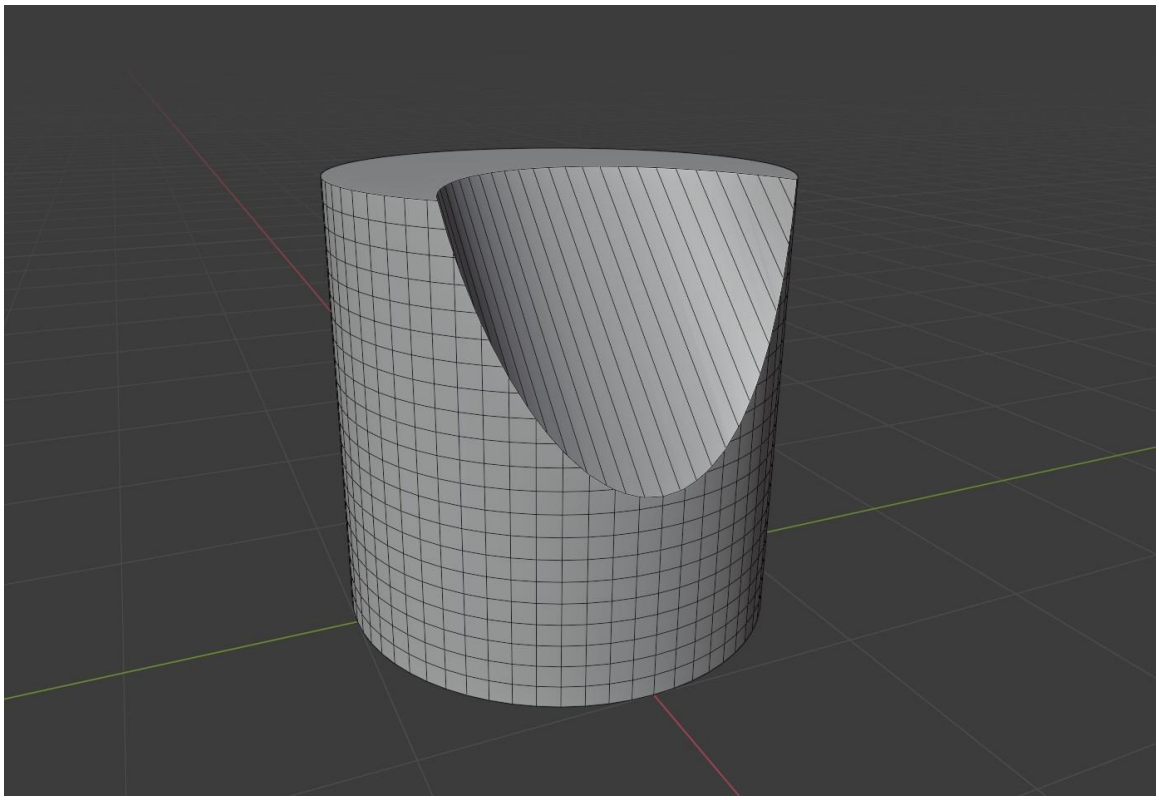


In the first picture you can see the shading artifacts.

Gross!

You might think, well why don't we just run some loops on the base cylinder?

Well I did that already (check the next picture).

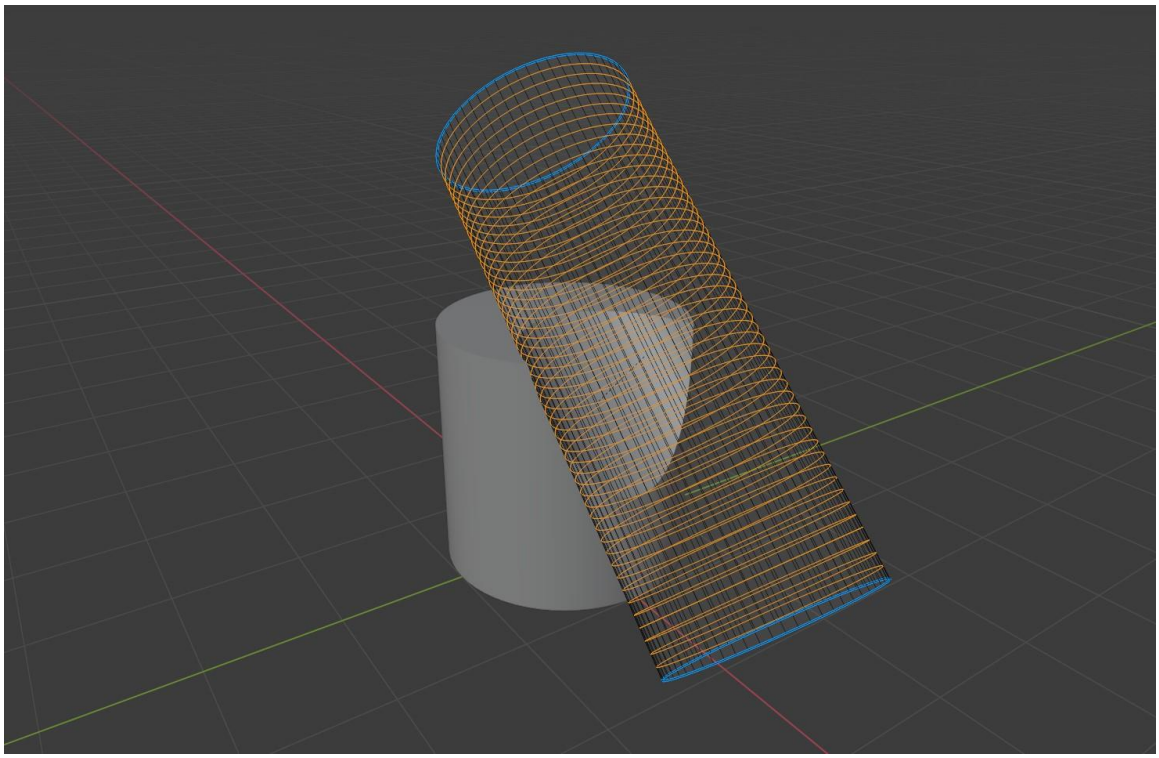


How in the world do we possibly get the loops to transfer to the cut area?

Well, if the Boolean is applied, this isn't so easy. We'll probably be stuck and have to use some meticulous combination of knife projects and excess cleanup.

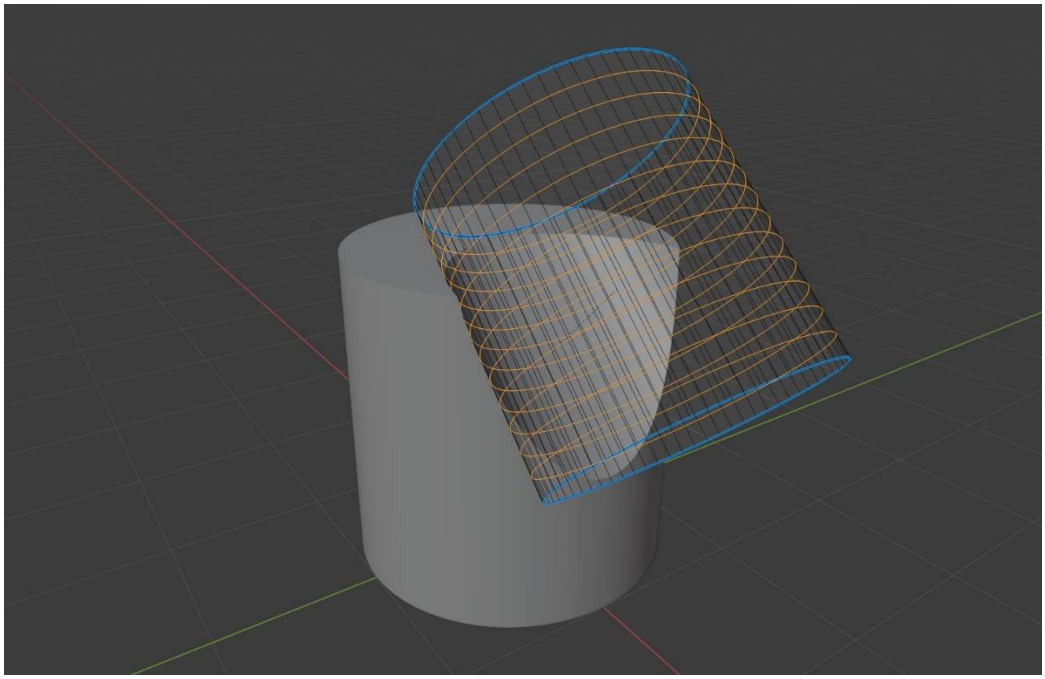
Don't worry about that though, let me instead show you the right way to dice the Boolean area.

Simply run some loop cuts through the cutter!



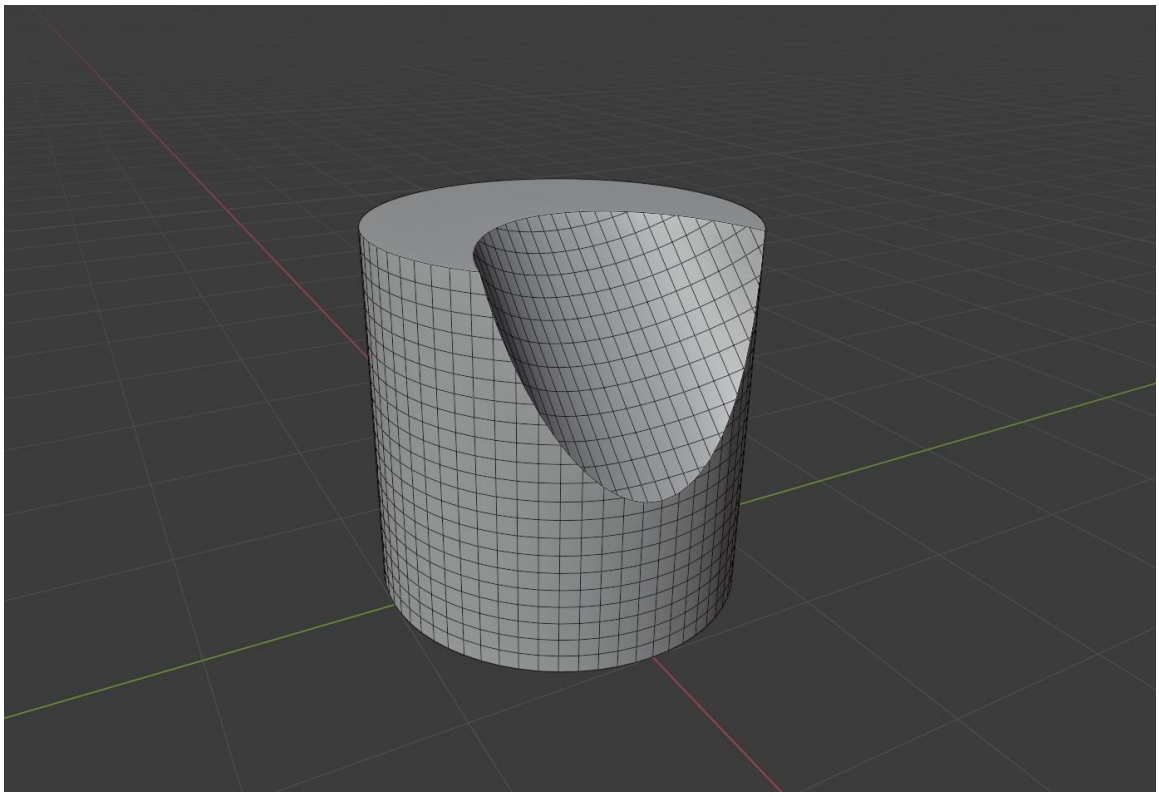
So essentially, just go into edit mode for the cutter and press ‘CTRL+R’ to add some loop cuts and scroll up to increase the count.

You might want to scale the cutter down and make it just big enough so it only fits the area you wish to cut.



Reason being is because if it is scaled beyond the area you need you'll just spend more time rolling the scroll wheel adding loop cuts that fit.

Save time where you can!



As a result, we now have loops running *exactly* where we need them!

Whenever you run loop cuts on a cutter object, all of the geometry on the cutter will be transferred to the base mesh.

I call this "forcing geometry".

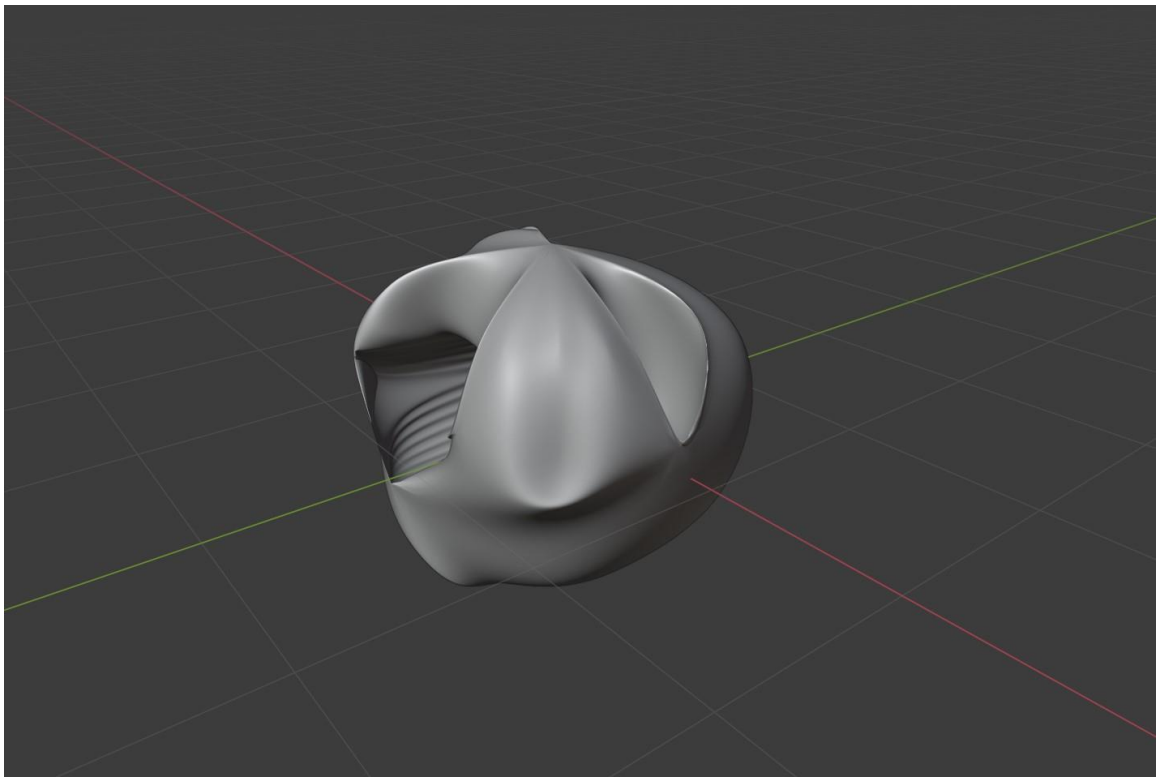
In conclusion, if you have a weird cut like shown, you might need to dice the cutter object.

Subd with Booleans

Subdivision surfaces with Booleans. Ah, the days I thought this was the worst thing ever. I almost felt brainwashed, being fed false information.

The truth is that it is all about *how* you use them together.

Obviously, if you have crazy ngons everywhere and try to run subd's, you'll probably get some mess like this:



It'll collapse over itself, have awful artifacts and look like some sort of alien from outer space.

Maybe you can get away with it if you're into glitch art, but that's about it.

So the rules of no subd's on ngons still hold, and always will.

But no one talks about running Booleans *after* the subdivision surface modifier.

Remember the modifier stack order? Blender executes operations top-to-bottom. This means that if the subsurf is first in the stack, any Booleans after it won't be affected by the subd!

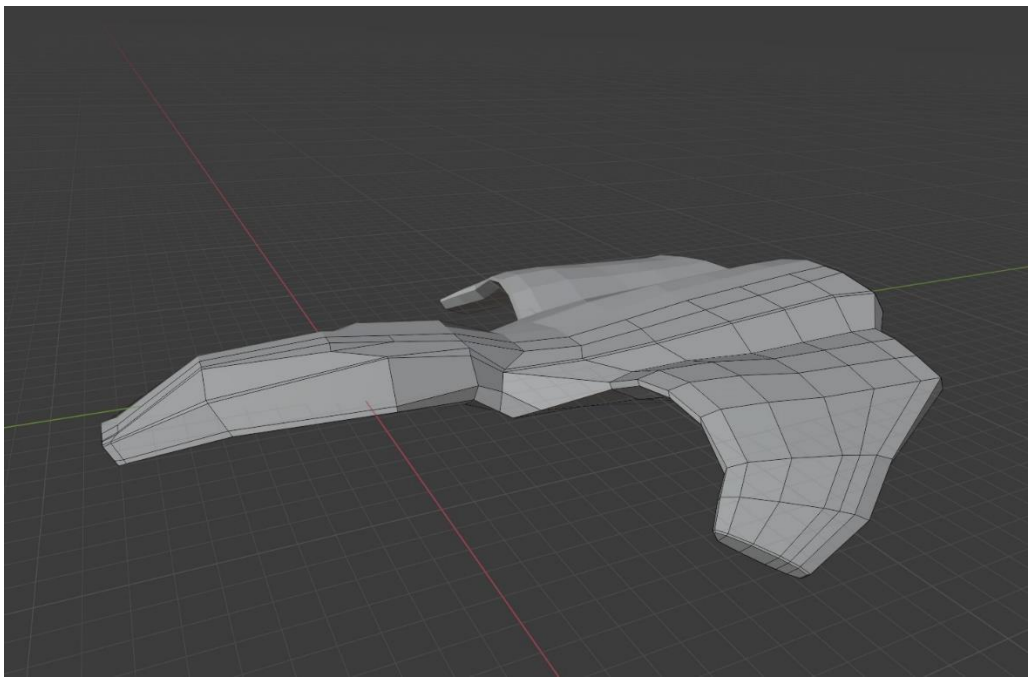
Let's do a simple example.

I didn't really feel like creating another boring example, so I am using the model from my partner Ryuu's organic hard-surface modeling tutorial for demonstration.



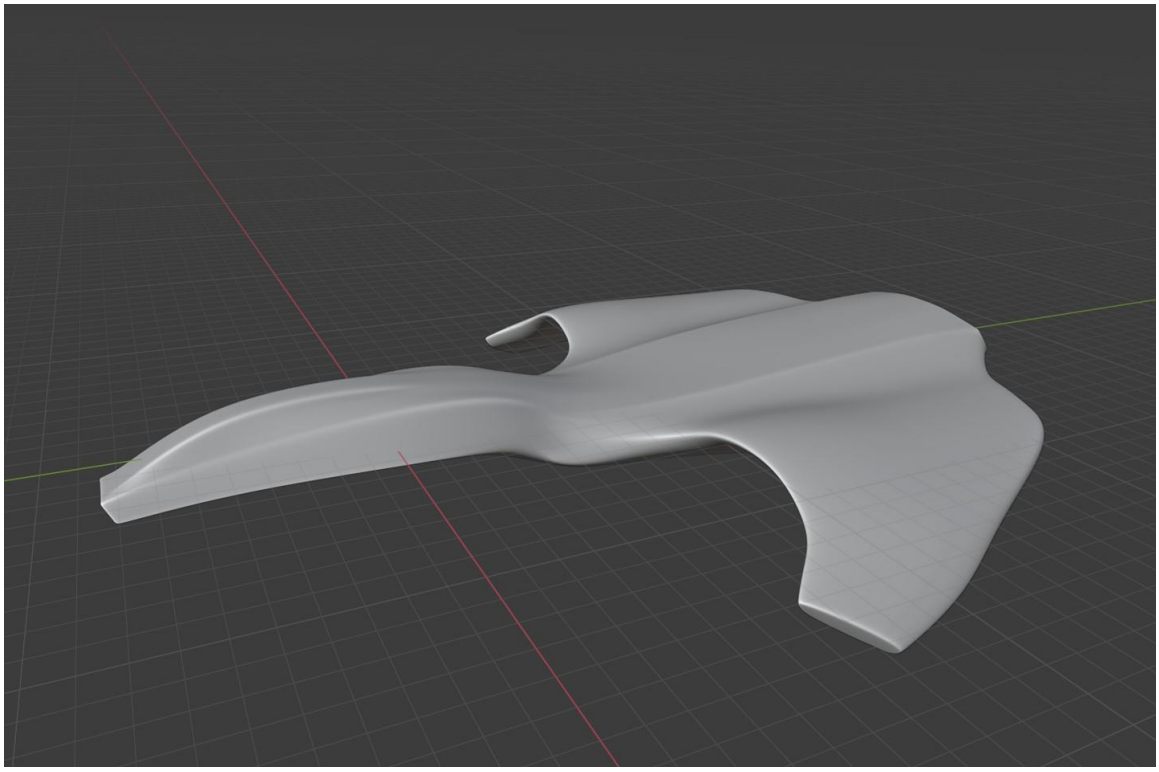
Let's break down how this thing was actually made.

This is the object without subd applied:



As you can see, it is (mostly) all quads and loop cuts. A beginner could probably block something similar out in a few minutes, using proportional editing, scaling and rotating.

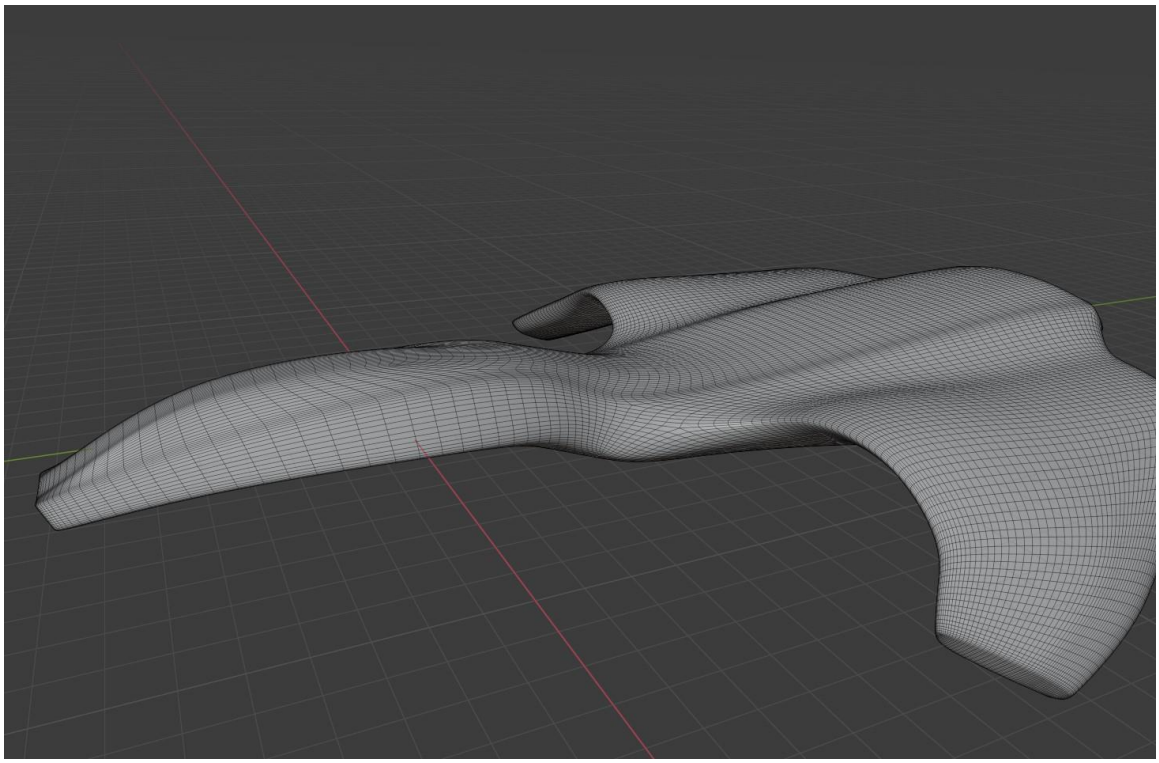
After we apply subd, it looks much more organic and smoother:



Obviously, there are no Booleans in this object. Just quads and subsurf.

Let's assume this was our final form, and we were ready to start cutting and running Booleans.

If we apply the subdivision surface modifier, it will be nice and smooth.



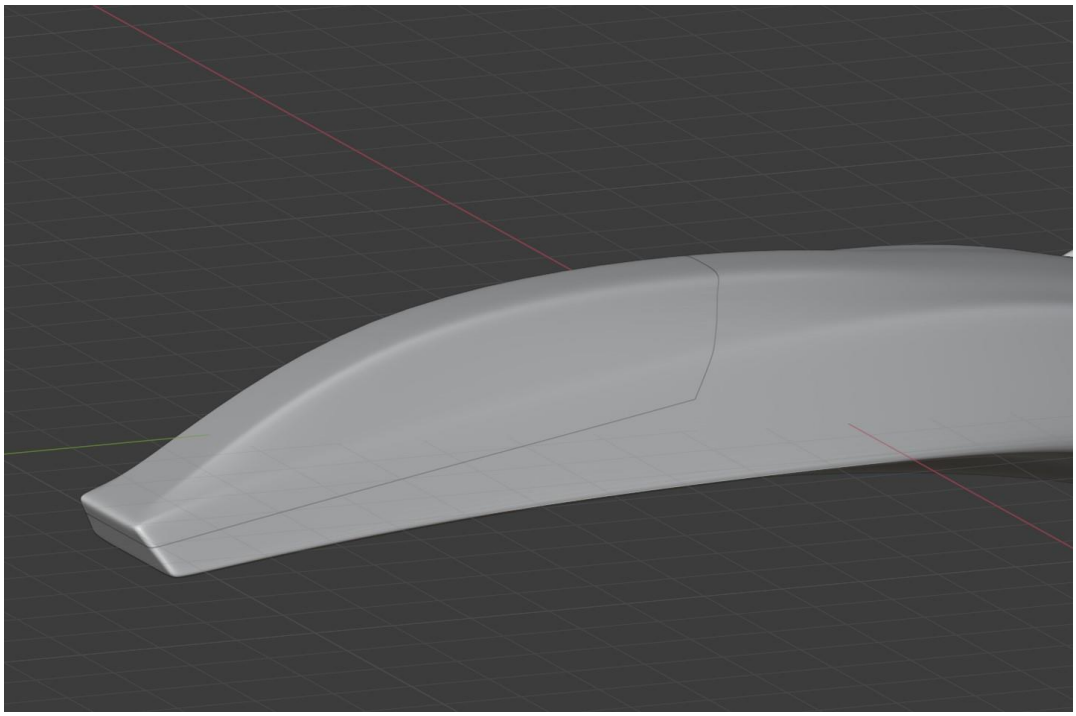
Dense and full of quads. Lovely!

At this point we can start running Booleans.

It is important to not even think about the subd at this point – it is done and over with since we applied it.

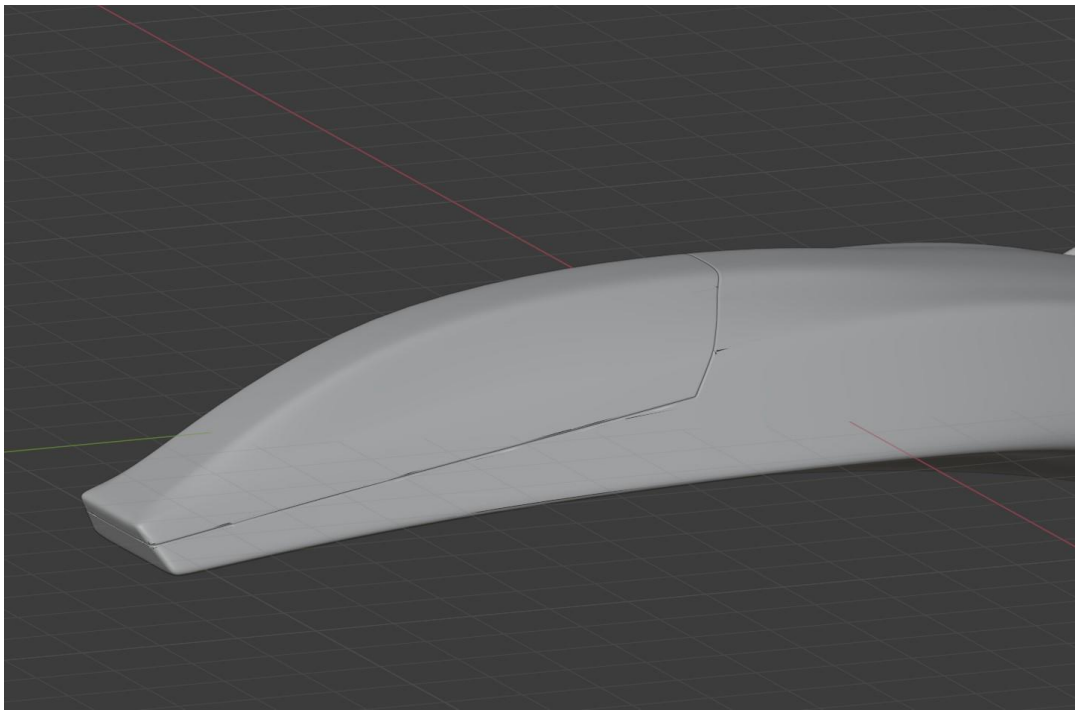
This looks scary because it's organic and smooth now, but the *exact* same methods presented in this book apply when running Bools on this surface.

Let's run a small slice on the front:



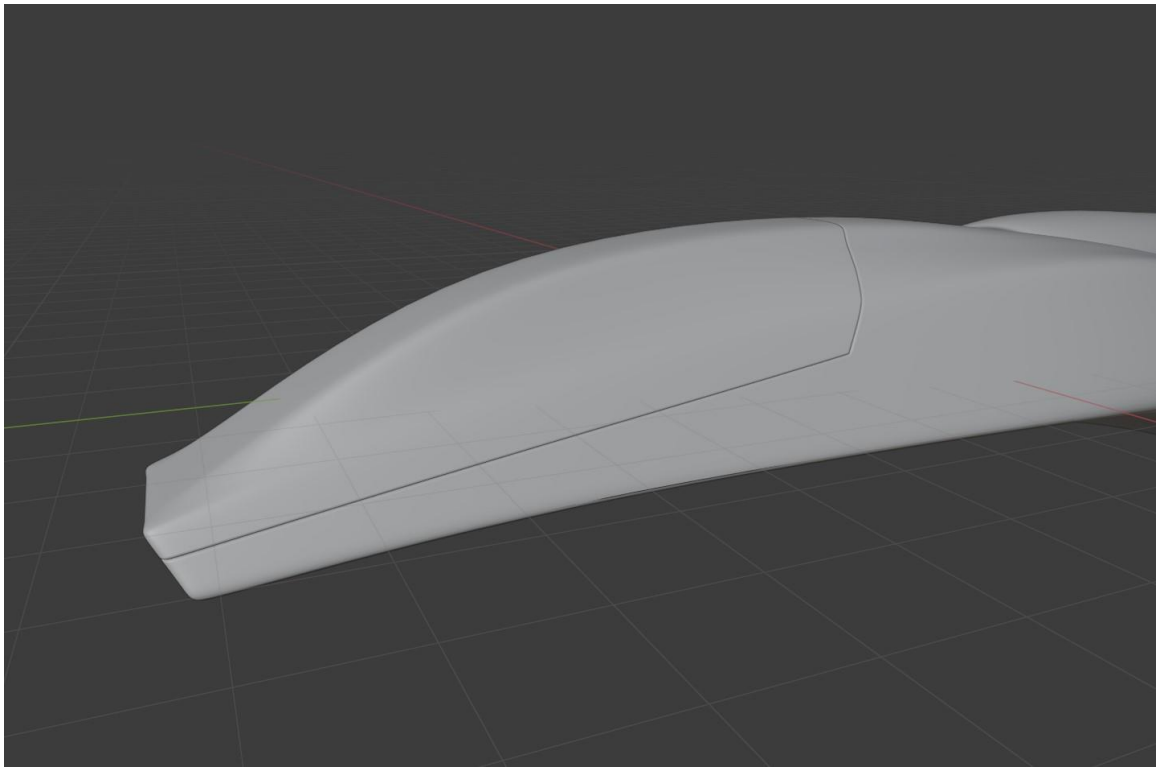
See? No problem at all. The subsurf isn't actively running anymore since we applied it, so this boolean is not affected by it.

Only thing we need now is a bevel on both pieces.



Obviously, since the geometry is so dense as a result of the subdivision surface modifier, bevel overlaps are going to occur almost immediately.

You can easily use the techniques taught in this book to fix them (apply your Booleans and start merging and sliding vertices until it's clean).



When people say "*never ever* use subds with Booleans", they are only speaking half the truth. What they actually mean is "*never ever* use subds *after* Booleans" because that is when it becomes a recipe for disaster.

Subds are actually perfect complements to Booleans, so long as the Booleans come *after* the subd modifier.

So, my friends, slice away, but just check your modifier stack!

Hard Ops/Boxcutter

If you've been following my content for any period of time, you'll know I'm a HUGE proponent of Hard Ops/Boxcutter.

For those of you not familiar, Hard Ops/Boxcutter are two fundamental hard-surface modeling addons for Blender. The bundle costs $40.

I was hesitant to buy it back in the day, not necessarily due to price but due to having to learn a completely different workflow.

Now, I couldn't live without them.

Lots of people like to complain about paid addons for free software like Blender. It makes no sense. Would you rather forge a hammer to nail in an object, or just buy the hammer and get on with your life? These addons have saved me literally *thousands* of hours, which was worth my $40 one hundred-fold.

This product was made by Jerry Perkins (aka masterxeon1001) and his development team TeamC.

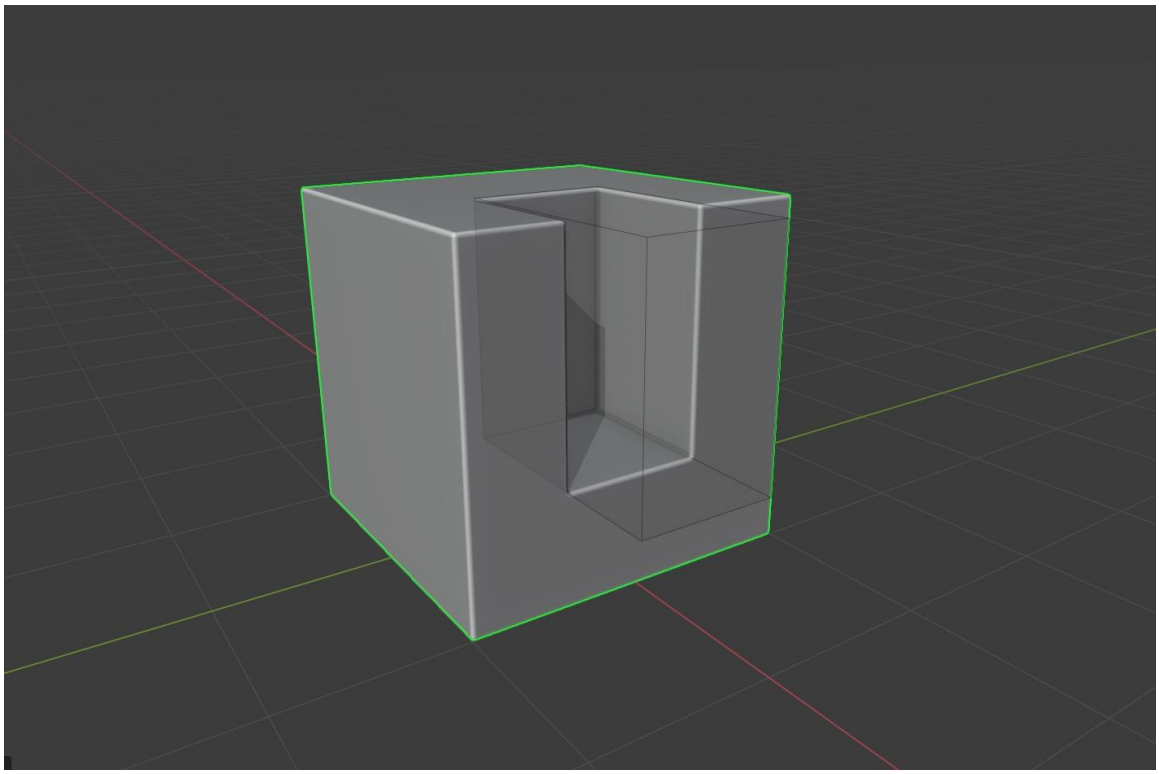
Let's show how you can use it to expedite your workflow.

We've already discussed how to use Booleans in your workflow.

But using custom objects to cut into other objects is annoying. I'd rather just click and drag to create a boolean cut.

Boxcutter allows you to do this.

You simply click, drag, and release. That's it.



Boolean cuts couldn't be easier. This is much more efficient than adding a cube, scaling it, positioning it, running a boolean modifier and having to go back and forth with this.

This is the gist of Boxcutter. It is obviously a lot more complex and would be impossible to teach the entire addon in this book.

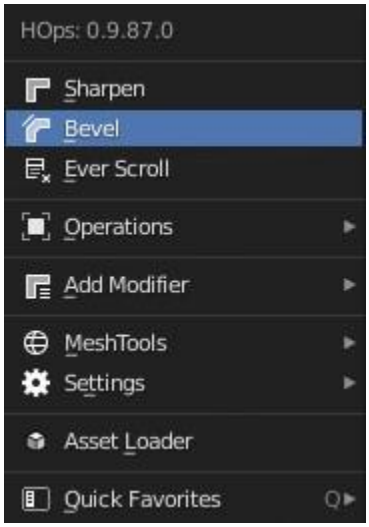
For that reason, check out Ryuu's YouTube tutorial series on using Boxcutter.

Hard Ops, on the other hand, is a modeling management system.

It can do everything that Blender can do, but the catch is it significantly reduces the number of necessary clicks to achieve a certain operation.

It might not seem like much, but in the long run you will be saving *millions* of clicks.

Hard Ops gives you quicker access to some of the more common menus and operations. Here's an example of the UI:



It looks simple, and it really is, but is tricky to get used to it in an intuitive workflow.

This is the base menu – there is a *lot* more functionality. This is just the heart of it.

For example, the bevel option here allows you to click and be done. No need to adjust the limit method, segment count, clamp overlap or harden normal settings. All of it is done in 1 click and is completely customizable depending on what you need.

Once again, check out Ryuu's tutorial series on using Hard Ops if you'd like to learn it start to finish.

This addon is constantly being updated, so some information may be out of date, but the general workflow remains the same.

I'd also suggest reading the documentation beginning to end and watching masterxeon1001's videos (he's the creator). This is how I learned it originally.

With all this being said, and as much as I love these addons, learning the vanilla (default) Blender workflow is essential.

This is because these tools are meant to be a modeling assistant and speed up your workflow, but if you run into problems and don't know how to solve them because you never learned the default Blender workflow, things will get messy and you'll get lost not knowing how to fix things.

I don't want to hammer addons too much, just wanted to mention them. They are a must-have for my workflow and extremely cheap compared to premium 3D modeling softwares like Maya, so there isn't a massive gatekeeper for newcomers wanting to learn 3D.

Also, support the creators, they are what make Blender great. Without the contributors, Blender would not be where it is today.

Other Modeling addons

Here is a list of some of the other modeling addons that I use daily in my workflow. Some are free, some are paid, but all are fantastic.

1. Hard Ops/Boxcutter
2. Machin3 Tools (free)
3. Decal Machine
4. Mesh Machine
5. Mira Tools (free)
6. Power Save (free)
7. Kit Ops 2 (free version)
8. Kit Ops 2 (paid version)
9. Cablerator
10. Definitely Eevee Materials system (for materials, works in Cycles)
11. Instamat (for materials)
12. Smudgr pro (for materials)
13. Quad remesher (don't really need it, it's good for remeshing pipes though)

Here is the tier of addons I'd purchase, from most important to least important (although all are *incredibly* important).

1. Hard Ops/Boxcutter bundle. Buy the bundle, they are essentially useless without complementing each other.
2. Decal Machine
3. Mesh Machine
4. Kit Ops 2 (paid has more features available, but free will do)
5. Cablerator
6. Definitely Eevee Materials System
7. Quad Remesher
8. Instamat (for materials/rendering)
9. Smudgr Pro (for materials/rendering)

Conclusion

We really hope this e-book provided you some sort of value.

Undoubtedly, you probably still have more questions. You can't learn to ride a bike by reading how to ride one – you have to actually get on and fall a hundred times until it finally clicks.

Hard-Surface Modeling is no different.

However, we as BlenderBros try to make the process as seamless as possible.

Our YouTube channels are full of valuable content, all for free.

However, part of being able to provide all this value so frequently stems from your support as well - so thank you for it!

If you're looking to get a bit more hands-on with 3D and speed up the whole learning process, you'll also love our courses.

You can find some of our premium courses and our Patreon below.

https://www.patreon.com/theblenderbros

https://gumroad.com/theblenderbros

Since you made it to the end, here's a free link to one of our beginner courses – it's on us. It'll give you a taste of some of our other, more advanced courses, which we heavily suggest you enroll in to beat your competition and flourish in 3D.

https://gumroad.com/l/qwuLs/hardsurfaceebook

Thanks a bunch for checking out this book.

Should you need to contact us for consultations, course questions, business inquiries, or anything of the sort, you can reach us at info@blenderbros.com



Blender Bros

THE HARD-SURFACE E-BOOK

HTTPS://WWW.BLENDERBROS.COM/