Tutorial: Basics of Glass in 3D

By Mike Morrison

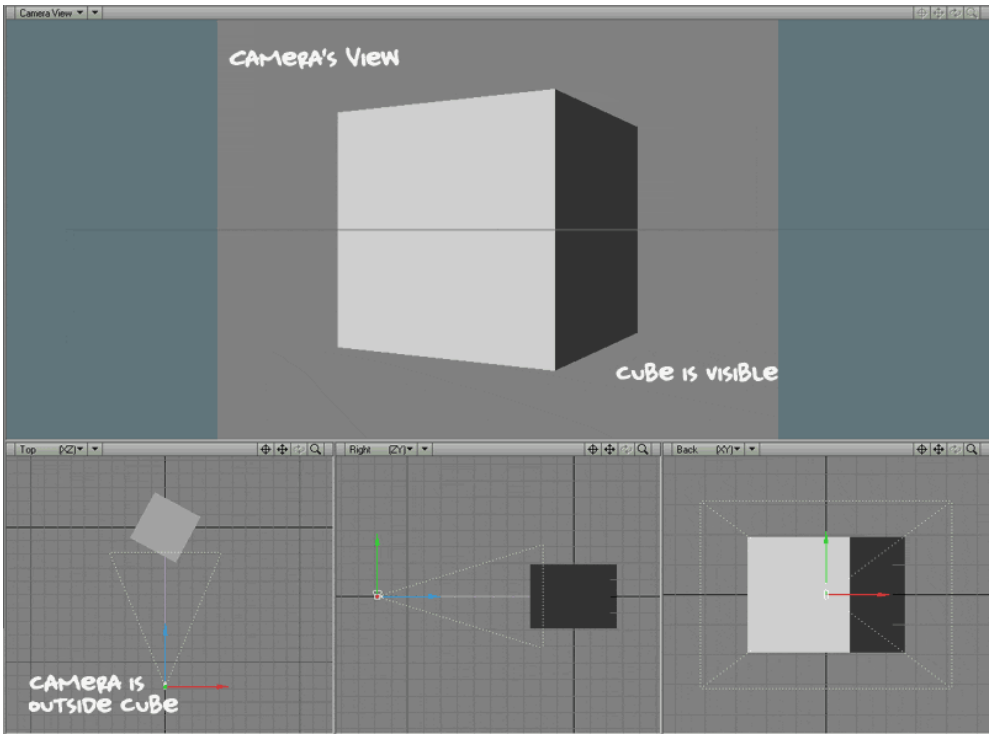
Building glass objects in 3D isn't difficult, but it does take a little planning and a bit of consideration with regard to how raytracing works when rendering.

(Note: I use LightWave, but nearly all of the concepts in this tutorial are software independent.)

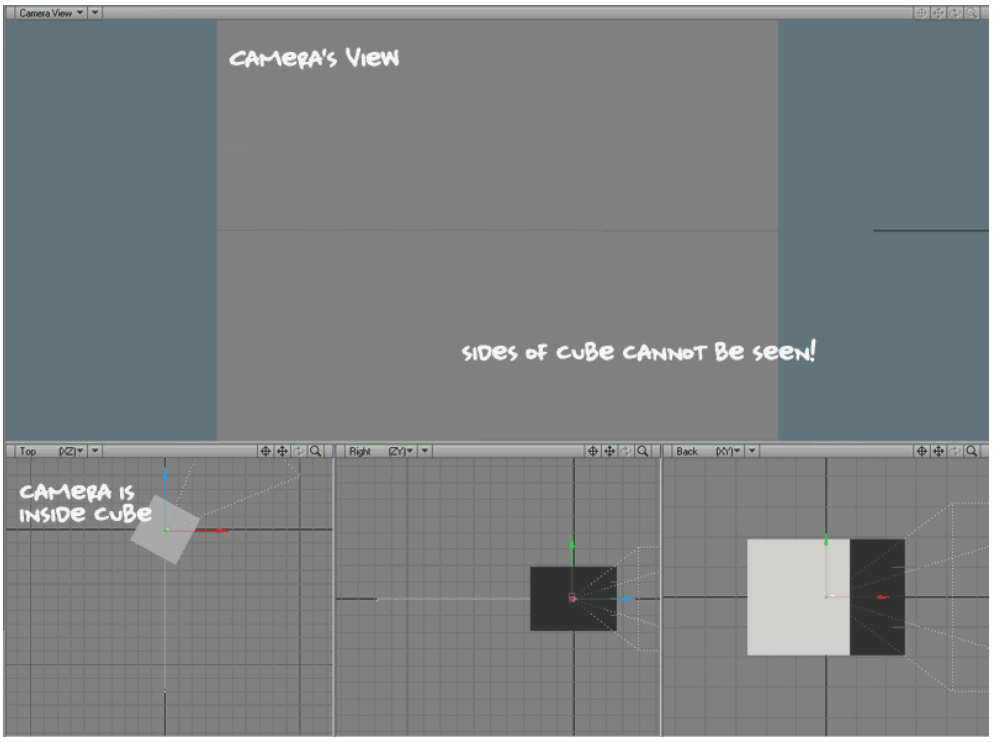
Single-Sided Polys and Normals

If we think of film in a real-world camera, it's easy to imagine the rays of light entering the lens of the camera and impacting the film, resulting in exposure. When raytracing, the virtual camera in a 3D application isn't all that different, except that it fires rays *into* the scene – one or more per pixel of the final rendered image size – and each ray makes its way through the scene until it stops and reports back the proper data for that pixel.

One of the first things to consider when trying to recreate glass in 3D is that most models are typically only one-sided. Create a box in your 3D application. Make it nice and big, maybe 10m per side.



Now move the camera inside the box. If you render the scene with the camera in that position, you won't see anything. The polygons are still there – still around the camera on all six sides of the box, but you can't see them because they're only one-sided, and currently they're facing outward.

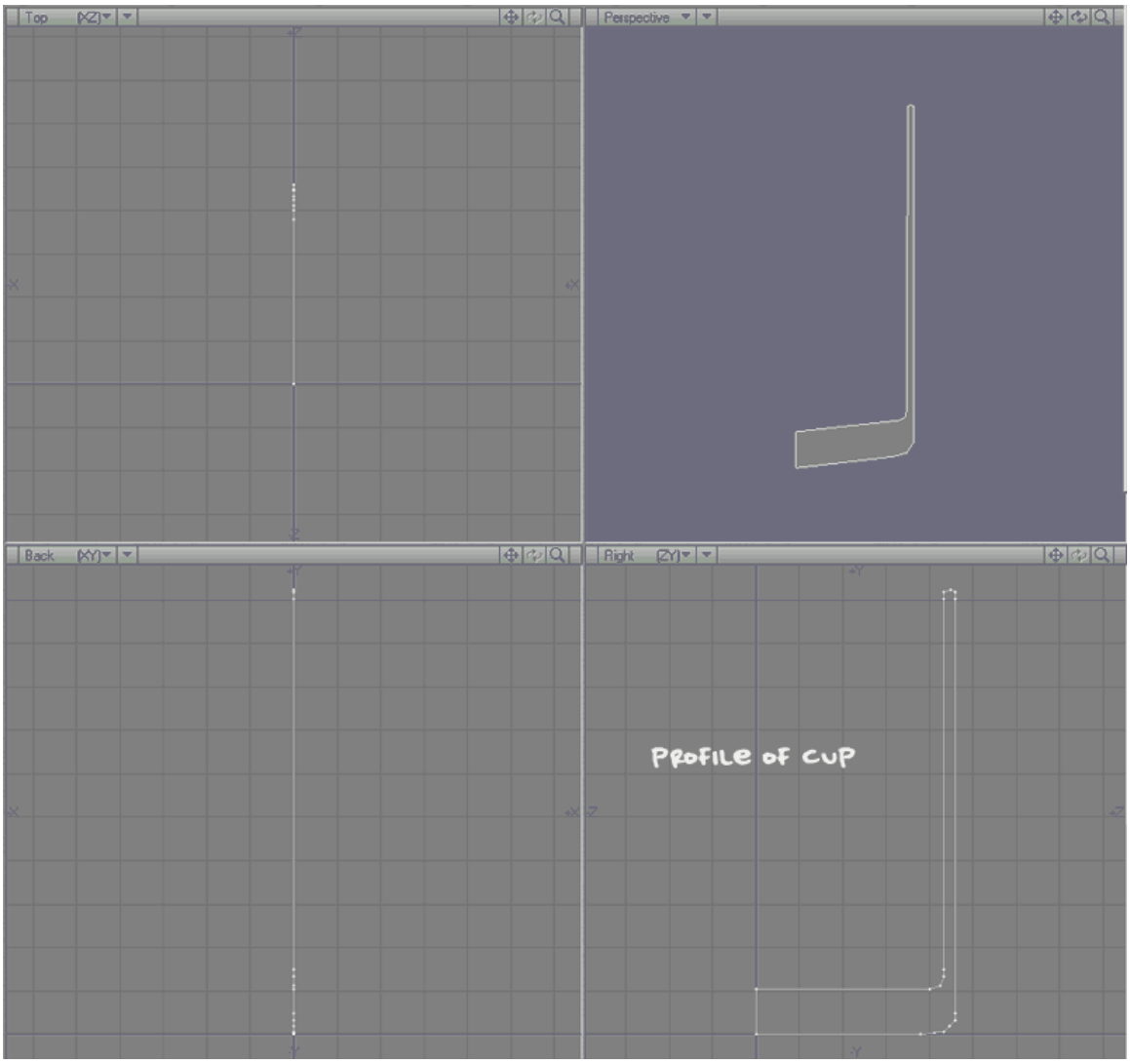


In the high falutin' circles of 3D (i.e. around people who read manuals and that sort of thing), the direction that a given polygon faces is defined by the "normal". Most applications will display polygon normals in the viewport(s) when you begin selecting polygons – each will show a small perpendicular line sticking outward.

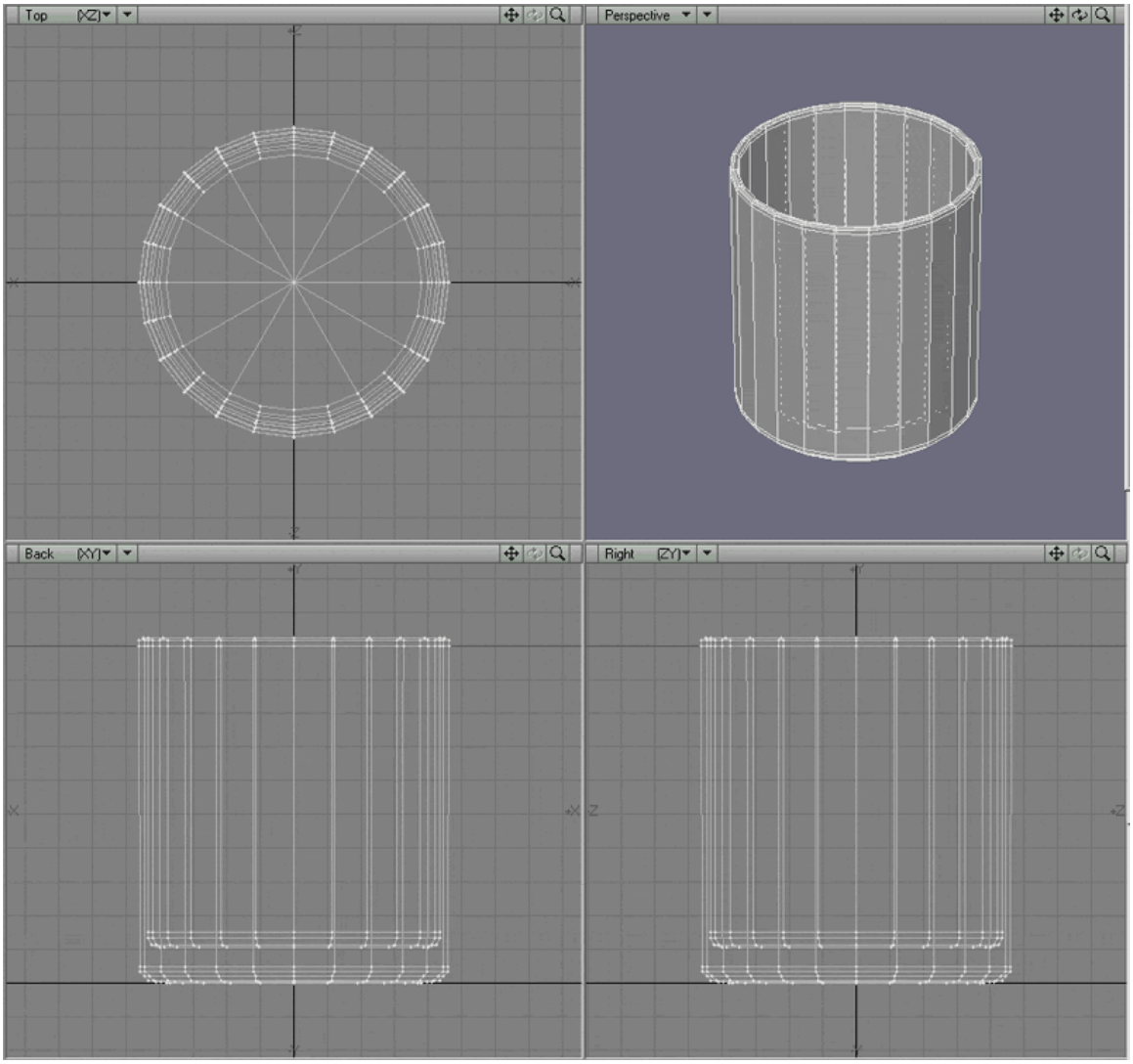
Okay, so now we know that polygons typically default to being single-sided and the direction that they face is determined by the normals. Cool. Let's make some glass.

Modeling the Glass

We're going to build a drinking glass by lathing a profile. Using your 3D weapon of choice, pick your favorite profile creation tool and make a profile of a cup. (I'll use the term "cup" to refer to the object for the rest of this text, to prevent confusion with the term "glass", which will refer to the surface/material.)



Now lathe it. Be sure to use enough sides to get a relatively smooth shape. A subdivided model is fine, too. Set all these polys to be the same surface and call it something clever, like "glass".

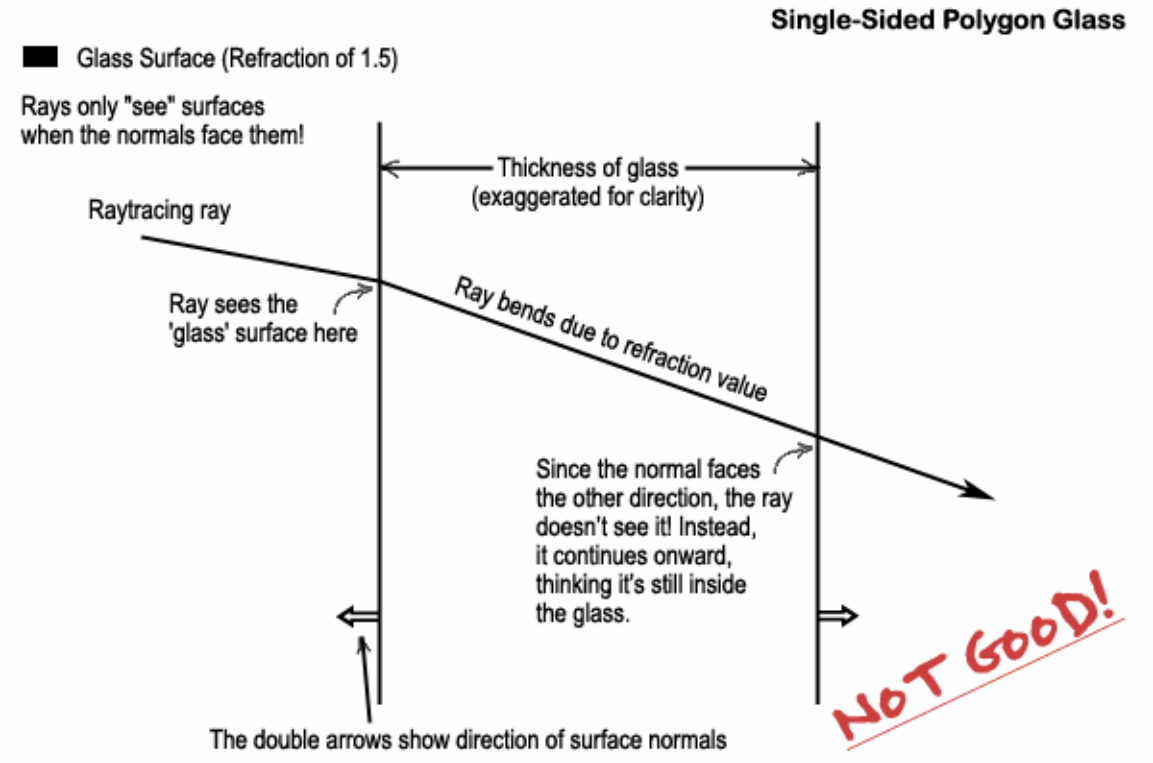


If this cup was of a solid, non-transparent material (like heavy plastic or metal), we could probably stop here and jump straight to surfacing and rendering. But glass has special properties – in this case, transparency and refraction – that need to be taken into consideration at the modeling stage.

Let's say the cup model we have right now is rendered with raytracing, and that it's mostly transparent. When a ray strikes the outermost part of the glass, it sees the transparency and refraction and says, "Ah! That means I have to keep going. Let's see what I hit next."

The ray enters the thickness of the glass, but because the next poly it hits is facing *away* from the ray, the ray doesn't see it – just like the camera couldn't see the back of the polys while it was inside the cube. The ray continues merrily along thinking it's still inside the glass. The next poly the ray will hit is the poly on the inside of the cup on the opposite wall, and guess what? That's got the refraction setting of glass as well. Since the next poly the ray will encounter is facing outward and away from the ray again, the

ray doesn't see it – and it continues along, still believing that everything it has encountered since the very first surface contact is all glass.



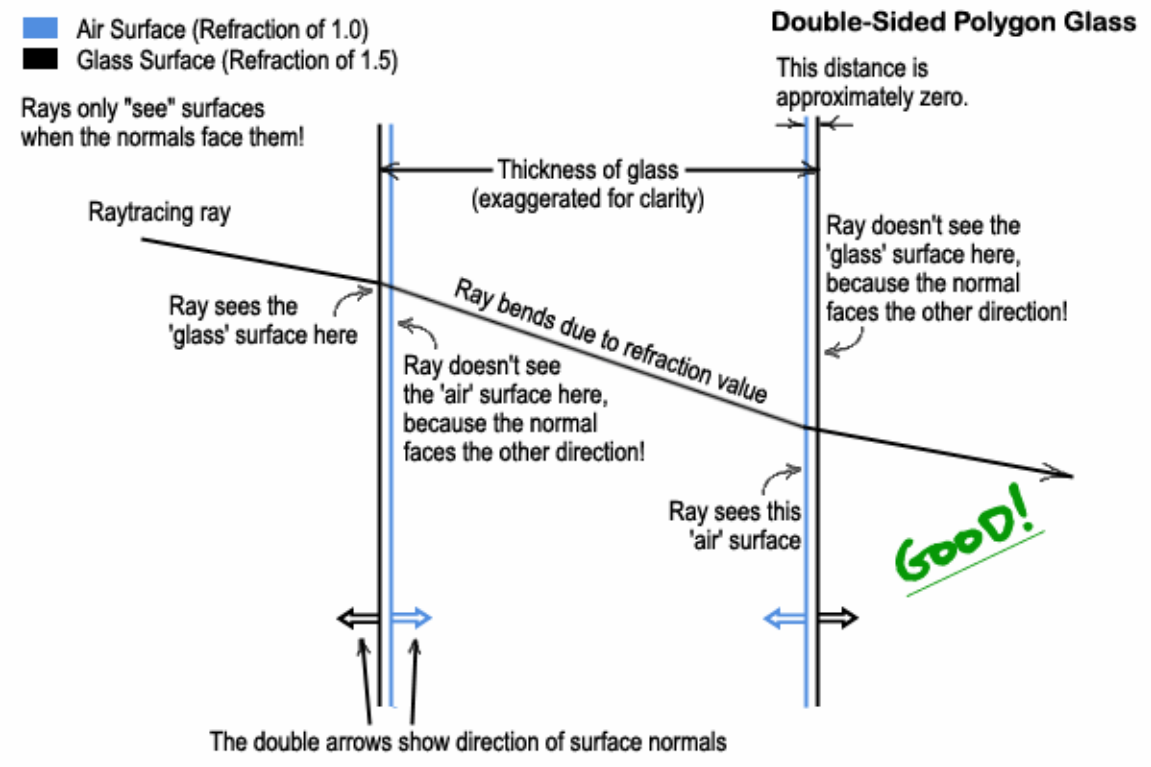
As you can imagine, this does not make for very realistic glass.

Compensating with "Air"

In order for the thickness of the glass walls of the cup to be accounted for, as well as the air within, copy the entire lathed cup. Paste it into a new layer and flip the polygon normals. Give these flipped cup polys a new surface name. Something like "air", or "inner glass".

DO NOT merge the points between the original and the copy. The points of these two sets of polys share the same exact coordinates. It's easy in most 3D apps to make the mistake of merging them.

Now when a ray hits, it enters the glass, then it sees the flipped poly of the first inside wall, where it will revert to "air" refraction for the entire distance inside the cup. Then it hits the opposite inside cup wall and reverts to glass refraction, then sees the flipped version of the outermost poly and will revert once again to air, as it should.



The model is complete. Let's get to surfacing!

Surfacing and Rendering

So what the heck does glass look like? Well, it's obviously got some transparency, but there's also refraction, and reflection. If you're going for very realistic glass, you could add some fingerprints, stains, or other maps to the diffuse and/or reflection channels, etc. I'm just going to cover the basics here.

For the "glass" surface, try Transparency around 99-100% with some Fresnel and/or incidence angle falloff down to about 30%. Set Color to be in the neighborhood of near-white, RGB 236, 236, 236.

Set Refraction at 1.5. For more realistic reflections than Specularity can offer, try an incidence angle gradient on the Reflection channel that goes from 1% to about 60% in a roughly exponential curve and a good HDR or panoramic backdrop.

Luminosity, Diffuse, Specularity, Translucency and Bump can all be set to zero.

For the "air" or "inner glass", we really just want the refraction value to be updated. Transparency should be 100%. Set Refraction to 1.0. Everything else can be zero.

Add a ground plane under your modeled cup, and a basic light and you're good to go! Be sure to turn raytracing on for all the appropriate effects (transparency, shadows, refraction, reflection).