**Basic transparency, secret information taken from the Illuminati\*\***

(\*\* so be careful with this very dangerous, powerful information. LoL.)

There’s two basic ways you can pull of “transparency”:

1. “Alpha” or “blend” transparency (technically it’s a form of “blend” if you wanna be pedantic)
2. Discard transparency

**Alpha/blend transparency:**

Draw the transparent thing “in front” of something else, combining or “blending” what’s already there with what you are now drawing.

Imagine two (2) stickers, one “solid” (opaque) and one semi-transparent, and you’re going to stick them onto your laptop. As you do.

You’d stick the opaque one on first, then the semi-transparent one on top of that. If you think about it, it looks “transparent” because you can see through the semi-transparent one. Or the colour of the semi-transparent sticker is combined with the sticker underneath, right?

If you put the semi-transparent one on first, then the opaque one, you couldn’t see the semi-transparent one at all. Why? Well it’s because one is covering (occluding) the other, but you could also think of it like this: the colour of the top sticker isn’t “being combined” with the sticker underneath.

Like if you were very artistically inclined, you *could,* in theory, colour the opaque stick (say with markers or whatever) to make it look like you could see the sticker underneath.

And if was coloured really well, how would you know the difference?

This is called “alpha” transparency and uses the “blend” function, which is a “fixed function” or “in hardware” feature of the hardware. In other words, you don’t “program it”, but set values and the hardware “just does it”.

The “blending” refers to the colour of the current pixel fragment, which is combined or “blended” with whatever is already on the frame/back buffer. Note that this involves a read from the buffer then a write, so theoretically takes at least 2x the speed of *not* blending. But like many things, I wouldn’t worry too much until it’s an issue.

You can also do all sorts of “blending”, so the settings are a little odd. We’re going to use the “1 – source alpha” setting. This takes whatever is currently there, then the current pixel is multiplied by (1-alpha) and then added to whatever is already there.

The transparency goes from 0.0 to 1.0, where 0.0 means it’s completely transparent, and 1.0 is completely opaque (i.e. completely overwrites what’s already there).

You have to set two things: the “blend function” and also enable it.

glEnable(GL\_BLEND);

glBlendFunc( GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA );

You can enable and disable this whenever you want, and should turn it off when it’s not needed (for opaque objects).

Since transparency only “works” when there’s something already on the back/frame buffer, you have to draw the transparent things last.

Also, if there’s a possibility two or more transparent things overlap, you have to draw the transparent things *in order from “back to front”* with *respect to the camera in that frame*.

Here’s the summary of an easy way to do that:

1. Split everything into “transparent” and “non-transparent” (opaque).
2. Turn off blending.
3. Draw all the non-transparent things in whatever order you’d like.
4. Sort the transparent things relative to the camera.
   1. The distance from the camera can be calculated using the ray between object and camera.   
      i.e. ObjectXYZ – CameraXYZ, then get the length of this vector.  
      If you don’t want to do the square root, you can use the “distance squared” (gives same result)
   2. You *don’t* need to do a full sort every time. In fact, if there’s lots of objects a complex cache-unfriendly sort (99% of the “classic” sorts you’ve learned BTW) will be a complete waste of time as well as slow.   
      (No matter what one you use)  
      Think about it: Once sorted, they will only get “out of order” when the camera has moved enough to make this happen. You can simply do *one pass* of the bubble sort per frame and it’s fine.   
      (i.e. not even the entire bubble, but just a *single pass*)  
        
      **Remember:** 
      * You are drawing at 30-60 fps minimum.
      * The bubble sort is *the* fastest (and cache friendly) sort if the list is sorted/almost sorted.
      * Don’t believe the “hype” about other sorts in trivial cases – it’s not 1989 anymore.
5. Turn on blending.
6. Draw the transparent things from “back to front” (far to near relative to the camera)

The transparency goes from 0.0 to 1.0, where 0.0 means it’s completely transparent, and 1.0 is completely opaque (i.e. completely overwrites what’s already there).

**Discard transparency:**

The other way to make things transparent is just not to draw them.

In the shader, calling “discard” simply exits the shader without doing anything. In other words, “discard” the current pixel fragment.

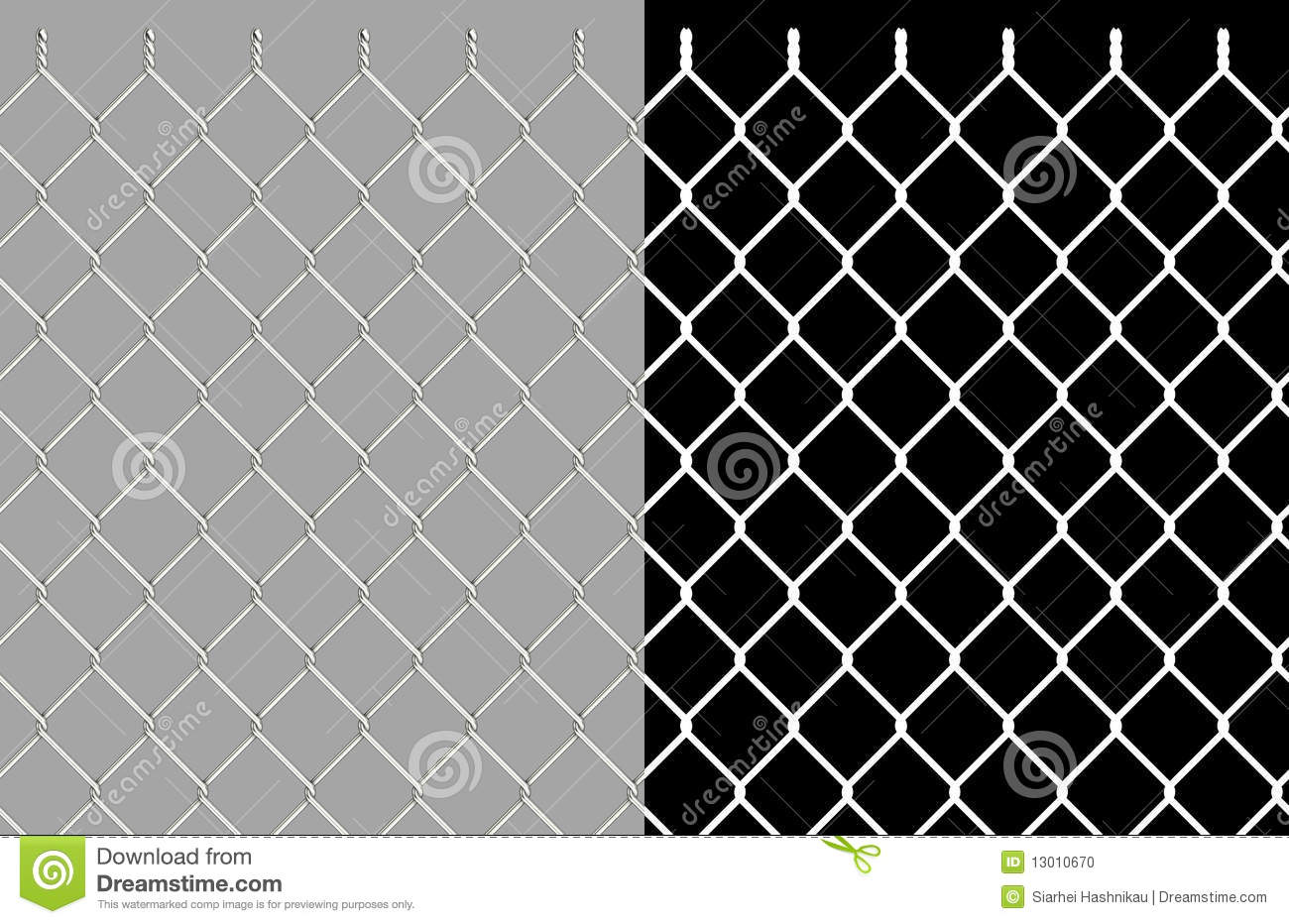
It’s literally this:

**discard;**

It immediately exits and *doesn’t* write to any buffer (back/frame, depth, stencil, etc.).

It’s like the pixel fragment never happened.

This is useful for objects that are either completely transparent or 100% opaque.

When would you do this? Well, if there’s “holes” in objects that you don’t want to model (make a 3D mesh), the classic example is a chain link fence:

This is from: <https://www.dreamstime.com/stock-photo-shiny-wire-chain-link-fence-image13010670>

This is really two textures, being shown side by side.

Ignoring the watermarks, note that the left hand side is the fence (with grey colour in the parts between the wires).

The right side is just a black and white “mask”, where the white parts match where the fence is and the black parts are were, well... where it isn’t.

If you place this texture on a flat surface, you can compare this right “mask” texture – where it’s black, you discard.

What happens is the fence is only drawn to the back/frame buffer where there’s a fence, but you can see through the “in between” parts – because they aren’t actually there; they were never drawn.

Unlike alpha blend transparency, we don’t have to worry about the draw order.

**Note:**

* With desktop/laptop cards, this *can* be a speed up, but not always. So you shouldn’t assume it is.
* On mobile devices, it straight up ***isn’t***a speed up – this is because mobile devices use “tiled” rendering.
* Anyway, the point is if you here “discard speed up rendering”, that’s simply not true. Maybe for trivial cases, but that’s not what it’s for, anyway – it’s meant for “faking” more complex geometry that isn’t really there.   
    
  So another “graphics lie”, in other words. LoL.