

3d Projection

Intro

This is one of the final steps before anything is actually blitted to the screen. A lot of programmer's choose to include volume clipping in their projection functions because it would be a waste of time to do projection on points or objects that are out of the viewing area. We are covering two types of projection, parallel and perspective. Most games use perspective projection to give the user a real view of the world around them, while parallel projection is typically used for graphics that don't need that sense of realism.

Parallel Projection

Parallel projection is just a cheap imitation of what the real world would look like. This is because it simply ignores the z extent of all points. It is just concerned with the easiest way of getting the point to the screen. For these reasons, parallel projection is very easy to do and is good for modeling where perspective would distort construction, but it is not used as much as perspective projection. For the rest of this tutorial, assume that we are dealing with members of our point class that we developed. Notice that our test object to the right after projection each edge is perfectly aligned just as it was in the world. Each edge that was parallel in the world appears parallel in the projection. The transition from world coordinates to screen coordinates is very simple.

$sx = wx + XCenter$
 $sy = -wy + YCenter$



Remember that our world is designed to be viewed from the center (0,0,0). If we just used the world coordinates, our view would be incorrect! This is because the upper left of the screen is (0,0) and this is NOT aligned with what we want. To correct this, we must add half the screen width and half the screen height to our world coordinates so that the views are the same. Also notice that we are taking $-1 * wy$. By now you should be able to guess why! It is because in our world, as the object moves up, its y extent increases. This is completely opposite of our screen where moving down increases its y extent. Multiplying the y extent by -1 will correct this problem as well! I told you it was simple!

Perspective Projection

This type of projection is a little more complicated than parallel projection, but the results are well worth it. We are able to realistically display our world as if we were really standing there looking around! The same display issues face us as before 1. We need to align the screen (0,0) with the world (0,0,0) 2. We need to correct the y directional error. We correct the two problems as we did before. We add half the screen width and height to our point, and then reverse the sign of the y extent. Here's a generalized equation for the actual perspective projection.

$$\begin{aligned} \text{OneOverZ} &= 1/wz \\ \text{sx} &= wx * \text{XSCALE} * \text{OneOverZ} + \text{XCenter} \\ \text{sy} &= -wy * \text{YSCALE} * \text{OneOverZ} + \text{YCenter} \\ \text{sz} &= wz \end{aligned}$$

The only real difference between these equations and our parallel relatives are the addition of the XSCALE, YSCALE and OneOverZ variables. Firstly we define OneOverZ (double) which stores 1 divided by the z extent of the point. We need to multiply our points by this because by its nature, as points move further away, they will move closer to 0. This is exactly what we can notice in the “real” world. If we look into a plastic tube, the sides look as if they are moving closer to the center. The longer the tube, the more this effect is noticed. We then define XSCALE and YSCALE. These are used to adjust our Field Of View. The larger our FOV, the more squished together objects appear. This makes sense since we have a finite viewing area, and in order to accommodate a larger FOV, we will have to squish stuff together. I’ve found that when displaying objects, each video mode needs a different FOV factor to make it look truly realistic. Play with the numbers to see what you like the best! Most vary from 40-150. Adjusting one of the FOV variables while leaving the other unchanged will make the display appear to stretch in that extent. Look at our cube now with a high FOV, the edges are far from being parallel!



This tutorial along with the 3d transformations tutorial is more than a good start in the right direction to make some really neat effects! Enjoy yourself! If you have any question, comments, complaints, whatever! send me some Feedback.

Contact Information

I just wanted to mention that everything here is copyrighted, feel free to distribute this document to anyone you want, just don’t modify it! You can get a hold of me through my website or direct email. Please feel free to email me about anything. I can’t guarantee that I’ll be of ANY help, but I’ll sure give it a try :-)

Email : deltener@mindtremors.com

Webpage : <http://www.inversereality.org>

Open Your Mind