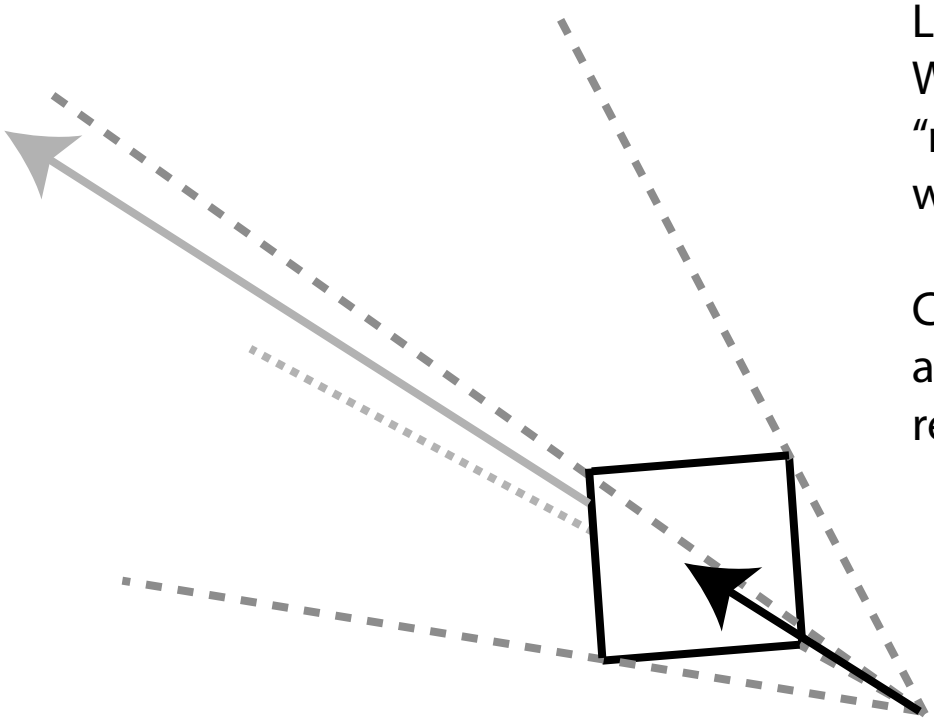
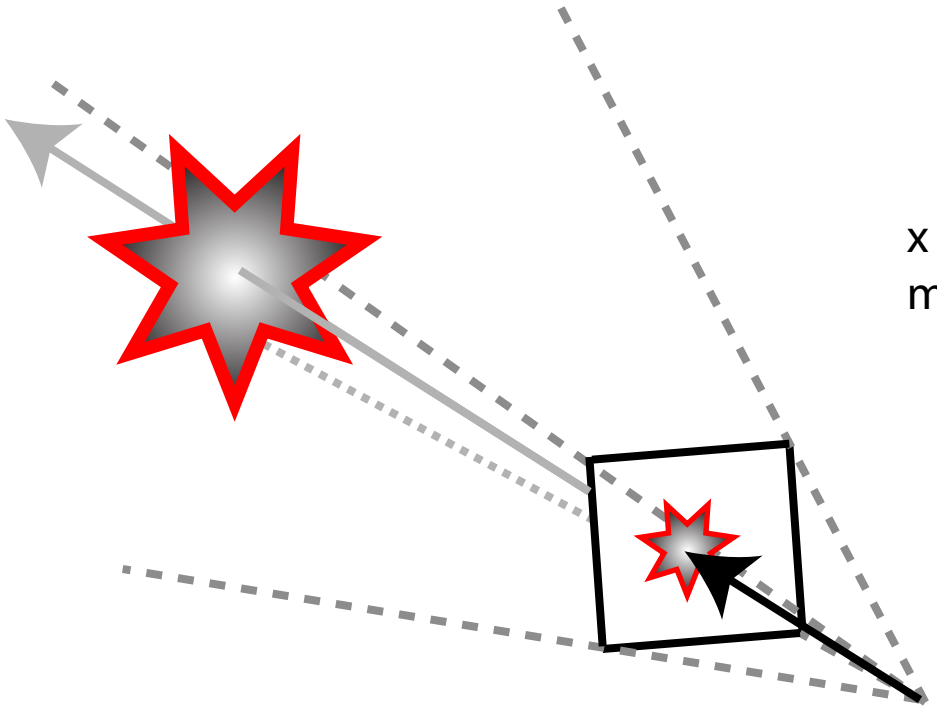
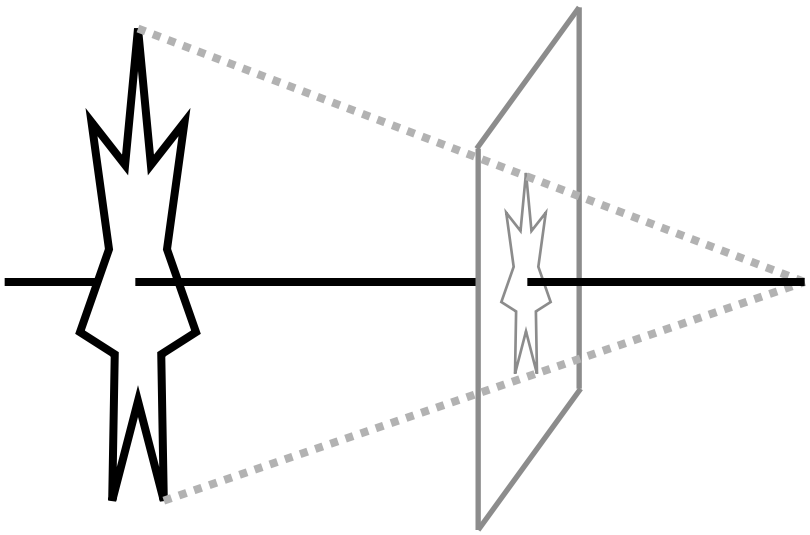


Looking down the z-axis (arrow). We will project geometry onto a "near plane" (black rectangle), which corresponds to the screen.

Consider, how this projection affects x and y coordinates of rendered geometry....

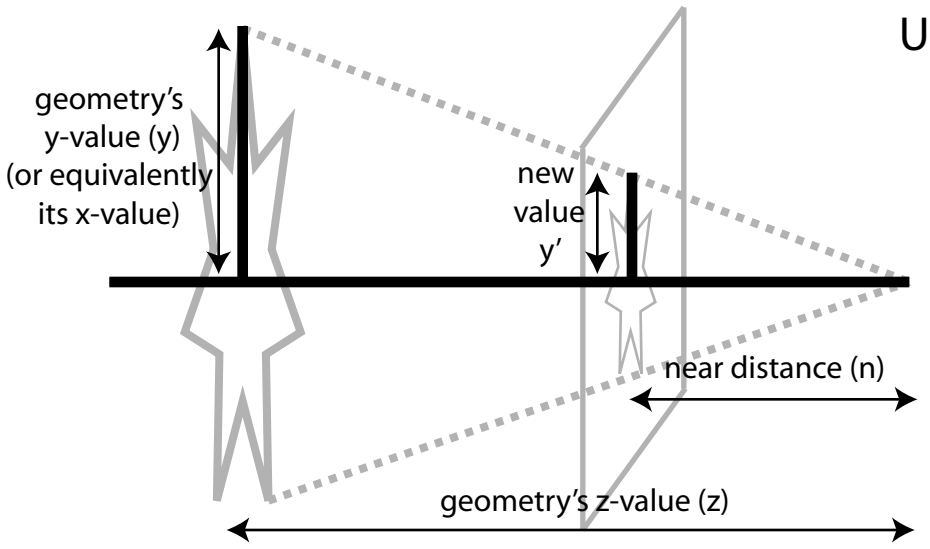


x and y values are simply scalar multiples of their original values!



So let's try to figure out how much scaling occurs!

In OpenGL, the eye is at (0,0,0), and we specify the distance to the projection plane (i.e., the "near"plane).



Use simple, high school geometry!
(similar triangles)

$$\frac{x'}{n} = \frac{x}{z} \qquad \frac{y'}{n} = \frac{y}{z}$$

That suggests a matrix like:

$$\begin{pmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} nx \\ ny \\ nz \\ z \end{pmatrix} = \begin{pmatrix} nx/z \\ ny/z \\ n \\ 1 \end{pmatrix}$$

There's two problems with this matrix!

$$\begin{pmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

(Q: Anyone have ideas what?)

Problem 1) It throws away all depth information in the scene! (z-values) This means there's absolutely no way to tell which object is on top in any pixel where multiple objects overlap.

Problem 2) It doesn't guarantee all geometry in the volume will project into the unit cube (where $x, y, z \in [-1...1]$). Remember, we want values in this range after projection ("normalized device coordinates").

In order to address this problem we start with a matrix of this form:

$$\begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & d \\ 0 & 0 & e & 0 \end{pmatrix}$$

and fill in the values for a, b, c, d, and e. Note, the added non-zero entry d allows some preservation of depth differentials. We need not add other non-zero entries in the 3rd row, since the final z-value should not depend on the input x any values!

So our desired matrix is of this form:

$$\begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & d \\ 0 & 0 & e & 0 \end{pmatrix}$$

First, realize this is a system of 4 equations with 5 unknowns.
(We'll get a set of solutions instead of a unique one!)

Fix this by setting e = -1.

Then, let's consider what happens when we apply this matrix to important points.

Q: If we apply the projection matrix to (0,0,-n,1) with this matrix, what should be the z-value of the projected point?? (Remember the projection matrix leaves points in "normalized device coordinates," where x,y,z $\in$ [-1...1].)

A: (0,0,n,1) should map to (0,0,-1,1) in normalized device coordinates!

Q: If we apply the projection matrix to (0,0,-f,1) with this matrix, what should be the z-value of the projected point??

A: (0,0,f,1) should map to (0,0,1,1) in normalized device coordinates!

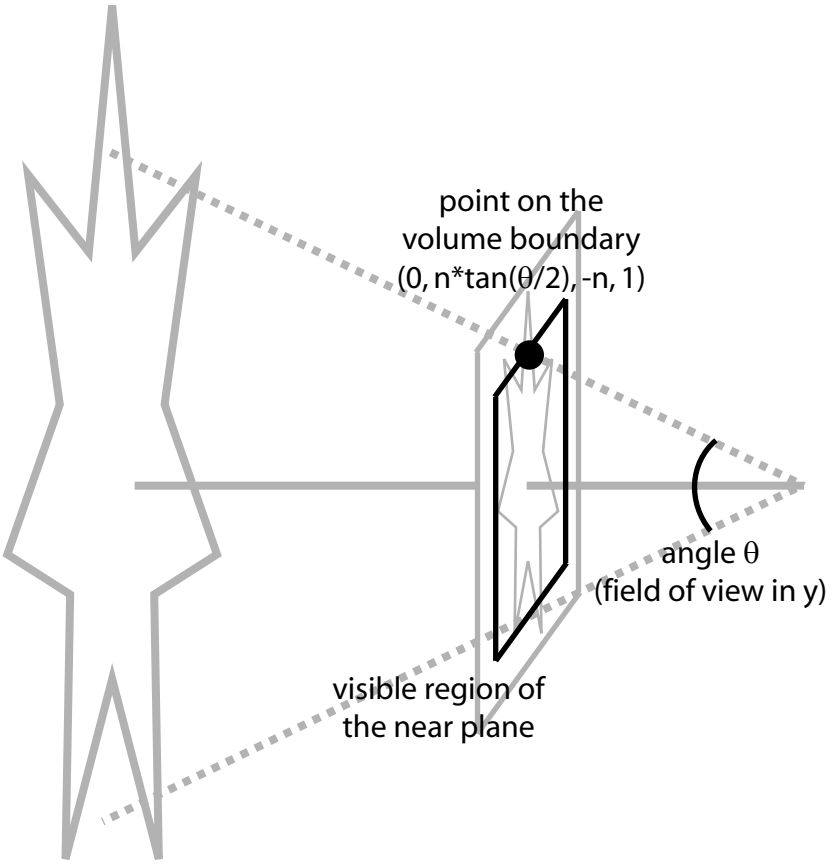
$$\begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & d \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ -n \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -cn+d \\ n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -c + d/n \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -1 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & d \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ -f \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -cf+d \\ f \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -c + d/f \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

So, -cn+d = -n, and -cf+d = f. Subtract the second from the first: -c(n-f) = -n-f

Thus, c = (n+f)/(n-f).

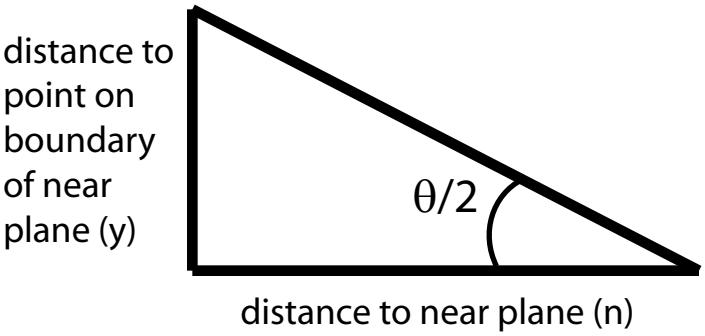
Also, -c + d/n = -1, and -c + d/f = 1. Subtract the second from the first:
d/n - d/f = -2 (df-nd)/(nf) = -2. Thus, d = 2nf/(n-f).



Now, let's tackle a and b.

We can do the same thing, plug in important points that project to the boundaries of the near plane!

To find what this point is, consider the view on the left (and the simplified view below).



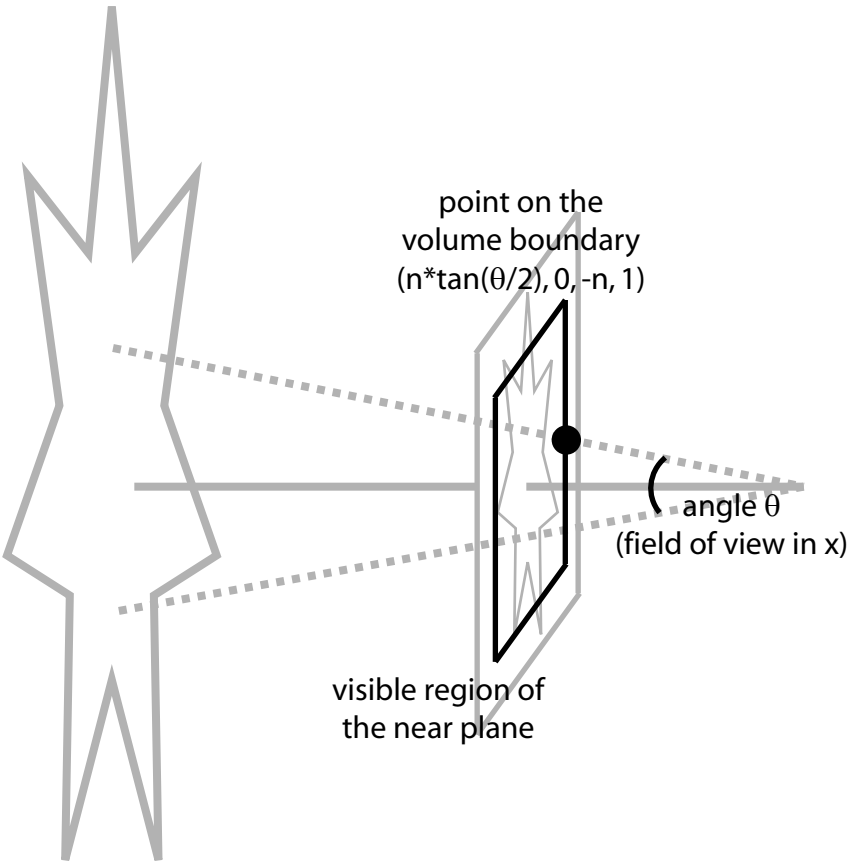
Basic trigonometry says:

$y = n \tan(\theta/2)$,

So the point $(0, n \tan(\theta/2), -n, 1)$ is a boundary condition, and should project to $(0, 1, -1, 1)$

$$\begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ n\tan(\theta/2) \\ -n \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ bn\tan(\theta/2) \\ (a\ mess) \\ n \end{pmatrix} = \begin{pmatrix} 0 \\ b\tan(\theta/2) \\ (a\ mess) \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ -1 \\ 1 \end{pmatrix}$$

This shows us that $b = 1/\tan(\theta/2)$. Simplifying, and calling θ by the name (*fovy*) we pass it to gluPerspective() with, we get: $b = \cot(\frac{fovy}{2})$



For computing a, the situation and math is almost identical....

Except the angle θ might be different if the window's aspect ratio is not 1 (i.e., if the window is not square)!

Remember the aspect ratio is:
width / height

Remember: $y = n \tan(\frac{fovy}{2})$, this means: $x = n \tan(\frac{fovx}{2})$,

But since we don't have fovx (remember we specify only fovy when calling gluPerspective()), we can compute it based on fovy and the aspect ratio (a_r)

It turns out that the boundary point is $x = n\, a_r \tan(\frac{fovy}{2})$. So, our boundary point is: $(n\, a_r \tan(\frac{fovy}{2}), 0, -n, 1)$

$$\begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} n\, a_r \tan(\frac{fovy}{2}) \\ 0 \\ -n \\ 1 \end{pmatrix} = \begin{pmatrix} a*n*a_r*\tan(\frac{fovy}{2}) \\ 0 \\ (a\ mess) \\ n \end{pmatrix} = \begin{pmatrix} a*a_r*\tan(\frac{fovy}{2}) \\ 0 \\ (a\ mess) \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -1 \\ 1 \end{pmatrix}$$

This shows us that $a = 1/(a_r \tan(\frac{fovy}{2}))$. Simplifying, we get:

$$a = \frac{\cot(\frac{fovy}{2})}{a_r}$$