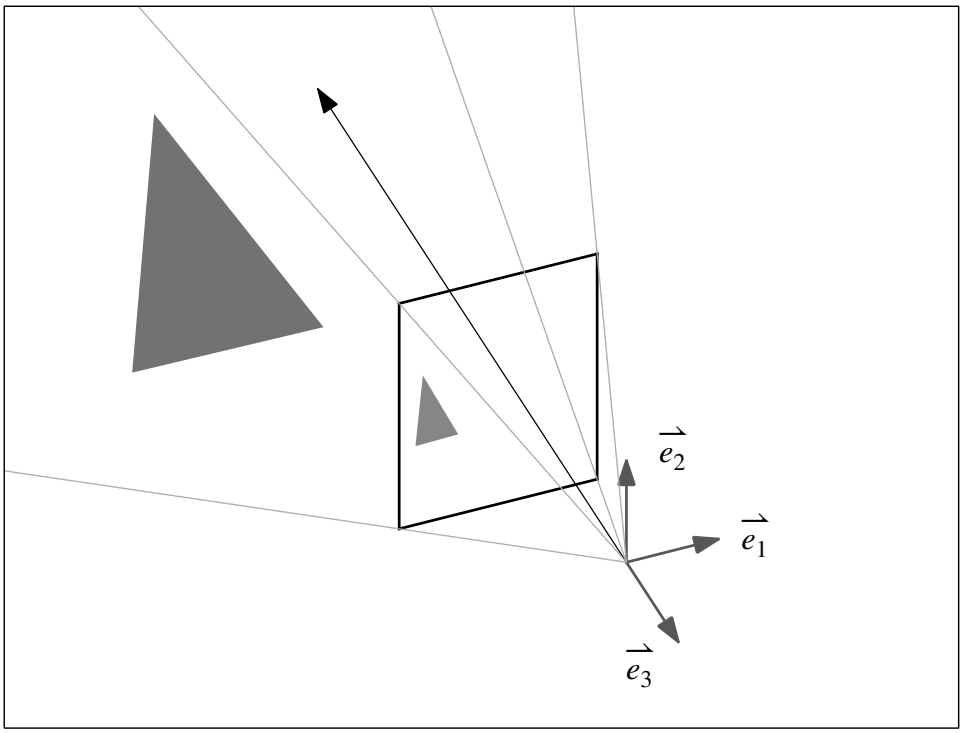
Projections

I have already mentioned that in order to see the world from the perspective of our imaginary eye we will have to translate every vertex in the graphics world into eye coordinates using an eye coordinate frame.

The next operation to perform is that of *projection*. In this object we imagine projecting every object visible from the eye location onto an imaginary *view rectangle* positioned in front of the eye.

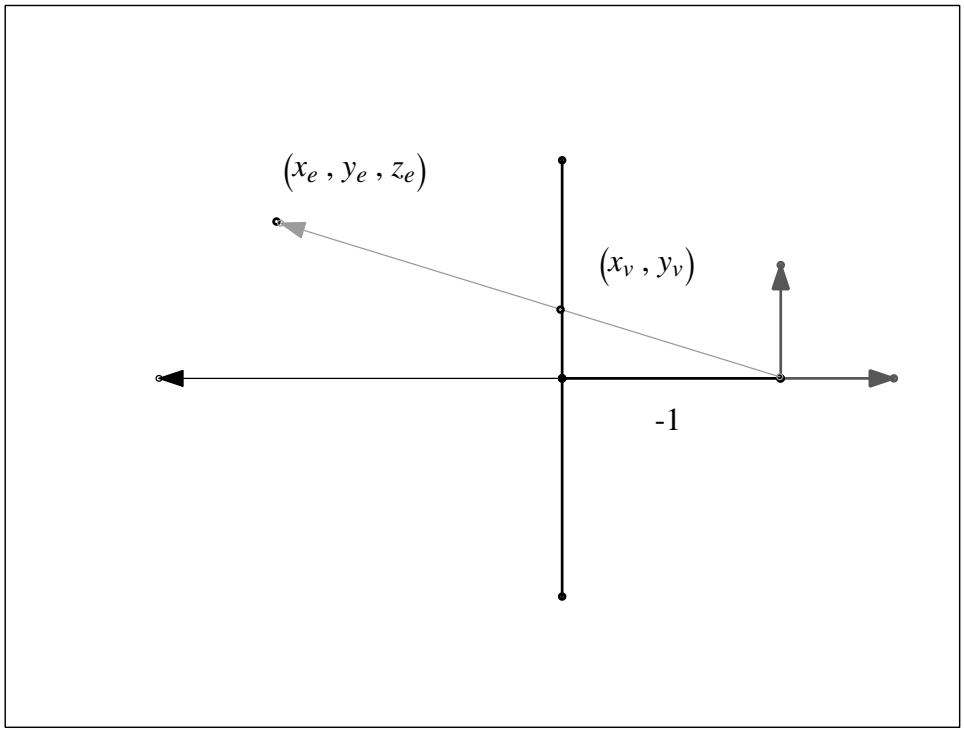


In its simplest form, we can imagine the view rectangle as being a rectangle centered at the point (0,0,-1) in eye coordinates and extending over the range $-1 \leq x \leq 1$ and $-1 \leq y \leq 1$ in eye coordinates. This view rectangle will be equipped with its own private coordinate system of view coordinates, (x_v , y_v).

Every vertex in the scene will then be projected onto the view rectangle. In the process, a vertex with eye coordinates (x_e , y_e , z_e) will get projected onto the view plane at a point (x_v , y_v).

The mathematics of projection

The mathematics of projection are actually quite simple to work out. Consider the diagram below, which shows a side view of the eye frame and the view rectangle. A vertex in the scene located at (x_e , y_e , z_e) gets projected to the point with view coordinates (x_v , y_v).



A simple similar triangles argument gives us that

$$\frac{y_v}{-1} = \frac{y_e}{z_e}$$

or

$$y_v = -\frac{y_e}{z_e}$$

A similar argument shows that

$$x_v = -\frac{x_e}{z_e}$$

The mathematics of the projection transformation are easy enough to work out. The only problem with this transformation is that it does not match the form of other coordinate transformations we have been carrying out. In particular, there is no matrix M such that

$$M \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{x_e}{z_e} \\ -\frac{y_e}{z_e} \\ -1 \\ 1 \end{bmatrix}$$

The closest we can get to producing something like this is to use a *projection matrix*

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

to produce

$$M \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix} = \begin{bmatrix} x_e \\ y_e \\ 1 \\ -z_e \end{bmatrix}$$

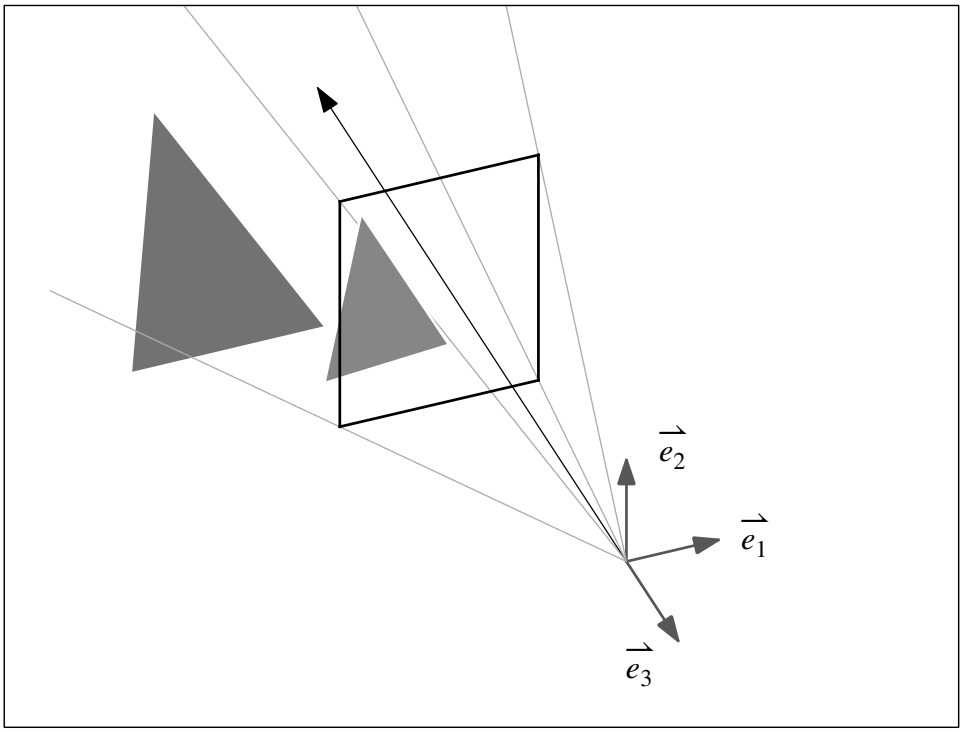
The obvious thing wrong with this result is that its last coordinate is not 1, which is what we expect to get for points in homogeneous coordinates. We can fix this problem by performing one final step, called *perspective division*. In this step we divide all four coordinates by the last coordinate.

$$\begin{bmatrix} x_e \\ y_e \\ 1 \\ -z_e \end{bmatrix} \Rightarrow \begin{bmatrix} -\frac{x_e}{z_e} \\ -\frac{y_e}{z_e} \\ -\frac{1}{z_e} \\ 1 \end{bmatrix}$$

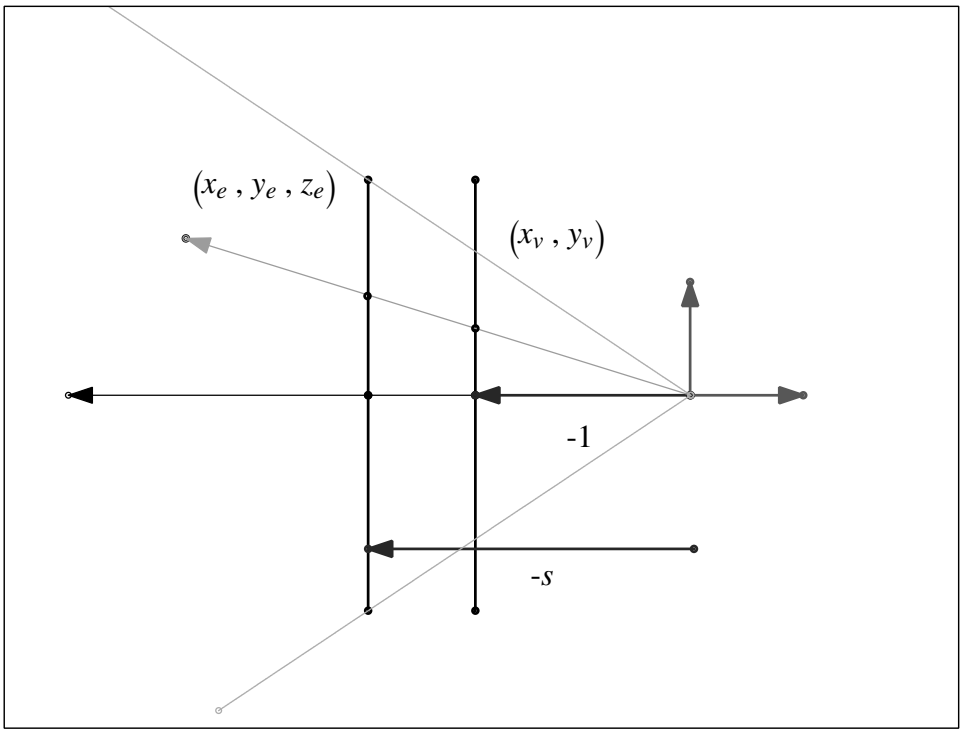
Achieving other effects

Besides doing projections onto a view rectangle, we will also want to extend the mathematics of the projection matrix step to achieve other useful effects. Below are some effects we may wish to achieve and ways to modify the projection matrix to produce these effects.

The first effect to consider is a zoom effect. We can produce a zoom effect by moving the view rectangle around in the scene. To zoom in on a portion of the scene, we can move the view rectangle away from the eye and closer to the object we want to zoom in on.



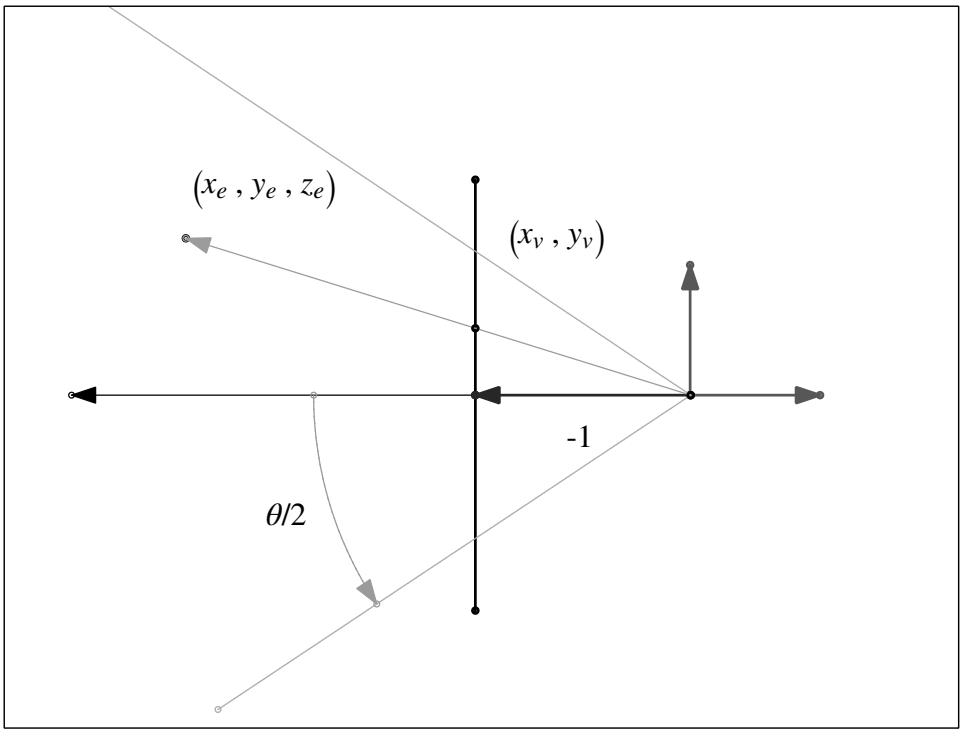
An equivalent way to achieve this effect is to narrow the field of view and subsequently rescale the projected image so that it fills up the standard view rectangle of -1 ≤ x_v ≤ 1 and -1 ≤ y_v ≤ 1.



We achieve either effect by using a projection matrix that looks like

$$M = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

We can specify this scale factor via the position of the view rectangle or by specifying the view angle θ:



The relationship between the view angle and the scale factor is

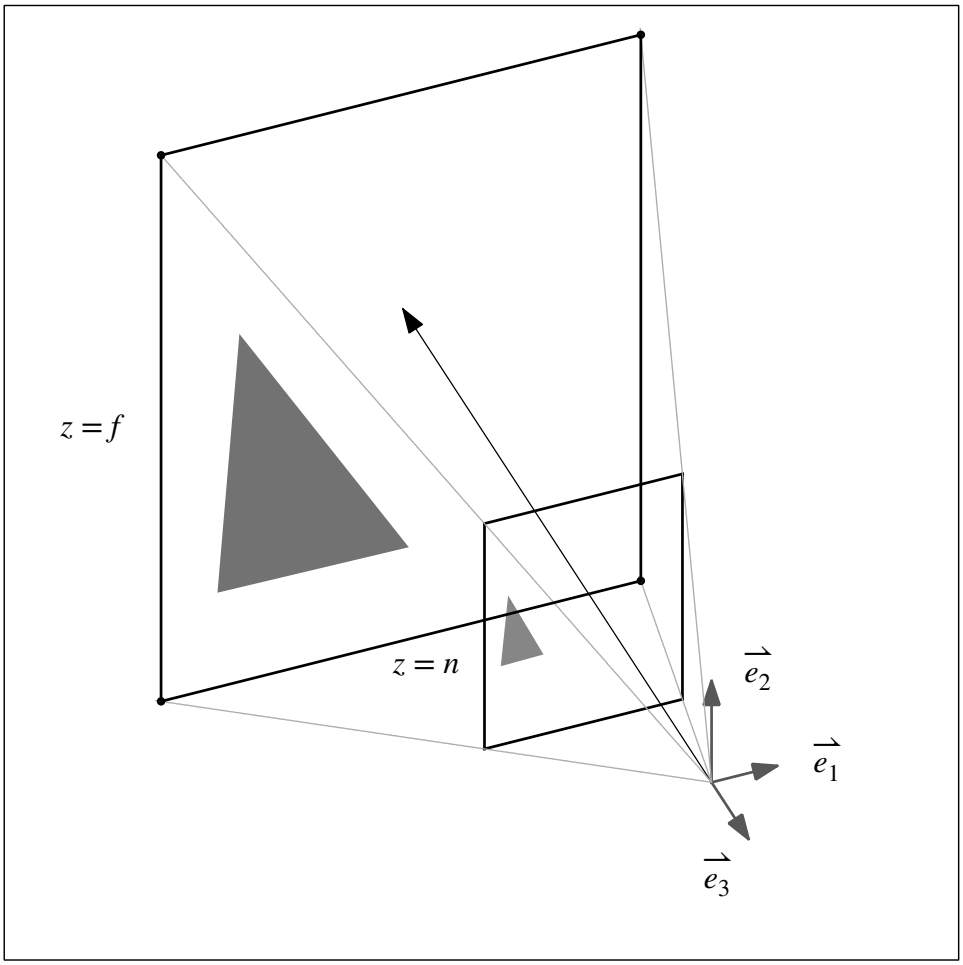
$$s = \frac{1}{\tan(\theta/2)}$$

One final effect we can achieve is reposition the center of the view rectangle. Right now, the negative z-axis in eye coordinates passes through the point $(0,0)$ in view coordinates. If we instead want to shift the view rectangle so that the negative z-axis passes through the point $(-c_x , -c_y)$ in view coordinates we can use the projection matrix

$$M = \begin{bmatrix} 1 & 0 & -c_x & 0 \\ 0 & 1 & -c_y & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Near and far planes

A further modification is to bound the viewable region in space by setting up a *view frustrum*. We do this by setting up a viewing rectangle, as before, and then setting up a second "far" rectangle to bound the extend of the scene that we will declare to be viewable. Rays from the eye to the corners of the "near" viewing rectangle pass through the corners of the far rectangle. The two planes and the rays together bound a region in space called the view frustrum.



Here is a variant of the standard projection matrix that has the effect of mapping any point with $z_e = f$ to -1 and any point with $z_e = n$ to 1:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Putting it all together

We can get full control over the view frustrum by specifying the bounds of the near plane in eye coordinates as well as the near and far values in eye coordinates. For a near rectangle with edges at $x_e = r$, $x_e = l$, $y_e = t$, and $y_e = b$ and near and far planes positioned at $z_e = n$ and $z_e = f$, the projection matrix

$$M = \begin{bmatrix} -\frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & -\frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

maps points inside the view frustrum to a cube in *clip coordinates* with $-w_c \le x_c \le w_c$, $-w_c \le y_c \le w_c$, $-w_c \le z_c \le w_c$. After the perspective division step in which we divide each point by its fourth coordinate, the points inside the view frustrum map to the cube $-1 \le x_n \le 1$, $-1 \le y_n \le 1$, $-1 \le z_n \le 1$ in *normalized coordinates.*