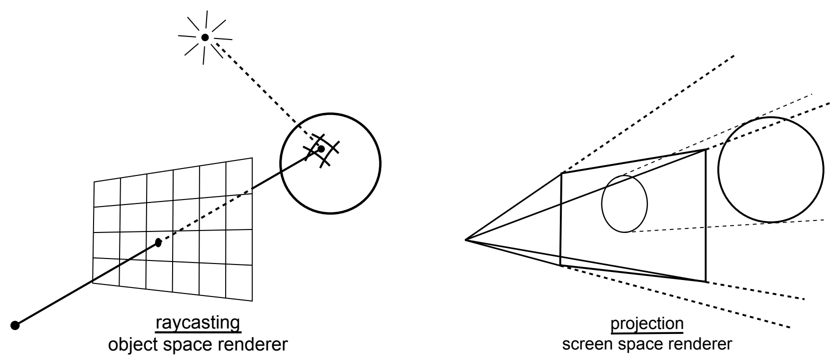
Chapter 10

Orthographic and Perspective Projection

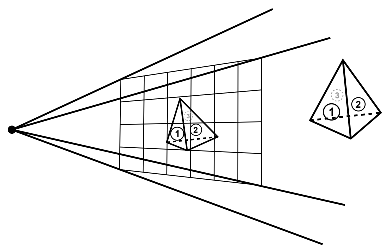


We have been, until now, creating images by raycasting. By shooting rays from the eyepoint out into the scene, we determine what is visible at the screen pixel that the ray passes through. This type of renderer is called an object (or scene) space renderer, since all viewing calculations are done in the world space of the scene. This approach is depicted above to the left. We are about to begin looking at *OpenGL*, which uses a very different approach. OpenGL uses a screen space rendering approach, as depicted above to the right, which works by:

1. transforming all scene objects from world coordinates to camera coordinates by a viewing transform,

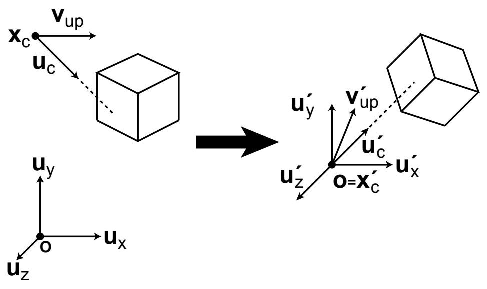
2. projecting all scene geometry into 2D screen space and then using this projection to produce a shaded image.

For example, consider the tetrahedron in the picture to the right. If we were to project the tetrahedron onto the view plane, it might be that only two of the four triangular faces would be visible. We could color all of the pixels covered by triangle 1 the color of this face, and all those covered by triangle 2 the color of that face, thus making an image of the tetrahedron from this viewpoint.



10.1 View Transform

Let us begin by considering how to transform the scene into camera coordinates. We want to view the scene from the camera's point of view. As a convention, in camera space we will let the camera's viewpoint be the origin $\mathbf{x}_c = (0, 0, 0)$, the camera's view direction the negative z axis $\mathbf{u}_c = (0, 0, -1)$, and the viewscreen's up direction the y axis $(0, 1, 0)$. If the camera does not start out in this position and orientation in world space, we have to make a change of coordinates. This amounts to first translating the entire scene so that the camera is at the origin, then rotating the scene about the x and y axes so that the view direction vector $\mathbf{u}_c$ is along the $-z$ axis, and finally rotating the scene about the z axis so that camera is in the desired orientation. This last operation can be facilitated by specifying a vector that we will call $\mathbf{v}_{up}$, and rotating the scene so that $\mathbf{v}_{up}$ lies in the upper half of the y-z plane. The figure below illustrates this transformation.



Summarizing, we do the following operations:

1. translate $\mathbf{x}_c$ to the origin so $\mathbf{x}'_c = \mathbf{0}$,
2. rotate about x and y so $\mathbf{u}'_c = \mathbf{u}'_z$, and
3. rotate about z so that $\mathbf{v}_{up}$ lies in the upper half of the $\mathbf{u}_y$-$\mathbf{u}_z$ plane.

So, the view transformation would be the product of a translation $T(-x_c, -y_c, -z_c)$ taking $\mathbf{x}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$ to the origin, and a rotation R, combining the three rotations above:

$$M_{view} = RT.$$

To understand how to do the rotations, remember first that we construct the camera coordinate frame as follows:

$$\begin{aligned} \mathbf{u}_x &= (\mathbf{u}_c \times \mathbf{v}_{up})/\|\mathbf{u}_c \times \mathbf{v}_{up}\|, \\ \mathbf{u}_y &= \mathbf{u}_x \times \mathbf{u}_c. \\ \mathbf{u}_z &= -\mathbf{u}_c. \end{aligned}$$

After rotation $\mathbf{x}' = R\mathbf{x}$, we want

$$\mathbf{u}'_x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{u}'_y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \text{ and } \mathbf{u}'_z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

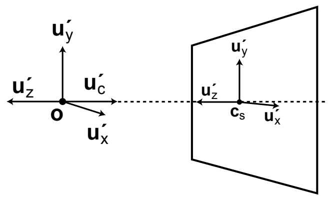
The matrix to do this is easy to find, if we remember that $\mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z$ are mutually orthogonal unit vectors. Thus, $\mathbf{u}_x \cdot \mathbf{u}_y = \mathbf{u}_x \cdot \mathbf{u}_z = \mathbf{u}_y \cdot \mathbf{u}_z = 0$ and $\mathbf{u}_x \cdot \mathbf{u}_x = \mathbf{u}_y \cdot \mathbf{u}_y = \mathbf{u}_z \cdot \mathbf{u}_z = 1$. Thus, if we construct a matrix whose rows are formed from $\mathbf{u}^t_x, \mathbf{u}^t_y, \mathbf{u}^t_z$ we will get the result we want:

$$R = \begin{bmatrix} \mathbf{u}^t_x \\ \mathbf{u}^t_y \\ \mathbf{u}^t_z \end{bmatrix},$$

since

$$R\mathbf{u}_x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, R\mathbf{u}_y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \text{ and } R\mathbf{u}_z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

10.2 Projection transform

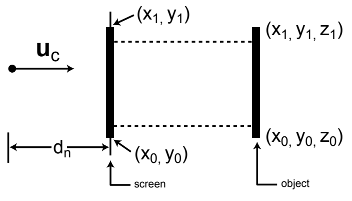


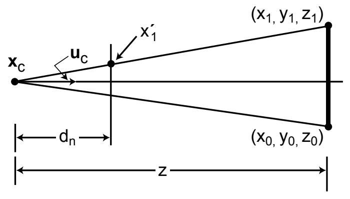
Now that our model is conveniently in camera coordinates, it is possible to determine a set of screen coordinates. The distance from the viewpoint to the view screen is known as the the *focal length* of the camera. In a real camera, the focal length is the distance from the lens at which distant objects come into focus on the camera backplane. In a pinhole camera, it is simply the distance to the view screen. If we define the camera's focal length to be d_n (distance to the near plane), then the screen center $\mathbf{c}_s = (0, 0, -d_n)$ and the x and y screen axes are given by $\mathbf{u}'_x$ and $\mathbf{u}'_y$, assuming that the screen is arranged so its normal is $\mathbf{u}'_z$ (i.e. it is perpendicular to the view direction $\mathbf{u}'_c$). Applying our view transform M_{view} to the entire scene, everything is in the same coordinate frame.

To do an orthographic projection of the scene onto the camera plane is now straightforward – we just discard the z coordinate of each vertex, as shown above to the right in a 2D plan view. To do a perspective projection, shown below to the right, we use the device of similar triangles:

$$x_1/z = x'_1/d_n$$
$$y_1/z = y'_1/d_n$$

Thus the transform is $\mathbf{x}' = \frac{d_n}{z}\mathbf{x}$.





10.3 Canonical view volumes

The view volume is the volume swept out by the screen through space in the projection system being used. For an orthographic projection, this is a rectangular solid, as shown in Figure 10.1. We use the distance d_n to denote the distance of the front face, or *near plane*, of the volume and d_f to denote an arbitrary, or *far plane* depth of the volume. Screen space cameras are generally set up to ignore (or clip away) any part of the scene not in the view volume. The width and height of the volume are determined by camera screen width w and height h.

In a perspective camera, the view volume has a frustum shape, as shown in Figure 10.2.

The idea of a canonical view volume is to provide a common frame of reference for processing after the projection is performed, which decouples shading and display of an image from the projection system used. The typical canonical view volume is a 2x2x2 cube aligned with the coordinate axes in viewing coordinates, and centered at the origin. The idea is that no matter what the viewing conditions are, after a projection transform is performed, all of the scene is transformed into this space, and anything in that scene that is not in the canonical view volume after that transformation is clipped away.

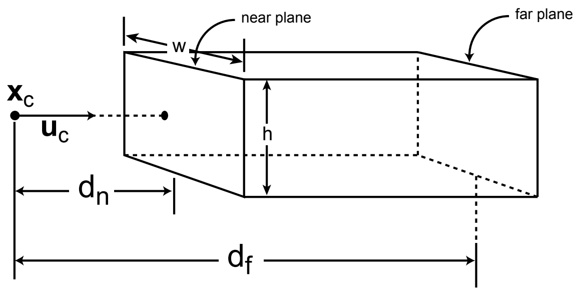


Figure 10.1: Parallel view volume.

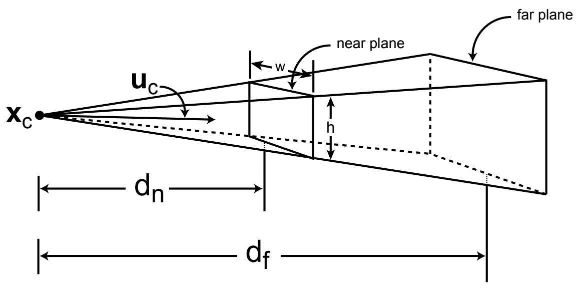


Figure 10.2: Perspective view volume.

We construct the projection transform by considering how it will affect the 8 corners of the original view volume, under whichever projection system is being employed.

10.3.1 Constructing the canonical orthographic view volume

To construct the canonical view volume under orthographic projection, corners $(w/2, h/2)$, $(-w/2, h/2)$, $(w/2, -h/2)$, and $(-w/2, -h/2)$ should be sent to $(1, 1)$, $(-1, 1)$, $(1, -1)$, and $(-1, -1)$ respectively (here z does not matter, as we are simply scaling in the x and y directions). In the depth (z) direction we want z_n to go to -1 and z_f to go to 1. We see that this is a scale

$$S = \begin{bmatrix} 2/w & 0 & 0 & 0 \\ 0 & 2/h & 0 & 0 \\ 0 & 0 & -2/(d_f - d_n) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

followed by a translation in the z direction to center the cube at the origin

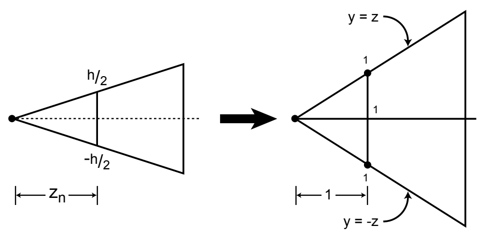
$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -(\frac{d_f+d_n}{d_f-d_n}) \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

So the final orthographic projection matrix to transform the scene into the canonical view volume is

$$P_{ortho} = TS = \begin{bmatrix} 2/w & 0 & 0 & 0 \\ 0 & 2/h & 0 & 0 \\ 0 & 0 & \frac{-2}{d_f-d_n} & -(\frac{d_f+d_n}{d_f-d_n}) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

10.3.2 Constructing the canonical perspective view volume

The construction of the canonical view volume under perspective projection follows a somewhat different derivation. First, we scale the perspective frustum so that its sides are $x = \pm z$, $y = \pm z$ and its front face is at $z = 1$:



$$S = \begin{bmatrix} 2/w & 0 & 0 & 0 \\ 0 & 2/h & 0 & 0 \\ 0 & 0 & -1/d_n & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

This is followed by a perspective transform. Remembering that by similar triangles, we showed that the perspective transform should scale vertex $\mathbf{x}$ by $\frac{d_n}{|z|}$ (note, that the absolute value signs are needed since in camera coordinates, all z values in the view volume are negative. Thus, a perspective transform should produce the result $\mathbf{x}' = \frac{d_n}{|z|}\mathbf{x}$. The matrix

$$\begin{bmatrix} d_n & 0 & 0 & 0 \\ 0 & d_n & 0 & 0 \\ 0 & 0 & -d_n & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

will transform $\mathbf{x}$ as follows:

$$P\mathbf{x} = P \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} d_n x \\ d_n y \\ -d_n z \\ -z \end{bmatrix}.$$

If this result is normalized into homogenous coordinates by scaling by $-1/z$, we have $\begin{bmatrix} -\frac{d_n}{z}x \\ -\frac{d_n}{z}y \\ d_n \\ 1 \end{bmatrix}$, which is the desired result. In general, a perspective transform has non-zero terms in the fourth row of the matrix and will transform points in the homogenous coordinates to points in 4-space that are not homogenous.

In order to place projected points back into homogeneous coordinates, the perspective matrix multiplication is followed by a normalization of each transformed point by dividing by its own w coordinate, to complete the perspective transform.

Now, to complete the transformation to the canonical view volume, we do a perspective transform combined with a scale and translation in z that will place z_n at -1 and z_f at 1:

$$P = \begin{bmatrix} d_n & 0 & 0 & 0 \\ 0 & d_n & 0 & 0 \\ 0 & 0 & s_z & t_z \\ 0 & 0 & -1 & 0 \end{bmatrix},$$

where the scale s_z and the translation t_z are yet to be determined.

Note that

$$P\mathbf{x} = P \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ s_z z + t_z \\ z \end{bmatrix} = \mathbf{x}'.$$

After normalizing by the w coordinate to place the result back in homogeneous coordinates, we have

$$\frac{1}{z}\mathbf{x}' = \mathbf{x}'' = \begin{bmatrix} x/z \\ y/z \\ s_z + \frac{1}{z}t_z \\ 1 \end{bmatrix}.$$

The requirements that $z_n \Rightarrow -1$ and $z_f \Rightarrow 1$ give us two linear equations (remember that because of the initial scale $z_n \Rightarrow 1$ and $z_f \Rightarrow -z_f/z_n$):

$$s_z + t_z = -1,$$
$$s_z + \frac{d_n}{d_f}t_z = 1.$$

Solving these equations simultaneously yields

$$t_z = \frac{2d_n d_f}{d_n - d_f},$$
$$s_z = \frac{d_n + d_f}{d_n - d_f}.$$

Thus, the final perspective matrix, transforming from camera space to the canonical view volume is

$$P_{persp} = PS = \begin{bmatrix} 2d_n/w & 0 & 0 & 0 \\ 0 & 2d_n/h & 0 & 0 \\ 0 & 0 & \frac{d_n+d_f}{d_n-d_f} & \frac{2d_nd_f}{d_n-d_f} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$