

Projection

Introduction to Computer Graphics

Arizona State University

Dianne Hansford

February 21, 2005

1 Introduction

The Graphics Pipeline involves several coordinate systems, namely

object \rightarrow world \rightarrow eye \rightarrow clip \rightarrow NDC \rightarrow window.

These coordinate transformations have been designed to optimize the algorithms applied to a primitive traveling down the pipeline. The discussion here focuses on the transformations from eye to clip coordinates and clip to *Normalized Device Coordinates (NDC)* coordinates¹. These two transformations prepare the vertices for the final projection into the window².

Therefore, this discussion assumes that your object lives in eye coordinates: consider yourself positioned at the origin looking down at your object positioned on or near the $-z$ -axis.

2 Projections in the Graphics Pipeline

Orthographic and perspective projections are the methods most commonly used in computer graphics. Figure 1 illustrates both methods. At the end of

¹We'll say "NDC coordinates" which I realize repeats a word, but that's just they way it's done!

²Funnily, this stage is called projection, but a projection as we normally think of it – a flattening – doesn't really occur until later in the pipeline

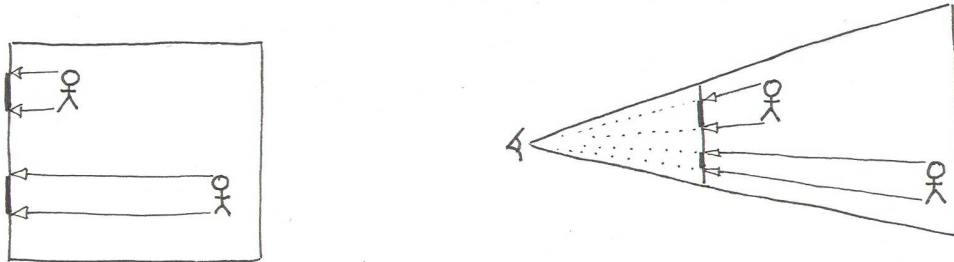


Figure 1: Orthographic and perspective projections: Left, the projectors of an orthographic projection are perpendicular to the viewing plane. Right, the projectors of a perspective projection pass through the center of projection. Foreshortening occurs with perspective.

the pipeline, the vertices will be projected into a *view plane*, which in turn is mapped to the window.

An orthographic projection projects all vertices in a direction perpendicular to the view plane. Architects and engineers tend to prefer orthographic images because distances and angles are preserved. However, these images can make it difficult to “see” an object as 3D and more than one view is commonly needed.

A perspective projection projects a vertex along a line defined by that vertex and the eye, or *center of projection (cop)*. This results in a foreshortening effect: objects farther away appear smaller than objects the same size that are closer to the eye. This foreshortening makes perspective very good for creating realistic images. Orthographic projection can be thought of as a special case of perspective: the center of projection is located infinitely far away.

In the graphics pipeline, specifying a projection has two purposes.

1. How is the geometry projected?
2. What is displayed?

Part of specifying a projection includes defining the parameters of a *viewing volume*. In Figure 1 observe that the orthographic viewing volume is a (rectangular) box and the perspective viewing volume is a *frustum*, or truncated pyramid. When we have our clip coordinates, all vertices outside of

the viewing volume will be clipped, or eliminated from the list of vertices continuing down the pipeline.

The transformation from eye to clip coordinates is controlled by the projection matrix in OpenGL. The key idea is that when we arrive in clip coordinates, regardless if we want an orthographic or perspective projection, all our vertices will live in a *normalized 4D cube*. (We’ll discuss how this works in the sections to follow.) The transformation from clip to NDC coordinates is the “perspective division” step which maps the 4D cube into the 3D cube with lower-left and upper-right vertices

$$\begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix},$$

respectively³. This process of bringing the geometry into this special volume is called *normalization*. This special volume is often times called the *canonical viewing volume*.

A parallel projection into the $z = 0$ plane is all that is needed to create the 2D vertices for rasterization. (This amounts to ignoring the z -value.) Importantly though, the z -values must be available at this stage in the pipeline for z -buffer hidden surface removal.

Let a homogeneous point in eye coordinates be $\bar{\mathbf{p}}_e$, its corresponding point in clip coordinates be $\bar{\mathbf{p}}_c$, and its corresponding (affine) point in NDC coordinates be \mathbf{p}_{ndc} .

3 Orthographic Projections

An orthographic projection is defined in OpenGL by the command

$$\text{glOrtho}(l, r, b, t, n, f),$$

and these parameters are illustrated in Figure 2. Since it is assumed we are in eye coordinates, we have defined a viewing volume box with lower-left and upper-right vertices

$$\begin{bmatrix} l \\ b \\ -n \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} r \\ t \\ -f \end{bmatrix},$$

respectively.

³The “lower-left” and “upper-right” is relative to our eye looking down the $-z$ -axis.

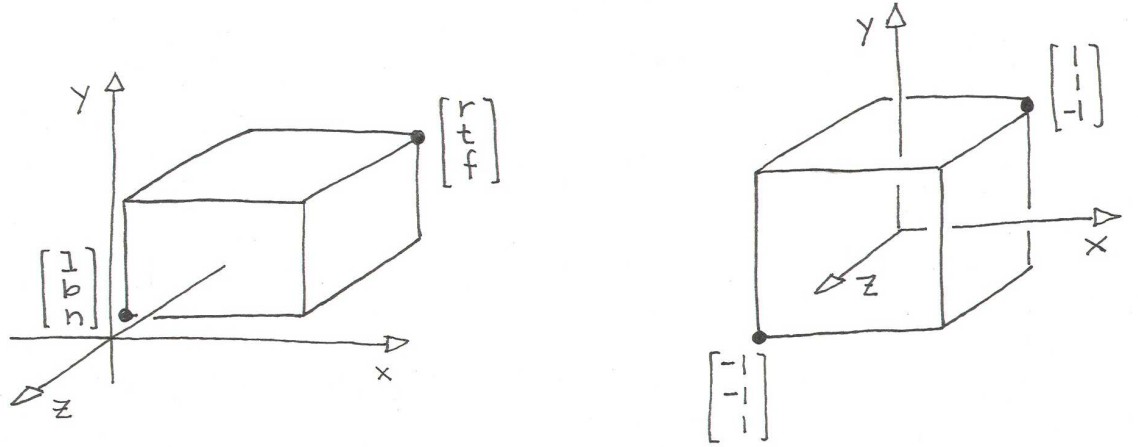


Figure 2: The glOrtho command defines a box by the parameters (l, r, b, t, n, f) and this box is mapped to a 3D box centered at the origin with each side of length 2.

The glOrtho command translates and scales this box (actually the vertices within) to its normalized 4D box with lower-left and upper-right vertices

$$\begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

respectively.

The center of the glOrtho box is

$$\mathbf{c} = \begin{bmatrix} 1/2(l+r) \\ 1/2(b+t) \\ 1/2(n+f) \end{bmatrix}.$$

Thus for an orthographic projection, the transformation from eye coordinates to clip coordinates is

$$\bar{\mathbf{p}}_c = \begin{bmatrix} 2/(r-l) & 0 & 0 & 0 \\ 0 & 2/(t-b) & 0 & 0 \\ 0 & 0 & 2/(n-f) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \bar{\mathbf{p}}_e. \quad (1)$$

The product of the matrices in (1) is the *orthographic projection matrix*

$$M_o = \begin{bmatrix} 2/(r-l) & 0 & 0 & -(r+l)/(r-l) \\ 0 & 2/(t-b) & 0 & -(t+b)/(b-t) \\ 0 & 0 & 2/(n-f) & -(f+n)/(n-f) \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

4 Perspective Projections

The parameters used to describe the frustum are illustrated in Figure 3. It is common practice to draw the frustum in a 2D view in the yz -plane. The center of projection is at the origin. The angle θ is called the *field of view* and n is the distance from the eye to the near plane. This plane corresponds to the view plane. Then f is the distance to the far plane. The height of the near plane, is denoted by h . These parameters are dependent, and they are related by

$$\tan(\theta/2) = \frac{h/2}{n}.$$

A human's field of view is roughly 65° . We can simulate a wide angle lens by creating a field of view that is large, for example 80° . And a telephoto lens can be simulated with a small field of view, for example 30° . This isn't an entirely correct analogy as the focal length of a camera is actually more closely related to the near distance. However since our hypothetical camera has a variable film size, relating the field of view to the lens is simpler.

In OpenGL there are two commands for defining the frustum. The easiest to use is

$$\text{gluPerspective}(\theta, a, n, f)$$

where a is the aspect ratio (width/height) of the near plane. From these parameters we can calculate any of the others that define the frustum:

$$h = 2n \tan(\theta/2) \quad \text{and then} \quad w = ha.$$

Commonly we choose the aspect ratio of the world to equal the aspect ratio of the viewport. This command assumes symmetry about the $-z$ -axis.

The `gluPerspective` command was built on top of the original command, `glFrustum` which has as input (l, r, b, t, n, f) . Interestingly, this command has the same input as `glOrtho`! Since we assume that the center of projection

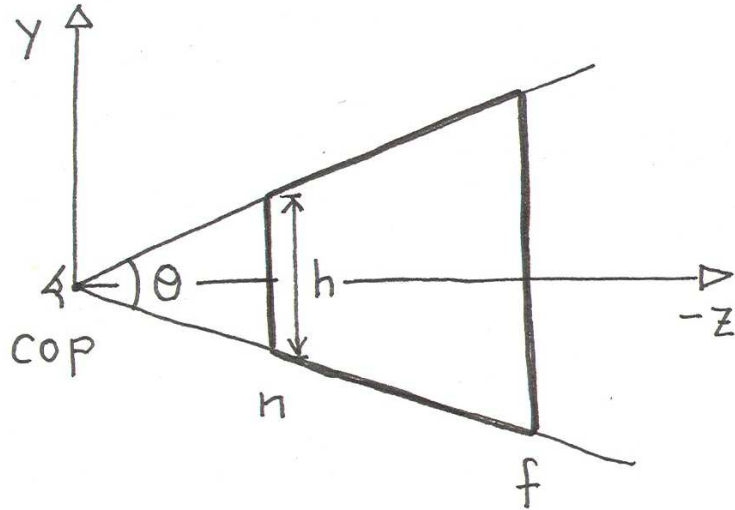


Figure 3: The `glPerspective` command defines a frustum by the parameters θ, n, f , and the aspect ratio of the near plane.

is at the origin, we can compute

$$\theta = 2 \cdot \arctan\left(\frac{h/2}{n}\right).$$

This command does not require symmetry about the $-z$ -axis and thus can be used for advanced techniques with perspective.

The action of the *perspective projection matrix* is to map the frustum to its normalized 4D box with lower-left and upper-right vertices

$$\begin{bmatrix} -n \\ -n \\ n \\ n \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} f \\ f \\ -f \\ f \end{bmatrix}, \quad (2)$$

respectively. As illustrated in Figure 4 this takes place in two steps: M_p is a matrix that maps the frustum to a box with the same parameters as the frustum (l, r, b, t, n, f) . Next, M_o is the orthographic mapping to take this general box to the normalized 4D box in (2).

The mapping M_p is a projective map which preserves the following properties.

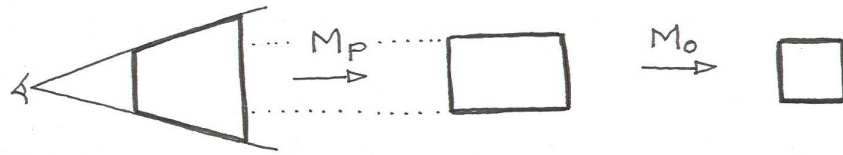


Figure 4: The two steps of the perspective projection matrix: M_p maps the frustum to a box and then M_o maps the box to a normalized box.

- Lines map to lines: if three points are collinear in eye coordinates, then they will be collinear in clip coordinates.
- Planes map to planes: if four points are coplanar in eye coordinates, then they will be coplanar in clip coordinates.
- The inverse of the map exists: this helps with screen picks – window coordinates can be transformed back through the pipeline to world coordinates.
- Relative z -depth is preserved: a point closer to your eye than another point in eye coordinates will also be closer in clip coordinates.

This perspective projection matrix takes the form:

$$M_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & (n+f)/n & -f \\ 0 & 0 & 1/n & 0 \end{bmatrix}.$$

Deriving this matrix is really beyond the scope of the class, however some insight into this is given in Section 6.

Let's look at M_p 's action on a point \mathbf{x} :

$$M_p \bar{\mathbf{x}} = \begin{bmatrix} x \\ y \\ z(\frac{n+f}{n}) - f \\ z/n \end{bmatrix}. \quad (3)$$

To understand the geometry, let's divide by the homogeneous “ w ” coordi-

nate, and look at the corresponding affine point

$$\mathbf{q} = \begin{bmatrix} x \cdot c/z \\ y \cdot n/z \\ n + f - fn/z \end{bmatrix}. \quad (4)$$

Let's examine this transformation. If $z = n$ then

$$\mathbf{q} = \begin{bmatrix} x \\ y \\ n \end{bmatrix},$$

illustrating that points in the near plane do not change. If $z = f$ then

$$\mathbf{q} = \begin{bmatrix} x \cdot n/f \\ y \cdot n/f \\ f \end{bmatrix},$$

and therefore points in the far plane stay in the far plane, however the x and y values do change. Just how they change is illustrated in Figure 5. Join the point in the frustum's far plane to the eye and record the point of intersection \mathbf{q}' in the near plane (view plane). Construct a line perpendicular to the near plane at \mathbf{q}' , and intersect this line with the far plane. This is \mathbf{q} 's position in the box.

Further examination of this mapping reveals that points farther from the eye are scaled more in x and y . The change in z is illustrated in Figure 5, and this follows in the same manner as outlined above for points in the far plane.

Summarizing, points in eye coordinates are mapped to points in clip coordinates as follows:

$$\bar{\mathbf{p}}_c = M_o M_p \bar{\mathbf{p}}_e.$$

In Appendix F in the OpenGL Red Book, the matrices M_o and M_p are given. The latter matrix has been "simplified" in the Red Book by taking advantage of the correspondence between homogeneous coordinates – the entire matrix is multiplied by n to eliminate the denominator in the (4,3) matrix element.

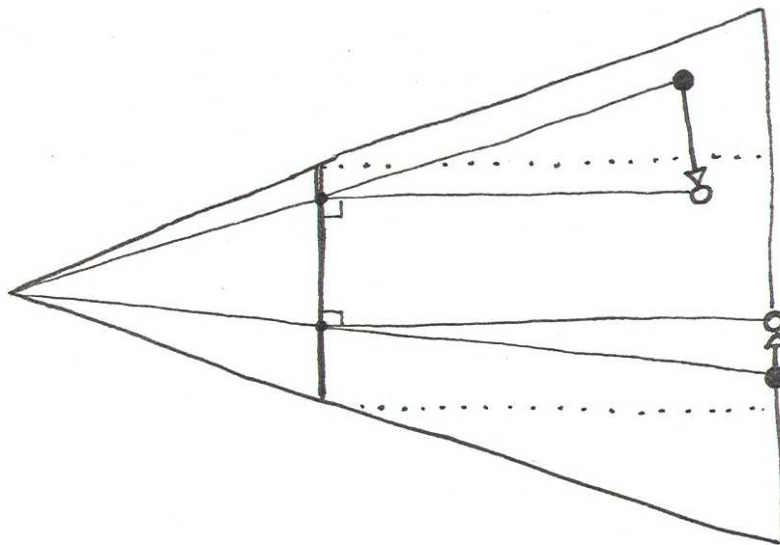


Figure 5: The mapping M_p “squishes” the frustum into the box as a projective map. A point is mapped to a line which is the parallel projector through the point’s image in the view plane. (Figure idea courtesy of P. Shirley [1].)

5 Perspective Division

Clipping is the process of removing all vertices (primitives) outside of the viewing volume. The vertices that remain are then projected into 3D by dividing by the homogeneous coordinate, thus

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \rightarrow \begin{bmatrix} x/w \\ y/w \\ z/w \end{bmatrix}.$$

This is called perspective division. And the points are now said to reside in NDC coordinates.

One might ask why clipping takes place in 4D, since working in affine spaces is much more intuitive. This interesting insight into the reasoning behind clipping in 4D was brought to my attention by P. Shirley [1].

To simplify the discussion, assume that we are dealing with positive near and far distances and z -values. In (3) we observed that a z -value, z_e , in eye coordinates is mapped by M_p to a (homogeneous) z -value, z_c , in clip coordinates:

$$z_c = z_e \frac{n + f}{n} - f.$$

Perspective division of (3) resulted in (4), and thus

$$z_{ndc} = n + f - fn/z_e, \tag{5}$$

the z -value in NDC coordinates. Figure 6 illustrates this function.

To this time, we have been interested in points with $z_e \in [n, f]$. But suppose some of geometry would lie outside the viewing volume. (This isn't uncommon, as we don't always want the entire world to be displayed.) The interesting aspect of Figure 6 is the mapping of points with z_e outside of the frustum (in z). Some points between our eye and the near plane are mapped behind our eye. Points behind our eye are mapped to points in front of our eye, and the closer they are to our eye, the farther away they are mapped. If a vertex is at our eye, it gets mapped to infinity.

Figure 7 illustrates that clipping after the perspective division (thus in NDC coordinates) can produce unwanted results. In this figure, a triangle has one vertex behind the eye, and therefore outside of the viewing volume. Perspective division would send this vertex to a place in front of the eye,

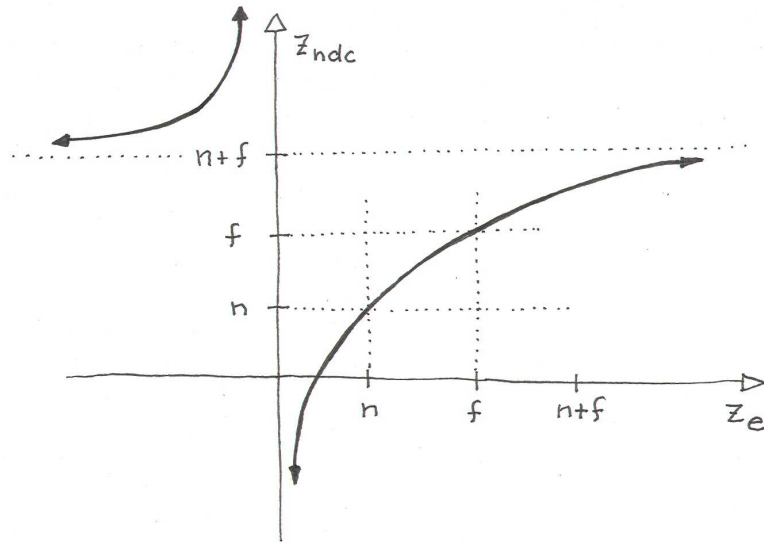


Figure 6: A graph of z_{ndc} as a function of z_e . This illustrates the change in z as a result of perspective division. (Figure idea courtesy of P. Shirley [1].)

past the far plane. Clipping at this point would result in a rectangular region with the far plane as an edge, which clearly is not correct. On the other hand, if we clip before perspective division, the vertex still be behind the eye, and clipping will produce a rectangular region with the near plane as an edge – the correct result.

6 Advanced: the Perspective Projection Matrix

The projective map M_p is a perspective map which preserves the cross ratio (cr) of four points, where

$$\text{cr}(\mathbf{a}, \mathbf{x}, \mathbf{y}, \mathbf{b}) = \frac{\text{ratio}(\mathbf{a}, \mathbf{x}, \mathbf{b})}{\text{ratio}(\mathbf{a}, \mathbf{y}, \mathbf{b})},$$

and if

$$\mathbf{x} = \alpha\mathbf{a} + \beta\mathbf{b},$$

then the

$$\text{ratio}(\mathbf{a}, \mathbf{x}, \mathbf{b}) = \beta/\alpha.$$

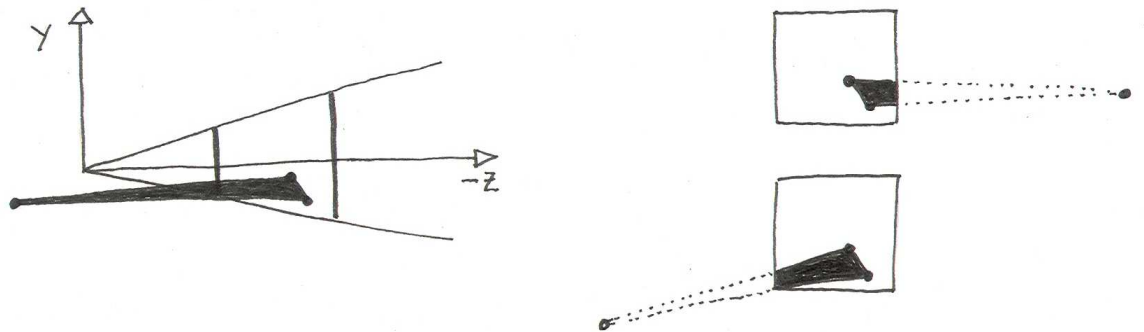


Figure 7: Clipping after perspective division could have unwanted side-effects if a primitive lives only partly within the viewing volume.

More information on cross ratios and projective maps may be found in [2].

A point \mathbf{p} in the frustum is mapped to \mathbf{q} in the box by satisfying the following equality

$$\text{cr}(\text{cop}, n, p_z, f) = \text{cr}(\infty, n, q_z, f).$$

Thus the center of projection is mapped to infinity, the near plane of the frustum is mapped to the near plane of the box, and likewise for the far plane.

7 References

- [1] P. Shirley, **Fundamentals of Computer Graphics**, A K Peters, Ltd, 2002.
- [2] G. Farin, **NURB Curve and Surfaces from Projective Geometry to Practical Use**, 2nd edition, A K Peters, Ltd., 1995.