U08181: Mathematics for Computer Graphics - Part II

Contents

1. Introduction

These notes accompany the lecture on Viewing.

2. Window to Viewport Mappings

The window to viewport mapping is a very important transformation in computer graphics. It is used to map one rectangular region into another, for example to map the coordinate system in which a 2D scene is defined onto a screen coordinate system for display. The SVG viewBox transformation is another example.

A rectangular window is mapped onto a rectangular viewport. See figure 2.1.

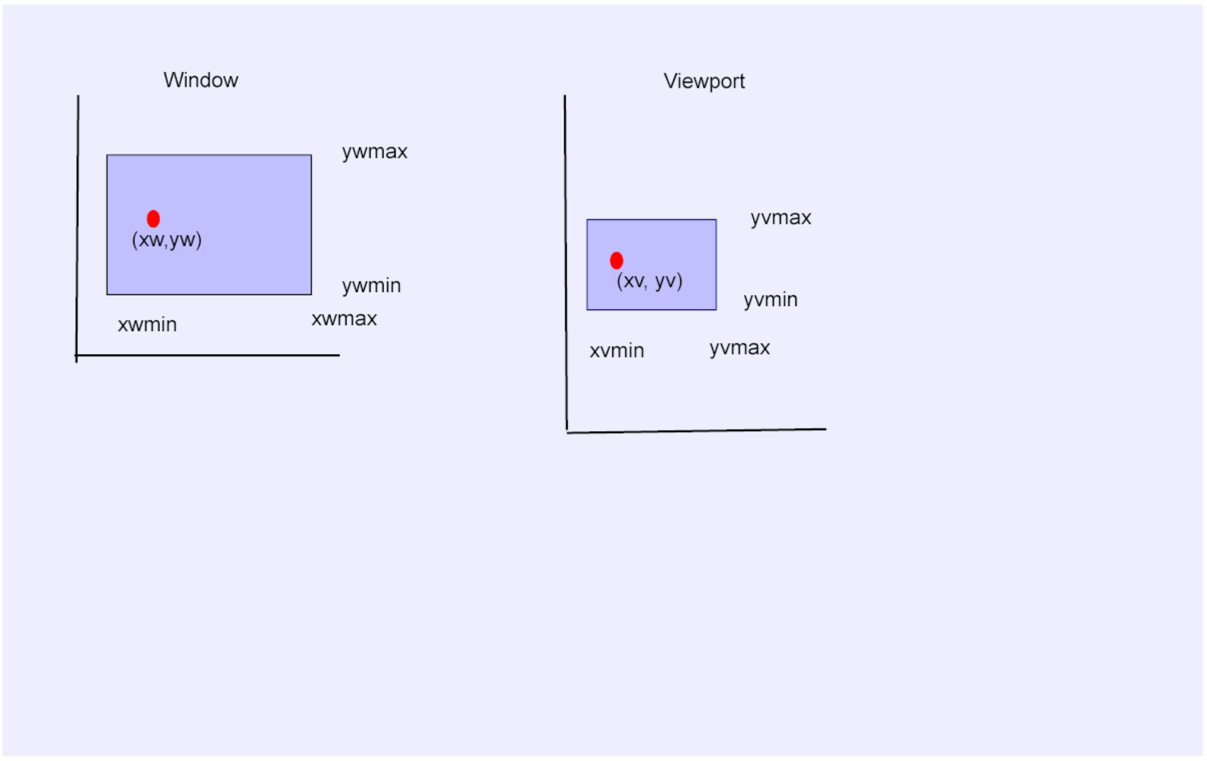

Figure 2.1: Simple Transformations

Suppose the limits of the window are xwmin to xwmax and ywmin to ywmax. Suppose the limits of the viewport are xvmin to xvmax and yvmin to yvmax. We want to find out the coordinates of the point in the viewport (xv,yv) corresponding to the point (xw, yw) in the window. One way to approach this is to spot that xv should be the same fraction of the distance from xvmin to xvmax as xw is from xwmin to xwmax.

The fractional distance from xwmin to xw is given by:

$$\frac{(xw - xwmin)}{(xwmax - xwmin)}$$

This is equal to the fractional distance from xvmin to xv:

$$\frac{(xv - xvmin)}{(xvmax - xvmin)} = \frac{(xw - xwmin)}{(xwmax - xwmin)}$$

Rearranging:

$$xv - xvmin = \frac{(xw - xwmin)(xvmax - xvmin)}{(xwmax - xwmin)}$$

$$xv = xvmin + \frac{(xw - xwmin)(xvmax - xvmin)}{(xwmax - xwmin)}$$

Similarly for yv:

$$yv = yvmin + \frac{(yw - ywmin)(yvmax - yvmin)}{(ywmax - ywmin)}$$

The next question is can this transformation be expressed in matrix form? The answer is yes. To see this we observe that the transformation consists of two translations and a scaling. The first transformation is a translation by (-xwmin, -ywmin). You can think of this as moving the window to the origin. The scalng transformation is then by the ratio of viewport width to window width and viewport height to window height. The final transformation is a translation to move the viewport from the origin to the point (xvmin, yvmin).

$$\begin{pmatrix} xv \\ yv \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & xvmin \\ 0 & 1 & yvmin \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -xwmin \\ 0 & 1 & -ywmin \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} xw \\ yw \\ 1 \end{pmatrix}$$

where

$$s_x = \frac{xvmax - xvmin}{xwmax - xwmin}$$

$$s_y = \frac{yvmax - yvmin}{ywmax - ywmin}$$

Perform the matix multiplications to convince yourself that this works!

3. 3D Transformations

3.1 3D Geometric transformations

Part 1 of this Primer introduced **homogeneous coordinates**, the use of a 3D coordinate system to describe 2D geometry. Homogeneous coordinates enabled us to represent rotation, scaling and translation transformations (and combinations of these transformations) as 3x3 matrices in a very convenient way. In a similar way we can use a 4D coordinate system to represent 3D points. Using a column vector formulation, we can represent a scaling transformation in 3D as:

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix}$$

This expands to:

$$x_2 = s_x x_1$$
$$y_2 = s_y y_1$$
$$z_2 = s_z z_1$$

as we would expect.

Rotations follow a similar pattern to 2D rotations, except that we can now rotate about any axis in 3D space. The easiest case to consider is where the rotation is about one of the coordinate axes. Rotation about any arbitrary axis in 3D space can be expressed as a combination of translation operations and rotations about the coordinate axes.

Consider rotation about the z axis. The z coordinates of points remain unchanged; the x and y coordinates are transformed by a 2D rotation in the x-y plane. Thus the part of the matrix representing the x-y transformation is the same as a 2D rotation matrix.

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix}$$

Rotations around the y and z axes take a similar form and can be obtained by cyclically replacing the coordinate parameters x, y and z. Thus a rotation about the x axis (x coordinates do not change this time, so replace z by x, x by y and y by z):

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix}$$

Translation transformations are represented by:

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix}$$

If you are interested in following up the mathematical background to 3D geometric transformations, look at Hearn and Baker chapter 5-9 onwards.

2.2 3D Viewing transformations

Viewing in 3D is rather more complicated than viewing in 2D. In 2D graphics we normally select a rectangular region of the 2D scene to view in a rectangular area of the display space. In its simplest form this is just a window-to-viewport transformation of the kind described earlier. A slightly more complicated case is when the rectangular window is not aligned with the coordinate axes, but this is easily addressed by applying a rotation to bring the window into alignment with the axes, before applying the window to viewport mapping.

In 3D we typically deal with **planar projections** where we project the 3D scene onto a 2D screen (called the **view plane** or **projection plane**). Conceptually we pass a projector through each point of the object and find where it intersects the projection plane. The intersection point is then the point in the projection plane corresponding to that point in the object. We saw that there are two types of projections:

- **parallel projections**: the projectors emanate from a point at infinity, i.e. all the projectors are parallel;
- **perspective projections**: the projectors emanate from a projection reference point (a finite point, a point not at infinity); the projectors are not parallel to each other in this kind of projection.

Human eyesight sees the world in perspective. If objects A and B are physically the same size, if A is further from the eye than B, A will appear smaller than B.

We can represent parallel and perspective transformations using matrices and 4D homogeneous coordinates, but first we need to say a little more about homogeneous coordinates. One of the issues in 3D projections is that we need to be able to talk about parallel lines as well as points. If we handle clipping operations in 3D properly, we also need to be able to talk about a case where clipping a line removes a section from a line and leaves the two bits that go off to infinity (to put it crudely!). Not all graphics systems get this right! We can represent the point (x, y, z) in 3D by the point (x, y, z, 1) in 4D. Suppose we generalise the 4D representation, Why limit the 4th coordinate to 1? Suppose instead we allow (x, y, z, w). It turns out that we can accommodate parallel lines by taking w=0 and ordinary points by all other values of w. Thus we say that the 4D homogeneous coordinates (x, y, z, w) represent the 3D point (x/w, y/w, z/w) providing that w is not 0. You are not expected to learn the details of this, but there is no harm in telling rather more of the truth about the mathematics than is sometimes told! Effectively what these coordinates do is to add things called **ideal points** to the Euclidean plane. Ideal points are points at infinity, points where parallel lines meet. In perspective transformations, parallel lines do meet, so it is important to be able to talk about the points at which parallel lines meet in the same framework as ordinary points. This is the reason why homogeneous (of the same kind) coordinates are so-called. For each family of parallel lines (lines which are parallel to each other) a single ideal point is attached to the plane. Also we can tell which points are ideal points and which are not by looking at the value of w. And clearly there may be transformations that map Euclidean points to ideal points.

So how does this help us to represent projection transformations as matrices?

Parallel projections are straightforward. The simplest case is an orthographic projection onto the x-y plane, i.e. the projectors are parallel to the z axis. The trick here is to transform the scene so that the projection plane is the z=0 plane. This can be done by a combination of rotation and translation operations. The translation is just a translation along the direction defined by the surface normal to the projection plane (the direction in which it is pointing) by a distance that is the shortest distance from the plane to the origin of the coordinate system. Having done this, the projection plane then passes through the origin, but needs to be rotated about the z and y axes to make it coincident with the x-y plane. The two rotations essentially get the projection plane normal pointing down the z axis. Rotation about the z axis gets the normal into the x-z plane and rotation about y aligns it with the z axis. (The Carlbom and Paciorek paper, appendix A.8, explains this if you are interested in the details.)

Projection onto the x-y plane is then represented by the matrix:

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

This matrix just replaces all z coordinates by 0, i.e. all points are projected onto the z=0 plane. x and y coordinates are unchanged.

Perspective projectins are a bit more complicated. Suppose we have applied translations and rotations as before so that the view plane is the x-y plane and the projection reference point lies on the z axis as shown in figure 2.2.

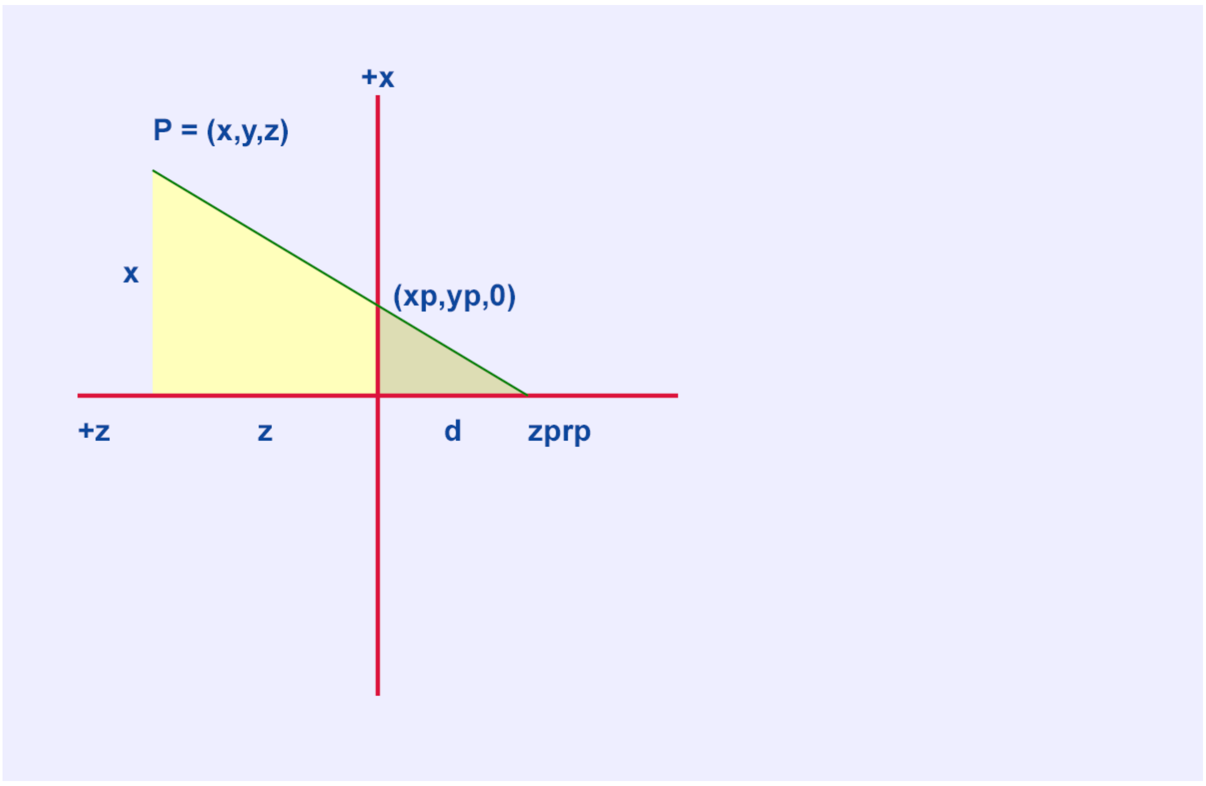

Figure 2.2: Perspective Projection

The problem is to find the point (xp,yp,0) to which the point (x, y, z) projects. The diagram shows the x-z plane. Observe that the larger shaded triangle is similar to the smaller shaded triangle. Hence by similar triangles:

$$\frac{xp}{d} = \frac{x}{z+d}$$

$$xp = x\left(\frac{d}{z+d}\right)$$

Similarly:

$$yp = y\left(\frac{d}{z+d}\right)$$

All we have to do now is to express this as a 4x4 matrix applied to the homogeneous representation of the point P. The following matrix achieves the desired result.

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

If we left out the "1" we would have a matrix that represents an orthographic projection. Including the off-diagonal "1" generates the term (z+d). We expand this below.

$$\begin{pmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & d \end{pmatrix}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} xd \\ yd \\ 0 \\ z+d \end{pmatrix}$$

Recalling the mapping from homogeneous to ordinary 3D Cartesian Coordinates, the resulting point corresponds to the 3D point:

$$\begin{pmatrix} xd/(z+d) \\ yd/(z+d) \\ 0 \end{pmatrix}$$

as required.