7. Projection Transformations

- Projection Transformations

Projections transform three dimensional eye coordinates into points in three dimensional clip coordinates

- Three dimensional object coordinates are projected onto the projection plane

- Determines how objects are projected onto the screen

The projection matrix also defines the viewing volume

The viewing volume determines which objects or parts of objects are projected onto the screen

The viewing volume is an six-sided enclosure defined by clipping planes

- Primitives outside the viewing volume are not displayed
The type of projection is determined by the relationship between the *center of projection* and the *projection plane*

- The relative size of an object is based upon the proportion of a plane that the object covers, where the plane is perpendicular to the viewing direction

In OpenGL there are two basic types of projections:

- Perspective
- Orthographic (Parallel)
• **Perspective Projection**

In a perspective projection the center of projection is a finite distance from the projection plane.

As the distance of an object from the projection plane is increased, the size of its projection on the projection plane will decrease.

- This change in size gives perspective projections what is know as *perspective foreshortening*.

- The effect is similar to photographic systems and the human eye.

- Perspective projection distorts the exact shape of objects.
Simplified Perspective Projection Transformation

The point \( p = (x, y, z) \) is projected to the point \( p' = (x', y', z') \) as follows:

- Let \( d \) be the distance from the center of projection to the projection plane, then

\[
\frac{x'}{d} = \frac{x}{z} \quad (7.1)
\]
- From the above figure

\[- \frac{yx}{d} = \frac{y}{z} \]  \hspace{1cm} (7.2)

Solving for \(x'\) and \(y'\) in 7.1 and 7.2 produces

\[x' = -d\left(\frac{x}{z}\right) \quad \text{and} \quad y' = -d\left(\frac{y}{z}\right) \]  \hspace{1cm} (7.3)

- \(d\) is a scale factor applied to \((x, y, z)\)
- Division by \(z\) causes distant objects to appear smaller
The projection transformation must be reversible
- Vertices must be recoverable from their projection

Recall that homogeneous coordinates have the form

\[(wx, wy, wz, w) \quad (7.4)\]

- All components are divided by the fourth element during perspective division

The matrix

\[
M = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & -\frac{1}{d} & 0
\end{bmatrix}
\quad (7.5)
\]

transforms the vertex

\[
p = \begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\quad (7.6)
\]

to

\[
p = \begin{bmatrix}
x \\
y \\
z \\
-\frac{z}{d}
\end{bmatrix}
\quad (7.7)
\]

Perspective division produces the results in 7.3 and 7.4
• **Orthogonal (Parallel) Projections**

In a parallel projection the center of projection is at an infinite distance from the projection plane.

- As the distance of an object from the projection plane is increased the size of its projection remains constant
- The projection can be used for exact measurements
- Parallel lines remain parallel
- Angles are not preserved
Orthogonal projections are a special case of perspective projections

- Vertices projected onto the projection plane retain their x and y values

\[ x' = x \quad \quad y' = y \]  \hspace{1cm} (7.8)

- If the projection plane is at \( z = 0 \)

\[ z' = 0 \]  \hspace{1cm} (7.9)

The homogenous projection matrix is defined by

\[
\begin{bmatrix}
    x' \\
    y' \\
    z' \\
    1
\end{bmatrix} =
\begin{bmatrix}
    1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 0 & 1 \\
    0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    z \\
    1
\end{bmatrix}
\]  \hspace{1cm} (7.10)

- Perspective division is unnecessary and has no effect
**Viewing Volumes**

The viewing volume of a perspective projection is a *frustum* or truncated pyramid

- The volume is bounded by the *near*, *far*, *right*, *left*, *top* and *bottom* clipping planes.

- Clipping requires that each primitive be checked against all six planes of the viewing volume

- Primitives which are inside all six planes are projected; all others are not
The viewing volume of a parallel (orthogonal) projection is a rectangular box.
• **Defining Projection Matrices and Viewing Volumes in OpenGL**

In OpenGL the viewing volume can be defined using

\[
\text{void glFrustum( GLdouble } l, \text{ GLdouble } r, \text{ GLdouble } b, \\
\text{ GLdouble } t, \text{ GLdouble } n, \text{ GLdouble } f);\]

where

- \((l, b, -n)\) and \((r, t, -n)\) specify in xyz coordinates the lower-left and upper-right corners of the near clipping plane
- \(n\) and \(f\) give the distances from the viewpoint to the near and far clipping planes, respectively

\[
\text{glFrustum generates the following projection matrix}
\]

\[
\begin{bmatrix}
\frac{2n}{r-t} & 0 & \frac{r+1}{t-b} & 0 \\
0 & \frac{2n}{t-b} & \frac{t-b}{f+n} & -\frac{2fn}{f-n} \\
0 & 0 & \frac{f-n}{f-n} & 0 \\
0 & 0 & -1 & 0
\end{bmatrix}
\] (7.11)

- This matrix is multiplied times the modelview matrix to produce *clip coordinates*
In OpenGL a *perspective view* volume can also be set using

```c
void gluPerspective( GLdouble fovy, GLdouble aspect,
                     GLdouble near, GLdouble far );
```

where

- `fovy` is the angle of the field of the view in the xz plane between 0 and 180.0 degrees
- `aspect` is ratio of width to height
- `near` and `far` specify the distances from the viewpoint to the near and far clipping planes, respectively

![Diagram](image)
- gluPerspective produces a projection matrix similar to glFrustum using multiple OpenGL commands

To create an orthographic parallel viewing volume in OpenGL use

```c
glOrtho( GLdouble left, GLdouble right, GLdouble bottom,
        GLdouble top, GLdouble near, GLdouble far );
```

where

- `(left, bottom, -near)` and `(right, top, -near)` define the lower-left and the upper-right corners of the near clipping plane

- `(left, bottom, -far)` and `(right, top, -far)` define the lower-left and the upper-right corners of the far clipping plane

- These corners are mapped directly to the viewport
• The Projection Matrix Stack

The current projection matrix describing the viewing volume is stored on top of the projection matrix stack

- To enable projection transformation changes

```
glMatrixMode( GL_PROJECTION );
```

- The stack is normally cleared using

```
glLoadIdentity();
```

- The projection matrix stack has a maximum depth of at least two (2)