Viewing and Projection
Graphics Pipeline

• Consider our program as a pipeline

• At each step, we manipulate the data

• Our goal is to output the data in screen coordinates
Graphics Pipeline

- Object coordinates
- Camera coordinates
- Clip coordinates

Vertices $\rightarrow T_m \rightarrow T_w \rightarrow T_c^{-1} \rightarrow P \rightarrow$ Vertices

Matrix multiplication: $PT_c^{-1}T_wT_m$
Positioning the Camera

• Initially, the object and camera frames are the same (the identity matrix)

• The camera is located at the origin and points in the negative $z$ direction

• We place the camera through the Model-view matrix
The LookAt Function

- mat4 LookAt(vec4& eye, vec4& at, vec4& up)
- eye: The position of the camera
- at: The position the camera looks at
- Up: The upside (y) direction of the camera
The LookAt Function

• Translate the camera to eye and change frame

mat4 LookAt(vec4& eye, vec4& at, vec4& up )
{
  vec4 n = normalize(eye - at);
  vec4 u = normalize(cross(up,n));
  vec4 v = normalize(cross(n,u));
  vec4 t = vec4(0.0, 0.0, 0.0, 1.0);
  mat4 c = mat4(u, v, n, t);
  return c * Translate( -eye );
}
Viewing

• Perspective vs. Orthographic
Orthogonal Projections

• How can we construct $P$?

$$x_p = x, \ y_p = y, \ z_p = 0$$

• In matrix form:

$$(x_p, y_p, z_p, 1)^T = MI(x, y, z, 1)^T$$
Orthogonal Projections

- How can we construct $P = MI$?

\[(x_p, y_p, z_p, 1)^T = MI(x, y, z, 1)^T\]

- Where $I$ is the identity matrix and

\[
M = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
Orthogonal Projections

• Take a finite portion of the projection plane

• The default **view volume** is a cube with sides of length 2 centered at the origin

• Namely, the planes $x = \pm 1, y = \pm 1, z = \pm 1$, determine the **clipping volume**
Orthogonal Viewing

- We can specify a different view volume

```
mat4 Ortho(left, right, bottom, top, near, far)
```

Values are in Camera coordinates
Projection Normalization

- The graphics card has a fixed viewing volume.

- Thus, we transform our viewing volume into the canonical one.
Projection Normalization

• How to normalize the viewing volume?

  – Translate: $T\left(\frac{\text{right}+\text{left}}{2}, \frac{\text{bottom}+\text{top}}{2}, \frac{\text{near}+\text{far}}{2}\right)$

  – Scale: $S\left(\frac{2}{\text{right}−\text{left}}, \frac{2}{\text{top}−\text{bottom}}, \frac{2}{\text{near}−\text{far}}\right)$

• In matrix form:

$$ST = \begin{pmatrix}
\frac{2}{\text{right}−\text{left}} & 0 & 0 & \frac{\text{left}+\text{right}}{\text{right}−\text{left}} \\
0 & \frac{2}{\text{top}−\text{bottom}} & 0 & \frac{\text{top}+\text{bottom}}{\text{top}−\text{bottom}} \\
0 & 0 & \frac{2}{\text{far}−\text{near}} & \frac{\text{far}+\text{near}}{\text{far}−\text{near}} \\
0 & 0 & 0 & 1
\end{pmatrix}$$
Summary: Orthogonal Projection

Projection transformation

\[ P = MST \]

- **T**: Translate
- **S**: Scale
- Normalize
- **M**: Project
  - Orthographic projection

Projection transformation
Perspective Viewing

• The user can specify a viewing frustum:

\[
\text{mat4 Frustum(left, right, bottom, top, near, far)}
\]
Perspective Viewing

- Given a viewing frustum, we need to transform it to the canonical view-volume.
Frustum to Canonical View Volume

• Step 1: shear to symmetrize the frustum
• $Z$ near and far do not change
Frustum to Canonical View Volume

• Step 1: shear to symmetrize the frustum
• Z near and far do not change

\[ H = \begin{pmatrix}
1 & 0 & \frac{\text{left} + \text{right}}{-2\text{near}} & 0 \\
0 & 1 & \frac{\text{top} + \text{bottom}}{-2\text{near}} & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix} \]
Frustum to Canonical View Volume

- Step 2: scale the frustum such that the sides intersect the projection plane at a 45-degree angle.
- Z near and far still do not change.

\[ S = \begin{pmatrix} -2\text{near} & 0 & 0 \\ \text{right} - \text{left} & 0 & 0 \\ \text{top} - \text{bottom} & 0 & 1 \end{pmatrix} \]
Frustum to Canonical View Volume

• After step 2, we have a frustum with COP at the origin, 90 degrees FOV and bounded by the planes $x = \pm z, y = \pm z$.
• Visualization for projection plane at $z = -1$:
Frustum to Canonical View Volume

• Step 3: transform to the canonical view volume

• $z$ near, far should be transformed to the range $[-1,1]$ (mind the sign, the camera is pointing in the negative $z$ direction)

• $x$, $y$ coordinates should be projected to the projection plane
Perspective Division

• Example: the projection plane is at $z = d = -1$ ($near = 1$)

$$x_p = \frac{x}{z/-1}, \quad y_p = \frac{y}{z/-1}, \quad z_p = -1$$
Frustum to Canonical View Volume

• Step 3: transform to the canonical view volume

\[ N = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \alpha & \beta \\ 0 & 0 & -1 & 0 \end{pmatrix} \]

• Applying N on \((x, y, z, 1)^T\) and dividing by the 4\(^{th}\) coordinate gives:

\[ \left(-\frac{x}{z}, -\frac{y}{z}, -\alpha - \frac{\beta}{z}\right)^T \]

1 for \(z = -\text{near}\)
-1 for \(z = -\text{far}\)
Frustum to Canonical View Volume

• We choose

\[
\alpha = -\frac{near + far}{near \, - \, far}, \\
\beta = -\frac{2 \times near \times far}{near \, - \, far}
\]

• Then, our frustum is transformed into the default viewing volume (i.e., bounded by the planes \(x = \pm 1, \ y = \pm 1, \ z = \pm 1\))
Perspective Projection

• The final perspective projection matrix is:

\[ P = NSH \]

• \( P \) transforms an arbitrary frustum to the canonical view volume
Perspective Viewing

• We can also define a frustum by the aspect ratio and FOVY (field of view, y axis)

\[ \text{aspect} = \frac{w}{h} \]

• \text{mat4 Perspective(fovy, aspect, near, far)}
Viewport Transformation

• How to transform into screen coordinates?

\[ r = \frac{w}{2} (x + 1), \ s = \frac{h}{2} (y + 1). \]
Complete Graphics Pipeline

Object coordinates → Camera coordinates → Clip coordinates → Normalized device coordinates → Screen coordinates

$V \rightarrow \text{Model-view} \rightarrow \text{Projection} \rightarrow \text{Perspective division} \rightarrow \text{Viewport} \rightarrow V$
Suggested Readings

• Interactive Computer Graphics, Chapter 4