Viewing and Projection

Images were taken from the book: “Interactive Computer Graphics” by Angel and Shreiner
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

Vertices → $T_m$ → $T_w$ → $T_c^{-1}$ → $P$ → Vertices

- **Object coordinates**
- **Model-view transformation**
- **Camera coordinates**
- **Projection transformation**
- **Clip coordinates**
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

- \texttt{mat4 LookAt(vec4& eye, vec4& at, vec4& up)}

  - \textit{eye}: The position of the camera
  - \textit{at}: The position the camera looks at
  - \textit{Up}: The upside (\(y\)) direction of the camera
The LookAt Function

• Translate the camera to eye and change frame

```c
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:

$\begin{pmatrix} x_p, y_p, z_p, 1 \end{pmatrix}^T = M I (x, y, z, 1)^T$
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

• \text{mat4 Ortho(left, right, bottom, top, near, far)}

Values are in \text{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: \( T \left( \frac{right+left}{2}, -\frac{top+bottom}{2}, \frac{near+far}{2} \right) \)
  
  – Scale: \( S \left( \frac{2}{right-left'}, \frac{2}{top-bottom'}, \frac{2}{near-far'} \right) \)
  
• In matrix form:
  
  \[
  ST = \begin{pmatrix}
  \frac{2}{right - left} & 0 & 0 & -\frac{left + right}{right - left} \\
  0 & \frac{2}{top - bottom} & 0 & -\frac{top + bottom}{top - bottom} \\
  0 & 0 & \frac{2}{far - near} & -\frac{far + near}{far - near} \\
  0 & 0 & 0 & 1
  \end{pmatrix}
  \]
Summary: Orthogonal Projection

Projection transformation

\[ T \rightarrow S \rightarrow M \]

- Translate
- Scale
- Normalize
- Project

Orthographic projection
Perspective Projections

• How can we construct \( P \) for \( d = -1 \)?

\[
x_p = \frac{x}{z/-1}, \quad y_p = \frac{y}{z/-1}, \quad z_p = -1
\]
Perspective Projections

• How can we construct $P$?

$$P = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & -1 & 0
\end{pmatrix}$$

Perspective division
Perspective Projections

- $P$ corresponds to a frustum with COP at the origin, 90 degrees FOV and bounded by the planes $x = \pm z, y = \pm z$. 
Perspective Viewing

• We can specify a different viewing frustum

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

- We can specify a different viewing frustum

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

- `mat4 Perspective(fovy, aspect, near, far)`
Perspective Normalization

- $P$ is independent of the far clipping plane
- We take
  $$P = \begin{pmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & \alpha & \beta \\
  0 & 0 & -1 & 0
  \end{pmatrix}$$
- Thus, $(x, y, z, 1)^T$ goes (after division) to
  $$\left(\frac{-x}{z}, \frac{-y}{z}, -\alpha - \frac{\beta}{z}\right)^T$$
Perspective Normalization

• 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 \( x = \pm 1, \ y = \pm 1, \ z = \pm 1 \))
How to transform into screen coordinates?

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