Viewing and Projection
The topics

• Interior parameters
  • Projection type
  • Field of view
  • Clipping
  • Frustum...

• Exterior parameters
  • Camera position
  • Camera orientation
Transformation Pipeline

- Local coordinate
  - Local->World
  - World coordinate
    - World->Eye
    - Eye coordinate
      - Projection Matrix
        - Clip coordinate
          - others
            - Screen coordinate

- ModelView Matrix
Projection

• The projection transforms a point from a high-dimensional space to a low-dimensional space.

• In 3D, the projection means mapping a 3D point onto a 2D projection plane (or called image plane).

• There are two basic projection types:
  • Parallel: orthographic, oblique
  • Perspective
Orthographic Projection

Image Plane

Direction of Projection

$z = k$

$$
\begin{bmatrix}
  x \\
  y \\
  k \\
  1
\end{bmatrix}
= 
\begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 0 & k \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix}
$$
Orthographic Projection
Oblique Projection

Image Plane

Direction of Projection
Properties of Parallel Projection

- Definition: projection directions are parallel.
- Doesn’t look real.
- Can preserve parallel lines
Properties of Parallel Projection

• Definition: projection directions are parallel.

• Doesn’t look real.

• Can preserve parallel lines

• Can preserve ratios

\[ s : t = s' : t' \]
Properties of Parallel Projection

- Definition: projection directions are parallel.
- Doesn’t look real.
- Can preserve parallel lines
- Can preserve ratios
- **CANNOT** preserve angles
Properties of Parallel Projection

• Definition: projection directions are parallel.

• Doesn’t look real.

• Can preserve parallel lines
• Can preserve ratios
• **CANNOT** preserve angles

• Often used in CAD, architecture drawings, when images can be used for measurement.
Properties of Parallel Projection

• No foreshortening
Perspective Projection

- Perspective projection has foreshortening:
Perspective Projection

• Images are mapped onto the image plane in different ways:

![Diagram of perspective projection]
Perspective Projection

- Images are mapped onto the image plane in different ways:

![Diagram showing perspective projection]

An image (640*640)

Center of Projection

Image Plane
Perspective Projection

• Angle of view tells us the mapping from the image to the image plane:
Perspective Projection

- Not everything will be displayed.
Perspective Projection

• The frustum of perspective projection looks like:
Ortho Projection

• What’s the frustum of orthographic projection?
Homogenous Coordinates

• In general, the homogeneous coordinate system is define as:

\[
\begin{bmatrix}
wx \\
wz \\
w
\end{bmatrix} = 
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]

In homogeneous space

• For example,

\[
\begin{bmatrix}
2 \\
1 \\
4 \\
0.5
\end{bmatrix} = 
\begin{bmatrix}
\cdot \\
\cdot \\
\cdot \\
\cdot
\end{bmatrix}
\]
Homogenous Coordinates

- Both homogenous vectors and matrices are scalable:

\[
\begin{bmatrix}
wx \\
wz \\
w
\end{bmatrix}
= 
\begin{bmatrix}
x \\
y \\
z \& 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
wm_{00} & wm_{01} & wm_{02} & wm_{03} \\
wm_{10} & wm_{11} & wm_{12} & wm_{13} \\
wm_{20} & wm_{21} & wm_{22} & wm_{23} \\
wm_{30} & wm_{31} & wm_{32} & wm_{33}
\end{bmatrix}
= 
\begin{bmatrix}
m_{00} & m_{01} & m_{02} & m_{03} \\
m_{10} & m_{11} & m_{12} & m_{13} \\
m_{20} & m_{21} & m_{22} & m_{23} \\
m_{30} & m_{31} & m_{32} & m_{33}
\end{bmatrix}
\]
Perspective Projection

- Derivation:

$$\begin{bmatrix}
x' \\
y' \\
-d \\
1
\end{bmatrix} = \begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}$$
Perspective Projection

• Derivation:

\[
\begin{bmatrix}
-xd/z \\
-yd/z \\
-d \\
1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]
Perspective Matrix

• Derivation:

\[
\begin{bmatrix}
-\frac{xd}{z} \\
-\frac{yd}{z} \\
-d \\
1
\end{bmatrix}
= \begin{bmatrix}
d \\
d \\
-d \\
1
\end{bmatrix}
= \begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]

How?

The z axis now becomes useless...
OpenGL Perspective Matrix

• In practice, OpenGL uses the z axis for depth test. Its matrix looks like this:

\[
\begin{bmatrix}
  -x/z \\
  -y/z \\
  -a-b/z \\
  1
\end{bmatrix} =
\begin{bmatrix}
  d \\
  d \\
  a \\
  b \\
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix}
\]

The depth value now can be used for depth test. We will discuss this in more details later...
Vanishing Point

• Given a ray:

\[
\begin{bmatrix}
p_x \\
p_y \\
p_z \\
1
\end{bmatrix}
\rightarrow
\begin{bmatrix}
p_x + tn_x \\
p_y + tn_y \\
p_z + tn_z \\
1
\end{bmatrix}
\]

• Its projection:

\[
\begin{bmatrix}
d(p_x + tn_x) \\
d(p_y + tn_y) \\
\vdots \\
-(p_z + tn_z)
\end{bmatrix}
= \begin{bmatrix}
d \\
a \\
b \\
-1
\end{bmatrix}
\begin{bmatrix}
p_x + tn_x \\
p_y + tn_y \\
p_z + tn_z \\
1
\end{bmatrix}
\rightarrow
\begin{bmatrix}
-d \frac{p_x + tn_x}{p_z + tn_z} \\
-d \frac{p_y + tn_y}{p_z + tn_z}
\end{bmatrix}
\]
Vanishing Point

• When \( t \) goes to infinity:

\[
\lim_{t \to \infty} \begin{bmatrix}
-d \frac{p_x + tn_x}{p_z + tn_z} \\
-d \frac{p_y + tn_y}{p_z + tn_z} \\
-d \frac{p_z + tn_z}{p_z + tn_z}
\end{bmatrix} = \begin{bmatrix}
-d \frac{n_x}{n_z} \\
-d \frac{n_y}{n_z} \\
1
\end{bmatrix}
\]

Parallel lines meet at the vanishing point.

• What if there is another ray:

\[
\begin{bmatrix}
q_x + tn_x \\
q_y + tn_y \\
q_z + tn_z \\
1
\end{bmatrix}
\]
Properties of Perspective Projection

• Lines are mapped to lines.
• Parallel lines may not remain parallel. Instead, they may meet at the vanishing point.
• Ratios are not preserved.
• It has foreshortening effects. So it looks real.
• Distances cannot be directly measured, as in parallel projection.
Basic OpenGL Projection

• Everything will be considered in the eye space:
  • Geometry objects have been transformed into the eye coordinate system using the GL_MODELVIEW matrix.
  • You define the projection matrix in GL_PROJECTION, also in the eye space.
  • OpenGL always assume that the viewing direction is the –z direction.

• OpenGL automatically processes each vertex using GL_PROJECTION:
  • After projection, the frustum is converted into a canonical view volume ( [-1, 1] in all coordinates)
OpenGL Orthographic Projection

```latex
\text{glOrtho(left, right, bottom, top, near, far)}
```

X range  Y range  Z range

(right, top, far)

(left, bottom, near)  (-1, -1, -1)  (1, 1, 1)
OpenGL Orthographic Projection

\[
glOrtho(l, r, b, t, n, f)
\]

- Translation so that the center is the origin.
- Scaling so that the size becomes \((2, 2, 2)\).

\[
\begin{bmatrix}
\frac{2}{r-l} & \frac{2}{t-b} & \frac{2}{f-n} \\
\frac{r+l}{r-l} & \frac{t+b}{t-b} & \frac{f+n}{f-n} \\
1 & 1 & 1
\end{bmatrix}
= \begin{bmatrix}
\frac{2}{r-l} & \frac{2}{t-b} & \frac{2}{f-n} \\
\frac{r+l}{r-l} & \frac{t+b}{t-b} & \frac{f+n}{f-n} \\
1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
1 & \frac{-r+l}{2} & 1 \\
1 & \frac{-t+b}{2} & 1 \\
1 & \frac{-f+n}{2} & 1
\end{bmatrix}
\]
OpenGL Perspective Projection

```
glFrustum(left, right, bottom, top, near, far)
```

(left, right, bottom, top) may not be centered along the axis.

Always positive, although it’s facing the –z direction.

Center of Projection

Near Plane (Image Plane)

Far Plane
OpenGL Perspective Projection

```c
gluPerspective(fov, aspect_ratio, near, far)
```

- **ratio** = \( \frac{\text{image\_width}}{\text{image\_height}} \)
- **top** = \( \text{near} \cdot \text{ctan}(\text{fov}/2) \)
- **right** = \( \text{near} \cdot \text{ratio} \cdot \text{ctan}(\text{fov}/2) \)

Always positive, although it’s facing the \( -z \) direction
glFrustum(...) is less useful than gluPerspective(...).

But we can still use it for demonstration purpose next.
OpenGL Perspective Projection

\[
\begin{bmatrix}
  n & n & n & -1 \\
  n & n & n & 1 \\
  n & n & z & 0 \\
  -1 & 1 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix}
= 
\begin{bmatrix}
  -xd / z \\
  -yd / z \\
  -n \\
  1
\end{bmatrix}
\]
OpenGL Perspective Projection

![Diagram of OpenGL Perspective Projection]

- Near plane
- Far plane
- Viewing Frustum

\[
\begin{bmatrix}
  n & n & a & b \\
  1 & 1 & -1 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix}
= 
\begin{bmatrix}
  -xd / z \\
  -yd / z \\
  0 \\
  1
\end{bmatrix}
\]

in \((-n, -f)\), why???
OpenGL Perspective Projection

\[
\begin{bmatrix}
-xd / z \\
-\frac{yd}{z} \\
-a - \frac{b}{z} \\
1
\end{bmatrix}
\]

\[
\begin{align*}
-a - \frac{b}{(-n)} &= -1 & (z = n) \\
-a - \frac{b}{(-f)} &= 1 & (z = f)
\end{align*}
\]

\[
\begin{align*}
a &= -\frac{f + n}{f - n} \\
b &= -\frac{2fn}{f - n}
\end{align*}
\]
OpenGL Perspective Projection

Near plane

Far plane

Viewing Frustum

\[
\begin{bmatrix}
1 & 1 & \frac{-f + n}{f - n} & \frac{-2fn}{f - n} & -1 \\
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]

\( (r/n, t/n, 1) \)

\( (l/n, b/n, -1) \)
OpenGL Perspective Projection

$$\begin{pmatrix}
\frac{2n}{r-l} & \frac{2n}{t-b} & 1 \\
\frac{2n}{r-l} & \frac{2n}{t-b} & 1 \\
1 & 1 & 1
\end{pmatrix} \begin{pmatrix}
1 & \frac{r+l}{2n} & \frac{r+l}{r-l} \\
\frac{t+b}{2n} & \frac{t+b}{t-b} & 1 \\
1 & 1 & 1
\end{pmatrix} = \begin{pmatrix}
\frac{2n}{r-l} & \frac{2n}{t-b} & \frac{r+l}{r-l} \\
\frac{2n}{r-l} & \frac{2n}{t-b} & \frac{t+b}{t-b} \\
1 & 1 & 1
\end{pmatrix}$$
OpenGL Perspective Projection

\[
\begin{bmatrix}
\frac{2n}{r-l} & \frac{r+l}{r-l} & \frac{2n}{t-b} & \frac{r+l}{t-b} & 1 & 1 \\
\frac{2n}{t-b} & \frac{t+b}{t-b} & 1 & 1 & \frac{f+n}{f-n} & \frac{2fn}{f-n} \\
1 & 1 & 1 & 1 & -1 & -1
\end{bmatrix}
\begin{bmatrix}
\frac{2n}{r-l} \\
\frac{r+l}{r-l} \\
\frac{2n}{t-b} \\
\frac{t+b}{t-b} \\
1 \\
1
\end{bmatrix}
= \begin{bmatrix}
\frac{2n}{r-l} & \frac{r+l}{r-l} & \frac{2n}{t-b} & \frac{t+b}{t-b} & 1 & 1 \\
\frac{f+n}{f-n} & \frac{2fn}{f-n} & \frac{f+n}{f-n} & \frac{2fn}{f-n} & -1 & -1
\end{bmatrix}
\]
What happens after projection?

- Clipping
- Viewport transformation
- Rasterization
Depth Test

• Before rasterization, all processes are done based on vertices.

• The z coordinate at each vertex is transformed into a new z value (or called the depth value).

• During rasterization, the z value of each pixel is interpolated from vertices.

• The z value then stored in the depth buffer, for occlusion tests. (smaller z means closer).
Depth Test

- The depth buffer is part of the frame buffer:
  ```c
  glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
  ```

- To enable or disable the depth buffer:
  ```c
  glEnable(GL_DEPTH_TEST);
  ```
  ```c
  glDisable(GL_DEPTH_TEST);
  ```

- Without the depth test, the occlusion is determined by the drawing order.
Common Issues

• When you set up the perspective matrix:
  • Near cannot be zero!
  • far/near cannot be too large! (far cannot be too large, or near cannot be too small.) Why?

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