EECS 487: Interactive Computer Graphics

Lecture 13:
• Planar Geometric Projections
• Orthographic projection
• Perspective projection
• Projections in OpenGL

Planar Geometric Projections

Planar == project onto a plane (vs. planetarium, e.g.)

Geometric == projectors are straight lines (vs. curved lines in cartography, e.g.)

Projection == map from \( n \) to \( n-1 \) dimensions

Euclidean geometry describes shapes “as they are”
• properties of objects that are unchanged by rigid motions: lengths, angles, parallel lines

Projective geometry describes objects “as they appear”
• lengths, angles, parallel lines become “distorted” when we look at objects

Projection System

Common elements:
• Center of Projection (COP)
• Direction of Projection (DOP)
• Projection/view/picture/image plane (PP)
• Projectors/sightlines

Taxonomy of Planar Geometric Projections
**Multiview Orthographic**

- projection plane parallel to one coordinate plane (project onto plane by dropping coordinate perpendicular to plane)
- projection direction perpendicular to projection plane
- good for exact measurements (CAD, architecture)

- preserves ratios, but not angles (→ not visible)
- parallel lines remain parallel
  ⇒ is considered an affine transform

**Axonometric Orthographic**

**Axonometric:**
- projection plane is not parallel to any coordinate plane
- projection direction perpendicular to projection plane

**Isometric:**
- preserves lengths along 3 principal axes
- principal axes make the same angle with each other (120°)

Are we done?

Yes, but we’ve lost depth (z) information, can’t do:
- hidden surface removal
- lighting, etc.

Need to preserve z dimension!
  ⇒ map view volume to canonical view volume

**Parallel Orthographic Projections**

**Orthographic Projection**

View volume defined by *left, right, bottom, top, near, and far planes:*

Map it to cvv:
Orthographic Projection Setup

Simple case: view volume axis-aligned with world coordinate system
• the view volume is in negative $z$, $n > f$

More generally, the view volume is not axis-aligned with world CS (it will always be axis-aligned with eye CS):

Orthographic Projection and Viewing Transform

From an arbitrary viewing volume to canonical view volume
• translate eye to origin
• transform to eye coordinate system
• apply orthographic projection

Orthographic Projection

From an arbitrary axis-aligned bounding box to canonical view volume
• translate and scale:

What would the $T$ and $S$ matrices be?

Perspective Projection

Objects appear smaller as distance from center of projection (eye of observer) increases (perspective foreshortening) ⇒ looks more realistic (human eyes naturally see things in perspective)

Preserves:
• lines (collinearity)
• incidence ("lies on", intersects)
• cross ratio

Does not always preserve parallel lines:
• lines parallel to projection plane remain parallel
• lines not parallel to projection plane converge to a single point on the horizon called the vanishing point (vp)
The Cross Ratio
For the 4 sets of 4 collinear points in the figure, the cross-ratio for corresponding points has the same value (can permute the point ordering)

\[ \left| \frac{p_2 - p_1}{p_4 - p_1} \right| \frac{p_2 - p_1}{p_4 - p_1} \]

Classes of Perspective Projection

**one-point:** projection plane parallel to one coordinate plane (\( \parallel \) to two coordinate axes, one coordinate axis cuts projection plane)

**two-point:** projection plane parallel to one coordinate axis (two coordinate axes cut projection plane)

Classes of Perspective Projection

**three-point:** projection plane not parallel to any coordinate axis (three coordinate axes cut projection plane)

Projective Geometry in 2D

Consider lines and points in \( \mathbb{P} \)
We extend to 3D to simplify dealing with infinity
• origin o out of \( \mathbb{P} \), at a distance = 1 from \( \mathbb{P} \)
To each point \( m \) in \( \mathbb{P} \) we can associate a single ray \( p = (x_1, x_2, x_3) \)
To each line \( l \) in \( \mathbb{P} \) we can associate a single plane \( (A, B, C) \)
• the equation of line \( L \) in projective geometry is \( Ax_1 + Bx_2 + Cx_3 = 0 \)
Homogeneous Coordinates

The ray \( p = (x_1, x_2, x_3) \) and \((\lambda x_1, \lambda x_2, \lambda x_3)\) are the same and are mapped to the same point \( m \) in \( \mathbb{P} \)

- \( p \) is the coordinate vector of \( m \),
  \( (x_1, x_2, x_3) \) its homogeneous coordinates

The planes \((A, B, C)\) and \((\lambda A, \lambda B, \lambda C)\) are the same and are mapped to the same line \( l \) in \( \mathbb{P} \)

- \( L \) is the coordinate vector of \( l \),
  \( (A, B, C) \) its homogeneous coordinates

Point \( p' \) is on line \( L \) if \( L \cdot p' = 0 \)

Perspective Divide

How do we “land” back from the projective world to the 2D Cartesian world of the plane?

- for point, consider the intersection of ray \( p = (\lambda x_1, \lambda x_2, \lambda x_3) \) with the plane \( x_3 = 1 \)
  \[ \Rightarrow \lambda = 1/x_3, \quad m = (x_1/x_3, x_2/x_3, 1) \]

- for line, intersection of plane \( Ax_1 + Bx_2 + Cx_3 = 0 \) with the plane \( x_3 = 1 \)
  is line \( l = Ax_1 + Bx_2 + C = 0 \)

Called “perspective divide”

For the mathematically inclined, or studying computer vision: what’s the geometric interpretation of \( x_3 = 0 \)?

Projective Geometry

Two lines always meet at a single point, and two points always lie on a single line

Projective geometry does not differentiate between parallel and non-parallel lines

Points and lines are dual to each other

To return from homogeneous coordinates to Cartesian coordinates, divide by \( x_3 \)

3D Projective Geometry

These concepts generalize naturally to 3D

Homogeneous coordinates

- projective 3D points have four coordinates: \( p = (x, y, z, w) \)

Projective transformations

- represented by \( 4 \times 4 \) matrices
Vanishing Points
What happens to two parallel lines that are not parallel to the projection plane?
The parametric equation for a line is:

\[ l = p + rt \]

After perspective transform:

\[
\begin{bmatrix}
  p_x \\
  p_y \\
  p_z \\
  1
\end{bmatrix} + t
\begin{bmatrix}
  v_x \\
  v_y \\
  v_z \\
  0
\end{bmatrix}
\]

At the limit, with \( t \to \infty \), we get a point!

\[
\begin{bmatrix}
  p_x + tv_x \\
  p_y + tv_y \\
  p_z + tv_z \\
  1
\end{bmatrix}
\]

Each set of parallel lines intersect at a vanishing point

Perspective Projection
Given the coordinates of the orange point
find the coordinates of the green point

\[ \tan \theta = \frac{y'}{z} \]

\[ y' = \frac{y}{d} \]

Is perspective projection simply:

\[(x, y, z, 1) \to \left( \frac{xd}{z}, \frac{yd}{z}, d, 1 \right) \]

then map to screen by throwing away the \( z \)-coordinate: \((xd/z, yd/z, 1)\)?

Perspective Projection Matrix
Projecting \((x, y, z, 1) \to (xd/z, yd/z, d, 1)\) and throwing away \( d \) does not preserve the depth information

Instead want \( P \) such that:

\[
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix}
\]

\[
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix}
\[
\begin{bmatrix}
  x' \\
  y' \\
  z'
\end{bmatrix}
\]

Just like orthographic projection, we need to map the view volume to a CVV instead of a 2D plane

Projection System Setup
The coordinate system

The eye (e)
- acts as the focal point and COP
- placed at the origin
- looking down (g) along the negative \( z \)-axis (axis of projection)

The screen
- lies in the projection plane
- \( \perp \) to the \( z \)-axis, \( \parallel \) to the \( x-y \) plane
- located at distance \( d \) from the eye
- \( d \) is a.k.a. the focal length

What does it mean for \( w = 1 \)?
What does it mean for \( w > 1 \)?
What is the homogeneous coordinate (HC) when projecting from 3D to 2D?
Perspective Projection View Frustum

View volume (frustum: truncated pyramid):
- defined by (left, right, top, bottom, near, far) clipping planes
- near (n) and far (f) distances along -z-axis, both negative numbers, \( n > f \)

- nothing nearer than \( n \) will be drawn
  → avoid numerical problems during rendering, such as divide by 0
- nothing further than \( f \) will be drawn
  → avoid low depth precision for distant objects

To preserve relative depth information, we must map the frustum to a CVV instead of a 2D plane

Perspective Projection Matrix

Want projection matrix \( P \) such that: 
\[
\begin{bmatrix}
  x \\ y \\ z \\ 1
\end{bmatrix}
\rightarrow 
\begin{bmatrix}
  xd/z \\ yd/z \\ z' \\ 1
\end{bmatrix}
\]

What should \( P \) be?
- we’re projecting from 3D to 2D (not 4D to 3D), use the HC of the projected point to store its depth info (i.e., the “real” HC in 3D to 2D projection)
- first attempt:

\[
Pp = \begin{bmatrix}
  d & 0 & 0 & 0 \\
  0 & d & 0 & 0 \\
  0 & 0 & d & 0 \\
  0 & 0 & 0 & z
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix} = 
\begin{bmatrix}
  xd \\
  yd \\
  zd \\
  z
\end{bmatrix}
= 
\begin{bmatrix}
  xd/z \\
  yd/z \\
  z'
\end{bmatrix}
\]

Any problem?

Perspective Projection Matrix

Second attempt: for a more generic matrix, grab the depth info from the point itself:

\[
Pp = \begin{bmatrix}
  d & 0 & 0 & 0 \\
  0 & d & 0 & 0 \\
  0 & 0 & d & 0 \\
  0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix} = 
\begin{bmatrix}
  xd \\
  yd \\
  zd \\
  z
\end{bmatrix}
= 
\begin{bmatrix}
  xd/z \\
  yd/z \\
  d
\end{bmatrix}
\]

Are we done?
- the projected \( x, y \), and HC are correct already, but after perspective divide, all depths mapped to \( d \!
- 3^{rd} \text{ row of matrix must be tweakd to preserve relative depth info (z')}

From Frustum to CVV
Perspective Projection Matrix

Let \( d = n \)

The 1\(^{st}\) and 2\(^{nd}\) rows of \( P \) are correct already, for the 3\(^{rd}\) row (third attempt):
- the computation of \( z' \) does not rely on \( x \) and \( y \), set the first two numbers of the row to 0
- we can use the remaining two numbers to compute \( z' \), let them be unknowns \( a \) and \( b \) for now:

\[
P_p = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} n \frac{x}{z} \\ 0 \\ a \frac{z+b}{z} \\ 0 \end{bmatrix} = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & a \frac{z+b}{z} & b \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
\]

Then divide by the homogenous coordinate
\( \Rightarrow \) squeezing the frustum into a rectangular box

\[
P_p = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} n \frac{x}{z} \\ 0 \\ (n+f) \frac{z-nf}{z} \\ 0 \end{bmatrix} = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
\]

\( \Rightarrow \) After perspective divide, an object further away appears to be smaller than an equal-size object nearby.
From Frustum to CVV

Now reposition and scale the rectangular box

\[ \begin{bmatrix}
\frac{2}{r-l} & 0 & 0 & \frac{r+l}{r-l} \\
0 & \frac{2}{t-b} & 0 & \frac{t+b}{t-b} \\
0 & 0 & \frac{2}{n-f} & \frac{n+f}{n-f} \\
0 & 0 & 0 & 1 \\
\end{bmatrix} P' = \text{STP} = \text{PP} = \text{PP} \]

Assume viewing transform has been done, so after perspective divide (not shown) we’re only dealing with axis-aligned viewing volume

\[ \begin{bmatrix}
\frac{2n}{r-l} & 0 & \frac{l+r}{l-r} & 0 \\
0 & \frac{2n}{t-b} & \frac{b+t}{b-t} & 0 \\
0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{f-n} \\
0 & 0 & 1 & 0 \\
\end{bmatrix} \]

See also http://www.songho.ca/opengl/gl_projectionmatrix.html

Losing Depth Precision

As a consequence of perspective foreshortening, \( z' \) is not linearly related to \( z \): the same amount of \( \Delta z' \) must represent larger \( \Delta z \)

For example: let \( n = 10, f = 90 \)

\[
\begin{align*}
\text{for } z_1 &= 10, z'_1 &= 10 & \Delta z_1 &= 1 \\
\text{for } z_2 &= 11, z'_2 &= 18.182 & \Delta z_2 &= 8.182 \\
\vdots \\
\text{for } z_k &= 89, z'_k &= 89.888 & \Delta z_k &= 1 \\
\text{for } z_k &= 90, z'_k &= 90 & \Delta z_k &= 0.112
\end{align*}
\]

Implication of the non-linear mapping:

- information on the far plane loses precision
  \( \Rightarrow \) z-buffer punch through or z-fighting
- distances closer to origin are exaggerated

Effect is ameliorated if \( n \) set further from origin
z-Buffer Quantization

z-values stored as non-negative integers

Integers are represented in $b \ (= 16 \text{ or } 32)$ bits,
giving a range of $B \ (= 2^b)$ values \{0, 1, 2, \ldots, $B-1$\}

Floating point $z'$-values are discretized into integer bins:
$\Delta z' = (f-n)/B$, so for example for $n = 10, f = 90$, both
$z_1 = 89, z'_1 = 100-(900/89) = 89.888$ and
$z_2 = 90, z'_2 = 100-(900/90) = 90$
are both discretized to $z' = 90$

Moral of the story: choose $n$ as far away from origin as
possible and $f$ as near as possible (to reduce $\Delta z'$)