Today

- Windowing and Viewing Transformations
  - Windows and viewports
  - Orthographic projection
  - Perspective projection
Screen Space

- Monitor has some number of pixels
  - e.g. 1024 x 768
- Some sub-region used for given program
  - You call it a window
  - Let's call it a viewport instead

- May not really be a “screen”
  - Image file
  - Printer
  - Other
- Little pixel details
- Sometimes odd
  - Upside down
  - Hexagonal

From Shirley textbook.
Screen Space

• Viewport is somewhere on screen
  • You probably don’t care where
  • Window System likely manages this detail
  • Sometimes you care exactly where
• Viewport has a size in pixels
  • Sometimes you care (images, text, etc.)
  • Sometimes you don’t (using high-level library)
Screen Space

Float Pixel Coordinates

\[ u = 0.35 = \frac{i + 0.5}{nx} \]
\[ v = 0.55 = \frac{j + 0.5}{ny} \]

Canonical View Space

- Canonical view region
  - 2D: [-1,-1] to [+1,+1]

\[ x=0.0, \ y=0.0 \]
Canonical View Space

• Canonical view region
  • 2D: [-1,-1] to [+1,+1]

\[
\begin{bmatrix}
  i \\
  j \\
  1
\end{bmatrix} = \begin{bmatrix}
  \frac{n_x}{2} & 0 & \frac{n_x - 1}{2} \\
  0 & \frac{n_y}{2} & \frac{n_y - 1}{2} \\
  0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

From Shirley textbook.
(Image coordinates are up-side-down.)

Remove minus for right-side-up

Canonical View Space

• Canonical view region
  • 2D: [-1,-1] to [+1,+1]

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} = \begin{bmatrix}
  \frac{n_x}{2} & 0 & \frac{n_x - 1}{2} \\
  0 & \frac{n_y}{2} & \frac{n_y - 1}{2} \\
  0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

From Shirley textbook.
(Image coordinates are up-side-down.)

Remove minus for right-side-up
Canonical View Space

- Canonical view region
  - 2D: [-1,-1] to [+1,+1]
- Define arbitrary window and define objects
- Transform window to canonical region
- Do other things (we’ll see clipping latter)
- Transform canonical to screen space
- Draw it.

Note distortion issues...
Projection

• Process of going from 3D to 2D
• Studies throughout history (e.g. painters)
• Different types of projection
  • Linear
    • Orthographic
    • Perspective
  • Nonlinear

Many special cases in books just one of these two...
Projection

• Process of going from 3D to 2D
• Studies throughout history (e.g. painters)
• Different types of projection
  • Linear
    • Orthographic
    • Perspective
  • Nonlinear

Orthographic is special case of perspective...

Many special cases in books just one of these two...
Linear Projection

- Projection onto a planar surface
- Projection directions either
  - Converge to a point
  - Are parallel (converge at infinity)

Linear Projection

- A 2D view

Perspective

Orthographic
Linear Projection

Orthographic  Perspective

Linear Projection

Orthographic  Perspective
Linear Projection

- A 2D view

Note how different things can be seen

Parallel lines “meet” at infinity

Orthographic Projection

- No foreshortening
- Parallel lines stay parallel
- Poor depth cues
Orthographic Projection

- Canonical view region
  - 3D: [-1,-1,-1] to [+1,+1,+1]
  - Assume looking down -Z axis
    - Recall that “Z is in your face”
Orthographic Projection

• Convert arbitrary view volume to canonical

*Assume up is perpendicular to view.
Orthographic Projection

- Step 1: translate center to origin
- Step 2: rotate view to -Z and up to +Y
Orthographic Projection

• Step 1: translate center to origin
• Step 2: rotate view to $-Z$ and up to $+Y$
• Step 3: center view volume

• Step 4: scale to canonical size
Orthographic Projection

- Step 1: translate center to origin
- Step 2: rotate view to -Z and up to +Y
- Step 3: center view volume
- Step 4: scale to canonical size

\[ M = S \cdot T_2 \cdot R \cdot T_1 \]
Perspective Projection

- Foreshortening: further objects appear smaller
- Some parallel line stay parallel, most don’t
- Lines still look like lines
- $Z$ ordering preserved (where we care)
Perspective Projection

Foreshortening: distant objects appear smaller

Vanishing points

- Depend on the scene
- Not intrinsic to camera

“One point perspective”
### Perspective Projection

- **Vanishing points**
  - Depend on the scene
  - Not intrinsic to camera

**“Two point perspective”**

---

### Perspective Projection

- **Vanishing points**
  - Depend on the scene
  - Not intrinsic to camera

**“Three point perspective”**
Perspective Projection

View Frustum

- Distance to image plane $i$
- Near $n$
- Far $f$
- Up
- Bottom $b$
- Top $t$
- Center
- Distance to image plane $i$
- View
- View Frustum
Perspective Projection

• Step 1: Translate center to origin

Perspective Projection

• Step 1: Translate center to origin
• Step 2: Rotate view to -Z, up to +Y
Perspective Projection

* Step 1: Translate \textit{center} to origin
* Step 2: Rotate \textit{view} to \(-Z\), \textit{up} to \(+Y\)
* Step 3: Shear center-line to \(-Z\) axis

Perspective Projection

* Step 1: Translate \textit{center} to origin
* Step 2: Rotate \textit{view} to \(-Z\), \textit{up} to \(+Y\)
* Step 3: Shear center-line to \(-Z\) axis
* Step 4: Perspective
Perspective Projection

- Step 1: Translate center to origin
- Step 2: Rotate view to -Z, up to +Y
- Step 3: Shear center-line to -Z axis
- Step 4: Perspective

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & \frac{i+f}{i} & f \\
0 & 0 & -1 & 0
\end{bmatrix}
\]

- Points at \( z = -i \) stay at \( z = -i \)
- Points at \( z = -f \) stay at \( z = -f \)
- Points at \( z = 0 \) goto \( z = \pm \infty \)
- Points at \( z = -\infty \) goto \( z = -(i+f) \)
- \( x \) and \( y \) values divided by \(-z/i\)

- Straight lines stay straight
- Depth ordering preserved in \([-i,-f]\)
- Movement along lines distorted
Perspective Projection

- Step 4: Perspective
  - Points at \( z=-i \) stay at \( z=-i \)
  - Points at \( z=f \) stay at \( z=f \)
  - Points at \( z=0 \) goto \( z=\pm\infty \)
  - Points at \( z=-\infty \) goto \( z=-(i+f) \)
  - \( x \) and \( y \) values divided by \(-z/i\)
  - Straight lines stay straight
  - Depth ordering preserved in [-\(i\), \(f\)]
  - Movement along lines distorted

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & i+f/i & f \\
0 & 0 & -1/i & 0 \\
\end{bmatrix}
\]

From Shirley textbook.
Perspective Projection

From Shirley textbook.

```
[1 0 0 0]  
[0 1 0 0]  
[0 0 i+f 0]  
[0 0 -1 0]  
```

"Eye" plane

Some horizontal line

View vector
Perspective Projection

Visualizing division of $x$ and $y$ but not $z$

Motion in $x, y$
Perspective Projection

Note that points on near plane fixed

Recall that points on far plane will stay there...
Perspective Projection

When we also divide $z$ points must remain on straight lines.

Perspective Projection

Lines extend outside view volume.
Perspective Projection

Motion in $z$
Perspective Projection

Motion in $z$

Total motion
Perspective Projection

- Step 1: Translate \textit{center} to orange
- Step 2: Rotate \textit{view} to $-Z$, \textit{up} to $+Y$
- Step 3: Shear center-line to $-Z$ axis
- Step 4: Perspective
- Step 5: center view volume
- Step 6: scale to canonical size

\[
M = M_o \cdot M_p \cdot M_v
\]
Perspective Projection

• There are other ways to set up the projection matrix
  • View plane at $z=0$ zero
  • Looking down another axis
  • etc...
• Functionally equivalent

Vanishing Points

• Consider a ray:

$$\mathbf{r}(t) = \mathbf{p} + t \mathbf{d}$$
Vanishing Points

• Ignore $Z$ part of matrix
• $X$ and $Y$ will give location in image plane
• Assume image plane at $z=-i$

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
\text{whatever} & & & \\
0 & 0 & -1 & 0
\end{bmatrix}
\begin{bmatrix}
I_x \\
I_y \\
I_z \\
I_w
\end{bmatrix}
=
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & -1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\]
### Vanishing Points

- Assume \( d_z = -1 \)

\[
\begin{bmatrix}
I_x / I_w \\
I_y / I_w
\end{bmatrix} = \begin{bmatrix}
-x / z \\
-y / z
\end{bmatrix} = \begin{bmatrix}
p_x + td_x \\
-p_z + t \\
p_y + td_y \\
-p_z + t
\end{bmatrix}
\]

\[
\lim_{t \to \pm \infty} = \begin{bmatrix}
d_x \\
d_y
\end{bmatrix}
\]

- All lines in direction \( d \) converge to same point in the image plane -- the vanishing point
- Every point in plane is a v.p. for some set of lines
- Lines parallel to image plane ( \( d_z = 0 \) ) vanish at infinity

What’s a horizon?
Perspective Tricks

Right Looks Wrong (Sometimes)

* From Collection of Geometric Perceptual Distortions in Pictures, Zorin and Barr SIGGRAPH 1995
Right Looks Wrong (Sometimes)

From WIRED Magazine

Strangeness

The Ambassadors
by Hans Holbein the Younger
Strangeness

The Ambassadors
by Hans Holbein the Younger

Ray Picking

• Pick object by picking point on screen

• Compute ray from pixel coordinates.
Ray Picking

• Transform from World to Screen is:

\[
\begin{bmatrix}
I_x \\
I_y \\
I_z \\
I_w
\end{bmatrix} = \mathbf{M}
\begin{bmatrix}
W_x \\
W_y \\
W_z \\
W_w
\end{bmatrix}
\]

• Inverse:

\[
\begin{bmatrix}
W_x \\
W_y \\
W_z \\
W_w
\end{bmatrix} = \mathbf{M}^{-1}
\begin{bmatrix}
I_x \\
I_y \\
I_z \\
I_w
\end{bmatrix}
\]

• What Z value?

\[
\begin{bmatrix}
W_x \\
W_y \\
W_z \\
W_w
\end{bmatrix} = \mathbf{M}
\begin{bmatrix}
I_x \\
I_y \\
I_z \\
I_w
\end{bmatrix}
\]

Ray Picking

• Recall that:

• Points at z=−i stay at z=−i
• Points at z=−f stay at z=−f

\[
r(t) = \mathbf{p} + t \mathbf{d}
\]

\[
r(t) = \mathbf{a}_w + t(\mathbf{b}_w - \mathbf{a}_w)
\]

\[
\mathbf{a}_s = [s_x, s_y, -i]
\]

\[
\mathbf{b}_s = [s_x, s_y, -f]
\]

Depends on screen details, YMMV

General idea should translate...
Depth Distortion

- Recall depth distortion from perspective
  - Interpolating in screen space different than in world
  - Ok for shading (mostly)
  - Bad for texture

![Diagram showing world, screen, and depth distortion](image)

Mathematically:

\[
S_1 = \frac{P_1}{h_1} \\
S_2 = \frac{P_2}{h_2} \\
S_3 = \frac{P_3}{h_3} \\
S_4 = \frac{P_4}{h_4}
\]
We know the $S_i$, $P_i$, and $b_i$, but not $a_i$. 

$$X = \sum_i S_ib_i$$

$$Q = \sum_i P_ia_i$$

$$X = Q/h = \left( \sum_i P_ia_i \right) / \left( \sum_j h_j a_j \right)$$
Depth Distortion

\[ S_1 = P_1/h_1 \]
\[ S_4 = P_4/h_4 \]
\[ S_3 = P_3/h_3 \]
\[ S_2 = P_2/h_2 \]

\[ X = \sum_i S_i b_i \]

\[ \sum_i S_i b_i = \left( \sum_i P_i a_i \right) / \left( \sum_j h_j a_j \right) \]
Depth Distortion

\[ b_i/h_i = a_i / \left( \sum_j h_j a_j \right) \quad \forall i \]

Independent of given vertex locations.

\[ \sum_i P_i h_i/h_i = \left( \sum_i P_i a_i \right) / \left( \sum_j h_j a_j \right) \]

Depth Distortion

Linear equations in the \( a_i \):

\[ \left( \sum_j h_j a_j \right) b_i/h_i - a_i = 0 \quad \forall i \]
Depth Distortion

For a line: \( a_1 = \frac{h_2b_1}{(h_1b_2 + h_1b_2)} \)

For a triangle: \( a_1 = \frac{h_2h_3b_1}{(h_2b_3b_1 + h_1b_3b_2 + h_1h_2b_3)} \)

Obvious Permutations for other coefficients.