EECS 487: Interactive Computer Graphics

Lecture 14:

- Projections in OpenGL

Projection Transforms

Shape view volume

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
```

Orthographic projection

```
glOrtho(l,r,b,t,n,f)
gluOrtho2D(l,r,b,t):
```

calls `glOrtho()` with $n = -1, f = 1$

Perspective projection

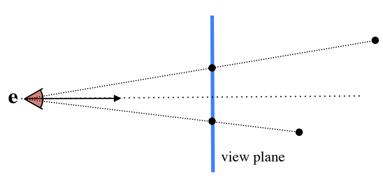
```
glFrustum(l,r,b,t,n,f)
gluPerspective(fovy, aspect, n, f)
```

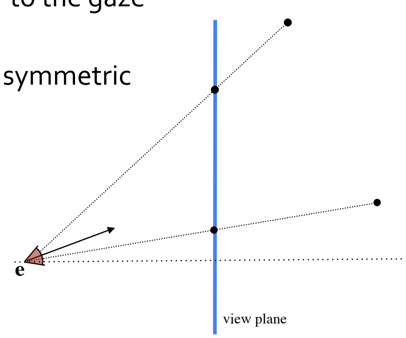
`gluLookAt()` must come "before in code, after in action" to other modeling transforms, but not so `gluPerspective()`, why?

OpenGL Perspective Projection

`glFrustum(l,r,b,t,|n|,|f|)`

- center of projection is at eye location
- gazing down the $-z$ axis (by convention)
- projection plane is perpendicular to the gaze direction (by convention)
- (frustum also doesn't have to be symmetric about the gaze direction)

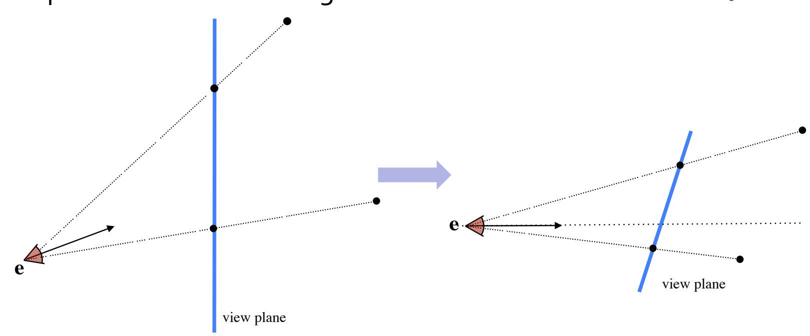




Shifted Perspective Projection

Off-axis projection:

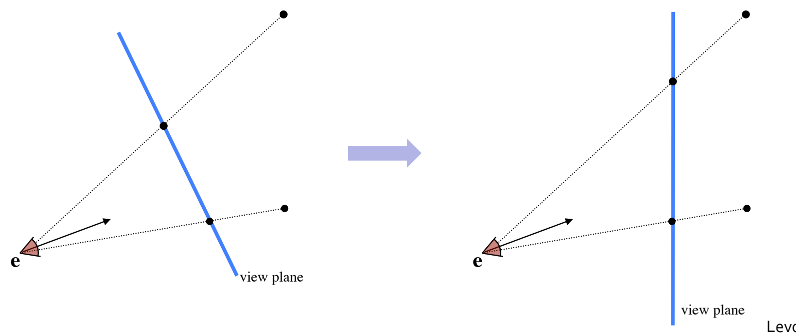
- frustum asymmetrical about the gaze direction
- projection plane not perpendicular to gaze direction
- equivalent to tilted PP if gaze direction is transformed to $-z$ axis



Uses of Shifted Perspective

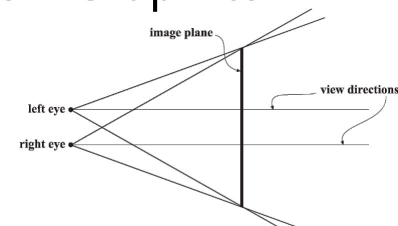
Architect's camera:

- tall buildings viewed from a low angle result in 3-point perspective
- to remove the 3rd vanishing point: set PP parallel to façade of buildings such that top of building is the same distance to PP as bottom of building



Levoy08

Stereo and Multi-view Graphics

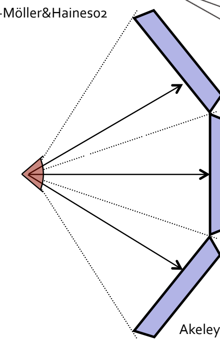


Akenine-Möller&Haines02

Stereo ("3D") rendering:

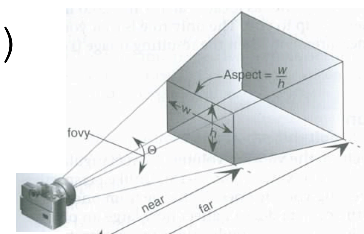
- generate two images, one in red, one in green, with off-axis perspective
- composite them and view with special glasses

Multi-view displays: there is one view direction, but three planes of projection, with the side ones off-axis



Akeley07

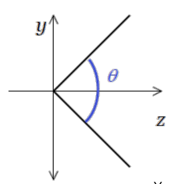
`gluPerspective()`



Redbook10

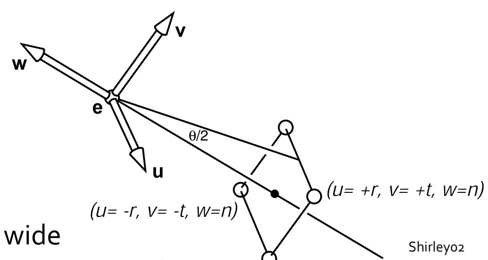
Simpler frustum setup

- symmetric: viewport is always centered about z-axis
- `gluPerspective(fovy, aspect, |near|, |far|)`
- `fovy`: field of view along the y (vertical) axis, in degrees
  - the angle (θ) between the top and bottom planes
- `aspect`: aspect ratio (width/height)) of PP
- will need to change every time window is resized



Yu08

Setting `fovy`



Shirley02

For the human eyes:

- $\theta = 2 \arctan(t/|n|)$
- a 21" monitor is about 16" wide
- recommended viewing distance (n) is 25"

⇒ set the viewing angle (θ) to 35°

In cameras:

- focal length determines n
- for 35mm (36mm×24mm) image/film size:
  - 18mm, super-wide angle lens, has fovy of 67°
  - 28mm, wide-angle lens: 46°
  - 50mm, "normal" lens: 27°
  - 100mm, telephoto lens: 14°

James07

Field of View and Perspective

FoV determines "strength" of perspective foreshortening

Ansel Adams

wide angle
prominent foreshortening
close viewpoint (small $|n|$)

telephoto: narrow angle
little foreshortening
far viewpoint (large $|n|$)

James07

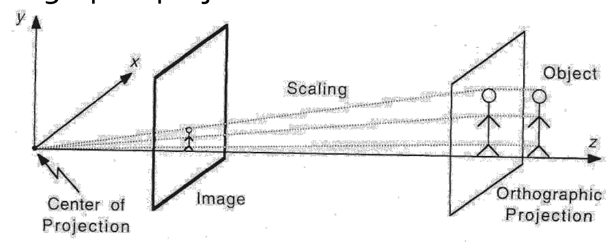
Focal Length and Perspective

Consider the perspective transform:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/n & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \right) = \begin{bmatrix} x \\ y \\ z/n \end{bmatrix}$$

Suppose $n \to \infty$ and $z \to \infty$, then $z/n \to 1$

In the limit, perspective projection gives us an orthographic projection:



Curless08

Projection Transform

Example:

```
// Projection transformation
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(60.0, 1.0, 1.0, 100.0);
            // fovy, aspect, |near|, |far|

// ModelView transformations
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
// viewing first
gluLookAt(10.0,10.0,10.0,1.0,2.0,3.0,0.0,0.0,1.0) ;
// modeling last
glTranslatef( 1.0, 1.0, 1.0 );
glRotatef( 90.0, 1.0, 0.0, 0.0 );
glutSolidTeapot();
```

Orthographic Projection in OpenGL

In deriving the projection matrices, we assumed positive dimensions of the view volume:
$r > l, t > b, n > f$

$$\mathbf{P}_o^{n>f} = \mathbf{ST} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{l+r}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{b+t}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

OpenGL uses absolute distance values which is equivalent to the view volume being on the $+z$-axis, $f > n$, the orthographic projection matrix becomes:

$$\mathbf{P}_o^{f>n} = \mathbf{ST} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{l+r}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{b+t}{t-b} \\ 0 & 0 & \frac{2}{|f|-|n|} & -\frac{|f|+|n|}{|f|-|n|} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Projection in OpenGL

$$\mathbf{P}_o^{f>n} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{l+r}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{b+t}{t-b} \\ 0 & 0 & \frac{2}{|f|-|n|} & -\frac{|f|+|n|}{|f|-|n|} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To draw into the screen, we mirror the view volume into $-z$-axis before applying projection:

$$\mathbf{P}_O^{OpenGL} = \mathbf{P}_O^{f>n} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{|f|-|n|} & -\frac{|f|+|n|}{|f|-|n|} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Perspective in OpenGL

Similarly for the perspective matrix:

$$\mathbf{P}_p^{openGL} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{|f|-|n|} & -\frac{|f|+|n|}{|f|-|n|} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} |n| & 0 & 0 & 0 \\ 0 & |n| & 0 & 0 \\ 0 & 0 & |f|+|n| & |f||n| \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

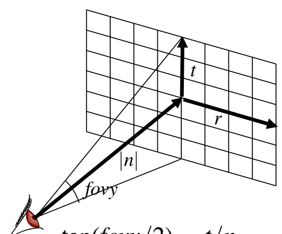
to negate the sign in the homogeneous coordinate obtained from the mirroring

$$= \begin{bmatrix} \frac{2|n|}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2|n|}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{|f|+|n|}{|f|-|n|} & -\frac{2|f||n|}{|f|-|n|} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

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

Perspective in OpenGL

$$\mathbf{P}_P^{glFrustum} = \begin{bmatrix} \frac{2|n|}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2|n|}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{|f|+|n|}{|f|-|n|} & -\frac{2|f||n|}{|f|-|n|} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$



$\tan(fovy/2) = t/n$
$\Rightarrow n = t/\tan(fovy/2)$
$r/t = aspect$
$\Rightarrow r = t{*}aspect$
these give us:

$$\mathbf{P}_p^{gluPerspective} = \begin{bmatrix} \frac{1}{aspect \cdot \tan(fovy/2)} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(fovy/2)} & 0 & 0 \\ 0 & 0 & \frac{|n|+|f|}{|n|-|f|} & \frac{2|f||n|}{|n|-|f|} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Why Separate ModelView and Projection Matrices?

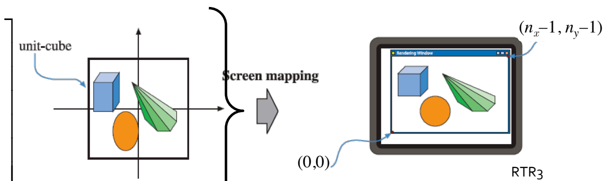
Why not collapse the $\mathbf{M}$, $\mathbf{V}$, and $\mathbf{P}$ matrices into a single $\mathbf{M}_{MVP}$ matrix?

- projection usually specified once, modelview changes when camera moves
- only modelview is used in transforming normals
- lighting: separate $\mathbf{P}$ means no need to explicitly specify viewer location:
  - in eye coordinates, view location is $[0, 0\ , 0]^T$ and view direction is $[0\ 0\ \text{-}1]^T$
  - particularly important for specular lighting: highlights depend on viewer location

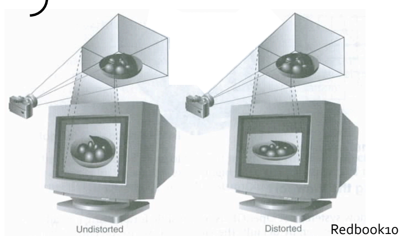
Screen Mapping/Viewport Transform

`glViewport(0,0,nx,ny)`: viewport transform of cvv to screen min coordinate $(0,0)$ and max coordinate $(n_x-1\,,\,n_y-1)$:

$$\underbrace{\begin{bmatrix} \frac{n_x}{2} & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & \frac{n_y-1}{2} \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{D}}$$



Resulting image distorted if viewport aspect ratio $\neq$ that of projection transform:



Direct3D

Location $(0,0)$ at the top left corner, not bottom left (scan order of CRT as opposed to Cartesian)

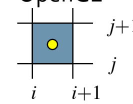
Uses row-major form in documentation, so $\mathbf{v}^T\mathbf{M}^T$ instead of $\mathbf{Mv}$, and in memory storage

Left-handed: z-positive into the screen

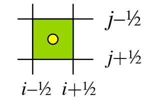
Near plane at $z=0$ instead of $z=-1$, z-depths in $[0,1]$ instead of $[-1,1]$ (however x and y still have $[-1,1]$ ranges)

$$(\mathbf{P}_p^{Direct3D})^T = \begin{bmatrix} \frac{2|n|}{r-l} & 0 & -\frac{r+l}{r-l} & 0 \\ 0 & \frac{2|n|}{t-b} & -\frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{|f|}{|f|-|n|} & -\frac{|f||n|}{|f|-|n|} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

OpenGL



Direct3D 9



Direct3D 10/11