**Projection Transforms**

Shape view volume
```c
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
```

Orthographic projection
```c
glOrtho(l,r,b,t,n,f);
gluOrtho2D(l,r,b,t);
```
calls `glOrtho()` with `n = -1, f = 1`

Perspective projection
```c
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

Stereo and Multi-view Graphics

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

gluPerspective()

Simpler frustum setup
- symmetric: viewport is always centered about z-axis
- \( \text{gluPerspective}(\text{fovy}, \text{aspect}, |\text{near}|, |\text{far}|) \)
  - \( \text{fovy} \): field of view along the y (vertical) axis, in degrees
  - \( \text{aspect} \): aspect ratio (width/height) of PP
  - will need to change every time window is resized

Setting \( \text{fovy} \)

For the human eyes:
- \( \theta = 2 \arctan(t/|n|) \)
- a 21” monitor is about 16” wide
- recommended viewing distance (\( n \)) is 25”
  \( \Rightarrow \) set the viewing angle (\( \theta \)) to 35°

In cameras:
- focal length determines \( n \)
  - for 35mm (36mm \( \times \) 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°
Field of View and Perspective

FoV determines “strength” of perspective foreshortening

- **wide angle**: prominent foreshortening
  - close viewpoint (small |n|)
- **telephoto**: narrow angle
  - little foreshortening
  - far viewpoint (large |n|)

Focal Length and Perspective

Consider the perspective transform:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1/n & 0
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
z/n
\end{bmatrix} = \begin{bmatrix}
x \\
y \\
z \\
z/n
\end{bmatrix}
\]

Suppose \( n \to \infty \) and \( z \to \infty \), then \( z/n \to 1 \)

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

Orthographic Projection in OpenGL

In deriving the projection matrices, we assumed positive dimensions of the view volume:

\( r > l, t > b, n > f \)

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

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

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

Perspective in OpenGL

Similarly for the perspective matrix:

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

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

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

Why Separate ModelView and Projection Matrices?

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

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

\[
\begin{align*}
p_{\text{MVP}} &= \begin{bmatrix}
\frac{2t}{r-l} & 0 & \frac{t+r}{r-l} & 0 \\
0 & \frac{2t}{t-b} & \frac{t+b}{t-b} & 0 \\
0 & 0 & \frac{1}{\text{aspect} \cdot \text{tan(fovy/2)}} & 0 \\
0 & \frac{1}{\text{tan(fovy/2)}} & 0 & 0
\end{bmatrix} \\
p_{\text{Perspective}} &= \begin{bmatrix}
\frac{1}{\text{aspect} \cdot \text{tan(fovy/2)}} & 0 & 0 & 0 \\
0 & \frac{1}{\text{tan(fovy/2)}} & 0 & 0 \\
0 & 0 & \frac{2t}{\|n\|} & \frac{-t\|n\|}{\|n\|} \\
0 & 0 & \frac{1}{\|n\|} & -1 \\
\end{bmatrix}
\end{align*}
\]
Screen Mapping/Viewport Transform

\( \text{glViewport}(0, 0, nx, ny) \): viewport transform of cvv to
screen min coordinate \((0, 0)\) and max coordinate \((nx-1, ny-1)\):

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

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 \(v^T M^T\) instead of \(M v\), 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 \([-1, 1]\)
(although \(x\) and \(y\) still have \([-1, 1]\) ranges)

\[
\begin{bmatrix}
\frac{2 n_x}{r-l} & 0 & \frac{-r+l}{r-l} & 0 \\
0 & \frac{2 n_y}{t-b} & \frac{-t+b}{t-b} & 0 \\
0 & 0 & \frac{1}{1-f_1} & \frac{-f_1}{1-f_1} \\
0 & 0 & 1 & 0
\end{bmatrix}
\]