Math of Projections: Overview
- Math of perspective projection, standard configuration
- OpenGL perspective projections
- Math of orthographic projection
- OpenGL orthographic projections
- Viewport transformations and setting them in OpenGL

Summary
- Viewing transformations
- Orthographic projection canonical viewing volume
- Perspective projection canonical viewing volume
- Hidden surface removal

Math of perspective projection
- The discussion here is carried out with respect to the camera (view) reference frame, \( (VRP,u,v,n) \)
- The projection transformation maps 3D points to 2D points in the projection plane
- Standard configuration:
  - COP=VRP
  - Projection plane is orthogonal to z-axis, at \( z=d \)

Math of perspective projection
- Standard configuration:
  - Let \( O \) be COP
  - Projection plane has equation \( z=d, d<0 \)
  - 3D point \( P \) has homogeneous coordinates \( (x,y,z,1) \)
  - It is mapped to point \( Q(xp,yp,d) \) in the projection plane
    - \( Q \) is on the segment \( PO \), thus
      - \( Q=cP+(1-c)O \), where \( 0<c<1 \)
      - Thus, \( xp = c.x, yp = c.y, d = c.z \Rightarrow c = d/z \)
  - Note that projection is not linear in affine coordinates.

Perspective Projections
- The graphics system applies a 4 x 4 projection matrix after the model-view matrix

Perspective Viewing
Math of perspective projection

- \( P = p(x,y,z,1) \rightarrow Q = q(d,x,y,z,d,1) \), in homogeneous coordinates.
- Perspective projection is not linear in affine coord.
- Perspective projection is linear in homogeneous

\[
M_{\text{per}} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1/d & 0
\end{bmatrix}
\]
\[
q = M_{\text{per}}p = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1/d & 0
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
d/1
\end{bmatrix}
\]

- From homogeneous to affine coordinates:
  \((a,b,c,w) \rightarrow (a/w, b/w, c/w)\)
- Thus \((x,y,z,d/2)\) in homogeneous becomes
  \((x/z, y/z, 1)\) in proj. plane.

Perspective Viewing in OpenGL

- Depth of projection plane (size of clipping window) inside the pyramid does not matter.
- All that matters is object size relative to the window.
- The projection plane depth does not affect relative size.
- Thus in CG systems usually far or near plane is selected to be the projection plane. Near in OpenGL.

OpenGL Perspective

- \( \text{glFrustum}(\text{left}, \text{right}, \text{bottom}, \text{top}, \text{near}, \text{far}); \)
- \( \text{glMatrixMode} \rightarrow \text{GL}_\text{PROJECTION}; \)
- \( \text{glLoadIdentity}(); \)
- \( \text{glFrustum}(\text{left}, \text{right}, \text{bottom}, \text{top}, \text{near}, \text{far}); \)
- The relationship between front(near) plane and COP(origin) define the steepness of the frustum.

Clipping

- Object parts outside of the view volume are clipped.
- First we will discuss the OpenGL functions that set up the projection transformations.
- Next we will discuss the viewport transformation and setting the viewport in OpenGL.
- Last, we will go back to projection, and see how the graphics systems carry out efficiently the more general perspective projection by reducing it to the standard perspective, and then to the canonical orthographic.
- You can set up any projection you want (parallel or perspective) by setting up the the projection matrix directly.
- Although, more often we use affine transformations to reduce more general projections to the canonical ones.

Perspective Viewing in OpenGL

- \( \text{gluPerspective}(\text{fovy}, \text{aspect}, \text{near}, \text{far}); \)

Projections in OpenGL

- Objects not in the view volume are clipped.
- View (projection) plane is front clipping plane.
OpenGL Orthographic Projection

```cpp
void glOrtho(GLfloat left, GLfloat right, GLfloat bottom, GLfloat top, GLfloat near, GLfloat far);
```

Standard Orthographic Projection

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

Viewport Transformation

- Viewport transformation corresponds to the stage where the size of the developed photograph is chosen
- The viewport is measured in pixels, in screen window coordinates, which reflect the position of pixels on the screen relative to the lower left corner of the window
- Vertices outside the viewing volume have been clipped

Defining the Viewport

- The window manager, not OpenGL, is responsible for opening a window on the screen
- By default the viewport is set to the entire pixel rectangle of the window that's opened
- Use the `glViewport()` command to choose a smaller drawing region

```c
void glViewport(GLint x, GLint y, GLsizei width, GLsizei height);
```

Defining The Viewport

- By default, the initial viewport parameters are \((0, 0, \text{winWidth}, \text{winHeight})\), where \(\text{winWidth}\) and \(\text{winHeight}\) are the window dimensions.
- The aspect ratio of a viewport should generally equal the aspect ratio of the viewing volume.
- If the two ratios are different, the projected image will be distorted as it's mapped to the viewport.
Viewport Distortion

Summary

- World to View coordinates
  - Camera position: VRP
  - Viewing transformation
    - Translate VRP to origin: T(-VRP)
    - Rotate, aligning view frame (u,v,n) with world frame
    - The composite transformation R.T will have the effect of transforming the coordinates of vertices from world to view coordinates

Summary

- Canonical view orthographic projection
  - VRP at the origin
  - Looking in negative z direction (COP does not matter)
  - View-up vector is (0,1,0)
  - View volume is a cube of side 2 center at origin,
    - (left, right, bottom, top, near, far)=(-1,1,-1,1,1,-1)

Summary

- Perspective projection standard viewing
  - VRP at origin
  - Looking in negative z-direction, look at is (0,0,-1)
  - COP coincides with VRP, viewing direction is orthogonal to view plane
  - View-up vector is (0,1,0)
  - View volume is a regular frustum
    - (left, right, top, bottom, near, far)=(-1,1,-1,1,1,z_far).
    - The near plane is at -1, the far is at -z.
    - Side clipping planes make 45 deg angles with z axis

Hidden Surface Removal

- z-buffer algorithm – as the polygons are rasterized we keep track of the distance from the VRP to the closest point on each projector. We display only the closest point
- Requires a depth or z buffer to store the necessary depth information

```c
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
glEnable(GL_DEPTH_TEST);
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
```
Hidden Surface Removal

- Display only visible surfaces (remove hidden)
- Painter's algorithm: sort polygons according to z, project in order of decreasing depth
- Pixel based: back track rays from pixels to first intersection
- Z-buffer algorithm: based on relative depth after projection

Perspective Normalization

- How can we apply general perspective projection?
- Derive the perspective projection matrix (general 4x4 matrix), load that as a PROJECTION matrix
- Trick: Using series of affine transformations to alter the world, so that the image of the distorted world under standard (canonical) projection is the same as the image of the undistorted world under the original general projection
- Graphics systems go even farther: perspective projections are implemented internally as orthographic projections (of the distorted world) with respect to the canonical view volume (2x2x2 cube). This way the clipping is very efficient, and z-buffer algorithm is supported by preserving relative depth in the process of transforming objects.

Perspective Normalization Idea

- The distortion is described by a homogeneous-coordinate matrix
- Concatenate this matrix with an orthogonal-projection matrix to yield a resulting projection matrix

General Perspective Projection

- Projection converts points in 3-D space to points on the projection plane
- Three steps (the graphics system implements them):
  - Converts the viewing volume (general frustum) to the canonical perspective view volume
  - Next converts the canonical perspective view volume to the canonical orthographic view volume
  - Applies orthographic projection matrix

The “most” canonical view volume

- Orthographic projection with respect to most canonical view volume
- Canonical view volume is a 2x2x2 cube whose center is at the origin (default view volume)
- How to clip?: simple
- How to project?: no division
General Orthogonal Projection

```c
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
```

Determine which objects are clipped out.

**Projection Matrix** maps a *view volume* to the *canonical view volume*.

Projection Matrix

```
| 0 0 0 2 |
| 0 2 0 0 |
| 0 0 2 0 |
| 0 0 0 1 |
```

Oblique Projection

```
```

Oblique Projection

```
```

Perspective Projection Matrices

- Look for a transformation allows a simple canonical projection by distorting the vertices of an object.
- Three steps: 1) select canonical viewing volume, 2) introduce the perspective normalization transformation and 3) derive the perspective projection matrix.
Perspective Projection Matrices

\[ \mathbf{N} \] is called the perspective normalization matrix. It converts a perspective projection to an orthogonal projection.

Projection and Shadows

- Shadows are not geometric objects but important for realism.
- A point is in a shadow if it is not illuminated by any light source.
- A shadow polygon is a flat shadow that results from projecting the original polygon onto a surface.