OpenGL 3D Coordinate System

Defining 3D Polygons in OpenGL
- e.g., front face of a cube
  glBegin(GL_POLYGON)
  glVertex3f(-0.5f, 0.5f, 0.5f);
  glVertex3f(-0.5f, -0.5f, 0.5f);
  glVertex3f(0.5f, -0.5f, 0.5f);
  glVertex3f(0.5f, 0.5f, 0.5f);
  glEnd();
- need to define the other faces

Projection Transformation
- First tell OpenGL you’re using the projection matrix
  glMatrixMode(GL_PROJECTION);
- Then Initialize it to the identity matrix
  glLoadIdentity();
- Then define the viewing volume, for example:
  glFrustum(-1.0, 1.0, -1.0, 1.0, 2.0, 7.0);  // for perspective
  glOrtho(-1.0, 1.0, -1.0, 1.0, 2.0, 7.0);    // for parallel
- Eye is at (0,0,0)

The Viewing Volume
- Everything outside viewing volume is clipped
- Think of near plane as being window’s client area
**Modelview Transformation**

Our cube is not visible
It lies in front of near clipping plane

---

**Positioning the Camera**

- Can set the camera point
- And the “lookat” point
- And the up direction

\[
\text{gluLookAt(xc,yc,zc,xa,ya,za,xu,yu,zu);} \\
(xc,yc,zc) \text{ coordinates of virtual camera} \\
(xa,ya,za) \text{ coordinates of lookat point} \\
(xu,yu,zu) \text{ up direction vector}
\]

- Example:
  \[
  \text{gluLookAt(2.0,2.0,2.0,0.0,0.0,0.0,0.0,0.0,0.1);} \\
  \text{camera at (2,2,2), looking at origin, z-axis is up}
  \]

---

**Modelview Transformation**

- Used to perform geometric translations, rotations, scalings
- Also implements the viewing transformation
- If we don’t position the camera, we need to move our cube into the viewing volume

\[
\text{glMatrixMode(GL\_MODELVIEW);} \\
\text{glLoadIdentity();} \\
\text{glTranslate(0.0f, 0.0f, -3.5f);} \\
\text{Translates cube down z-axis by 3.5 units}
\]

---

**OpenGL Composite Transformations**

- Combine transformation matrices
- Example: Rotate by 45 degrees about a line parallel to the z axis that goes through the point \((x_f, y_f, z_f)\) – the fixed point

\[
\text{glMatrixMode(GL\_MODELVIEW);} \\
\text{glLoadIdentity();} \\
\text{glTranslate(x_f,y_f,z_f);} \\
\text{glRotate(45, 0.0, 0.0, 1.0);} \\
\text{glTranslate(-x_f,-y_f,-z_f);} \\
\]

- Note last transformation specified is first applied
  - Because each transformations in OpenGL is applied to present matrix by postmultiplication

---

**Scaling and Rotating a Model**

\[
\text{glScalef(2.0f, 2.0f, 2.0f);} \quad //\text{twice as big} \\
\text{parameters: sx, sy, sz}
\]

\[
\text{glTranslatef(2.0f, 3.5f, 1.8f);} \quad //\text{move object} \\
\text{parameters: tx, ty, tz}
\]

\[
\text{glRotatef(30.0f, 0.0f, 0.0f, 1.0f);} \quad //\text{30 degrees about z-axis} \\
\text{parameters:} \\
\quad – \text{angle, (x, y, z)} \\
\quad – \text{coordinates of vector about which to rotate}
\]
Typical code for a polygon mesh model

```c
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(-1.0, 1.0, -1.0, 1.0, 2.0, 7.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0.0f, 0.0f, -3.5f);               // translate into viewing frustum
glRotatef(30.0f, 0.0f, 0.0f, 1.0f);           // rotate about z axis by 30
glClearColor(1.0f, 1.0f, 1.0f, 1.0f);     // set background color
glClear(GL_COLOR_BUFFER_BIT);  // clear window
glColor3f(0.0f, 0.0f, 0.0f); // drawing color
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
```

See 3dxform example program

**The OpenGL Utility Library (GLU) and Quadric Surfaces**

- Provides many modeling features
  - Quadric surfaces
    - described by quadratic equations
    - spheres, cylinders, disks
  - Polygon Tessellation
  - Approximating curved surfaces with polygon facets
- Non-Uniform Rational B-Spline Curves & Surfaces (NURBS)
- Routines to facilitate setting up matrices for specific viewing orientations & projections

**Modeling & Rendering a Quadric with the GLU**

1. Get a pointer to a quadric object
2. Make a new quadric object
3. Set the rendering style
4. Draw the object
5. When finished, delete the object

**OpenGL GLU Code to Render a Sphere**

```c
GLUquadricObj *mySphere;
mySphere=gluNewQuadric();
gluQuadricDrawStyle(mySphere,GLU_FILL);
// some other styles: GLU_POINT, GLU_LINE
```

**The GLUT and Quadric Surfaces**

- Many predefined quadric surface objects
  - `glutWire***()`
  - `glutSolid***()`
  - Some examples:
    - `glutWireCube(size); glutSolidCube(size);`
    - `glutWireSphere(radius,nlongitudes,nlatitudes);`
    - `glutWireCone(rbase,height,nlongitudes,nlatitudes);`
    - `glutWireTeapot(size);`
    - Lots of others
  - See cone_perspective example program