
CSE 472

Virtual Camera

CSE 472 Spring 2009

1

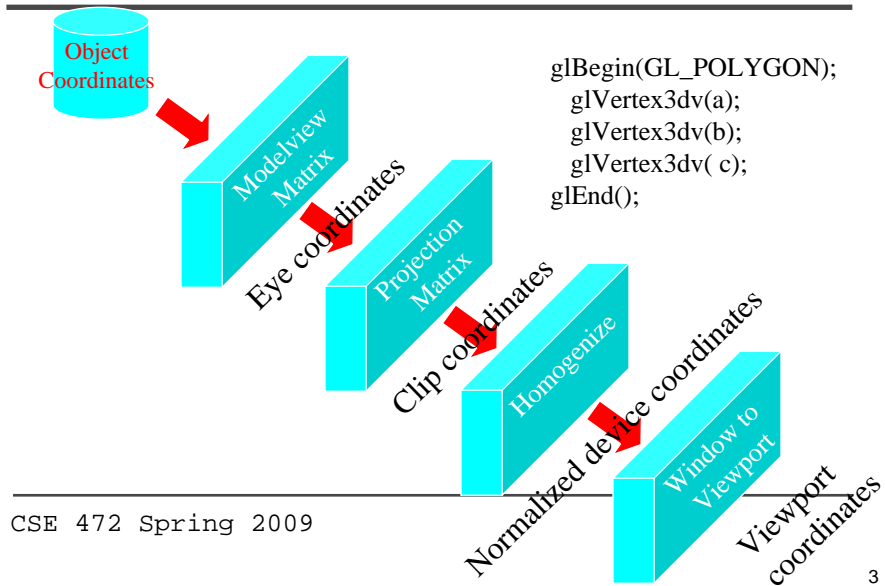
Moving to 3D

Camera Configuration
Parameters
Tessellation
Scene Graphs

CSE 472 Spring 2009

2

Within OpenGL



Program Structure

OnGLDraw()

- Clear buffers
- Set up camera
- Position camera
- Configure OpenGL
- Render
- Flush

Clear the Buffers

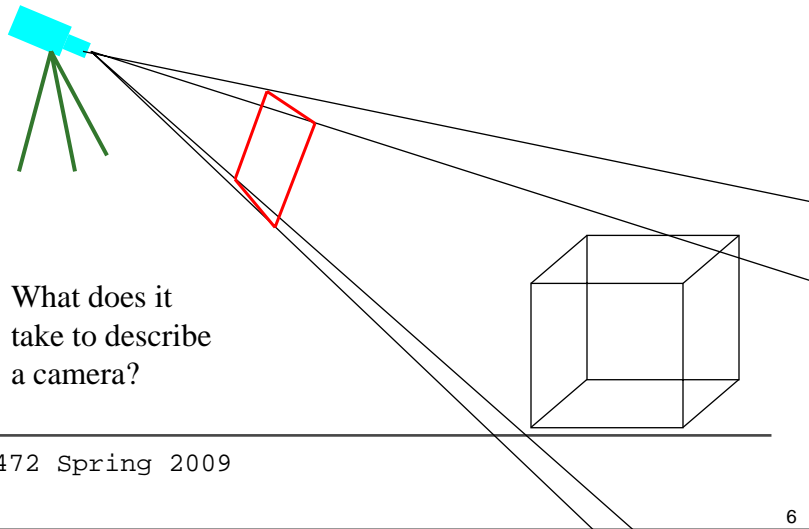
```
void CChildView::OnGLDraw(CDC *pDC)
{
    // Clear to black...
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f) ;
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    . . .
}
```

CSE 472 Spring 2009

5

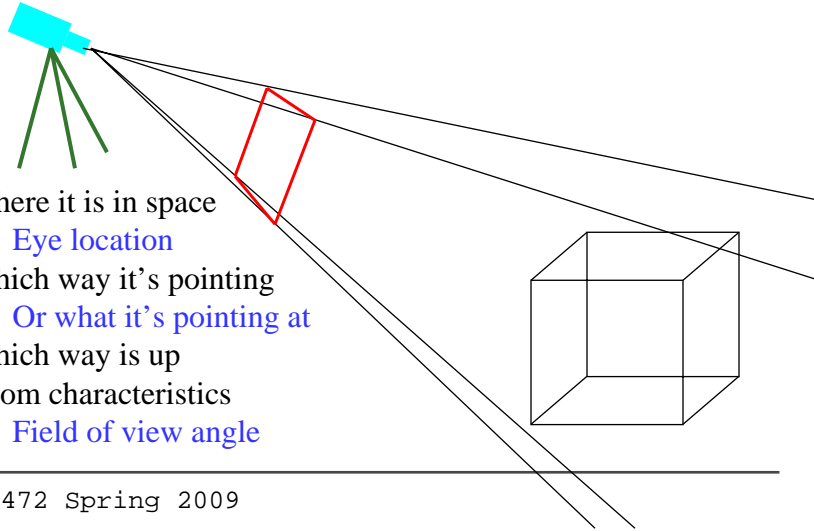
Camera Configuration



CSE 472 Spring 2009

6

Describing a Camera

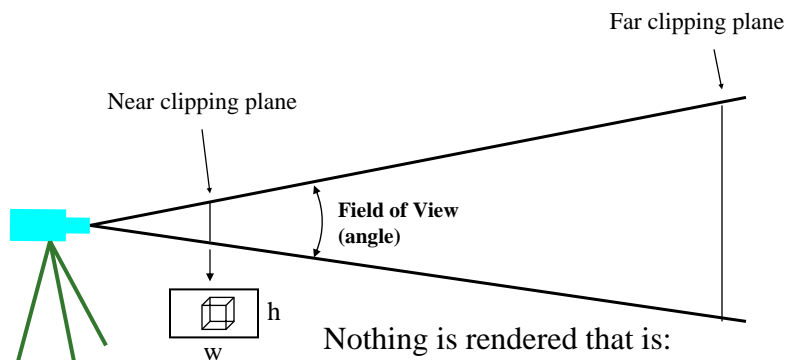


- ✓ Where it is in space
 - Eye location
- ✓ Which way it's pointing
 - Or what it's pointing at
- ✓ Which way is up
- ✓ Zoom characteristics
 - Field of view angle

CSE 472 Spring 2009

7

Setting up the Camera



Nothing is rendered that is:

- Closer than the near clipping plane
- Farther than the far clipping plane

Aspect ratio = w/h

CSE 472 Spring 2009

8

gluPerspective()

```
void gluPerspective(GLdouble fovy, GLdouble  
                  aspect, GLdouble zNear, GLdouble zFar);
```

fovy = Field of view angle

- Degrees, usually less than 90

Aspect = Ratio of width/height of window

zNear = distance to near clipping plane

zFar = distance to far clipping plane

glu = GL Utility

Important

What do the numbers **zFar**,
zNear represent?

- Always select some unit for your application
 - Inches, Feet, Meters, etc.
 - In my sample application, I used inches

gluPerspective()

```
//  
// Set up the camera  
//  
glMatrixMode(GL_PROJECTION);  
glLoadIdentity();  
  
// Determine the screen size so  
// we can determine the aspect ratio  
int width, height;  
GetSize(width, height);  
GLdouble aspectratio = GLdouble(width) /  
                        GLdouble(height);  
  
// Set the camera parameters  
gluPerspective(25.,      // Field of view.  
              aspectratio, // The aspect ratio.  
              10.,      // Near clipping  
              200.);     // Far clipping
```

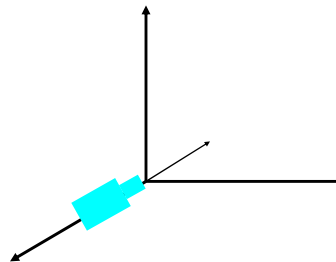
CSE 472 Spring 2009

11

After this

The camera is:

- At the origin
- Looking down the -Z axis



CSE 472 Spring 2009

12

Positioning

```
gluLookAt(eyex, eyey, eyez,  
          atx, aty, atz,  
          upx, upy, upz);
```

Eye – Where the camera is located

At – What the camera is looking at

Up – Which direction is up

- Can't be the looking direction

Matrix Modes

GL_PROJECTION

- 3D to 2D conversion
- Camera parameters

GL_MODELVIEW

- Translation, rotation, etc. of Graphical Models
- gluLookAt is a rotation and translation of your graphical model
 - the camera is really at the origin and looking down the z axis.

Using gluLookAt()

```
// Set the camera location
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

gluLookAt(50., 50., 50.,    // eye x,y,z
          0., 0., 0.,      // center x,y,z
          0., 1., 0.);     // Up direction

. . . Render from here on . . .
```



OpenGL Configurations

```
//
// Some standard parameters
//

// Enable depth test
glEnable(GL_DEPTH_TEST);

// Cull backfacing polygons
glCullFace(GL_BACK);
glEnable(GL_CULL_FACE);
```

Example: axes...

```
if(m_showaxis)
{
    glColor3d(0., 1., 1.);

    glBegin(GL_LINES);
        glVertex3d(0., 0., 0.);
        glVertex3d(12., 0., 0.);

        glVertex3d(0., 0., 0.);
        glVertex3d(0., 12., 0.);

        glVertex3d(0., 0., 0.);
        glVertex3d(0., 0., 12.);
    glEnd();
}
```

CSE 472 Spring 2009

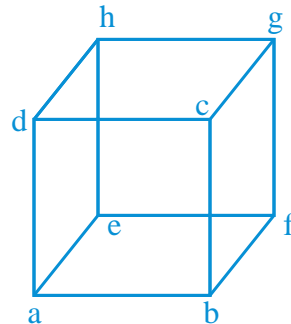
17

Example: A Cube

```
void CChildView::Cube(GLdouble size)
{
    GLdouble a[] = {0., 0., size};
    GLdouble b[] = {size, 0., size};
    GLdouble c[] = {size, size, size};
    GLdouble d[] = {0., size, size};
    GLdouble e[] = {0., 0., 0.};
    GLdouble f[] = {size, 0., 0.};
    GLdouble g[] = {size, size, 0.};
    GLdouble h[] = {0., size, 0.};

    glColor3d(0.8, 0., 0.);

    glBegin(GL_POLYGON);    // Front
        glVertex3dv(a);
        glVertex3dv(b);
        glVertex3dv(c);
        glVertex3dv(d);
    glEnd();
    . . .
```



See example program...

CSE 472 Spring 2009

18

Composite Objects

How would you build a robot?

- Body, head, arms, legs, etc.
- How would you build a hand?
 - Palm, fingers, etc.

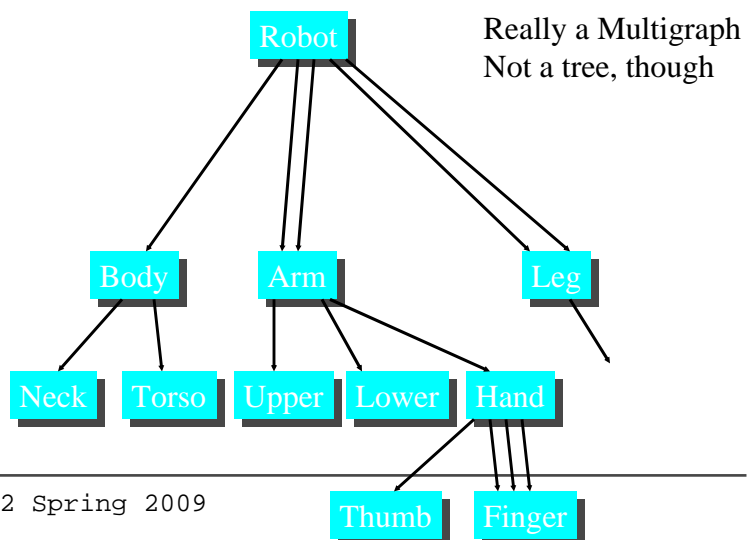
Observe

- Fingers may be the same
- Arms may be the same
- Only relationships may differ

CSE 472 Spring 2009

19

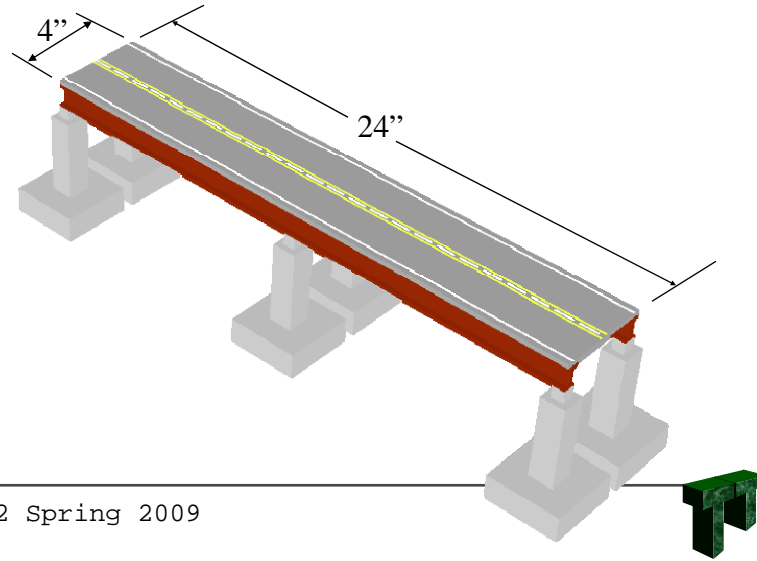
Robot Scene Graph



CSE 472 Spring 2009

20

Building a Bridge



CSE 472 Spring 2009