

Introduction to Computer Graphics

Farhana Bandukwala, PhD

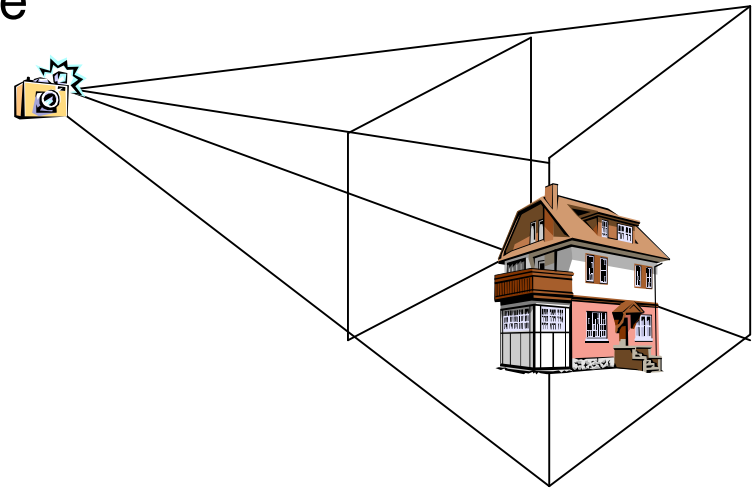
Lecture 12: Rendering in Open GL

Outline

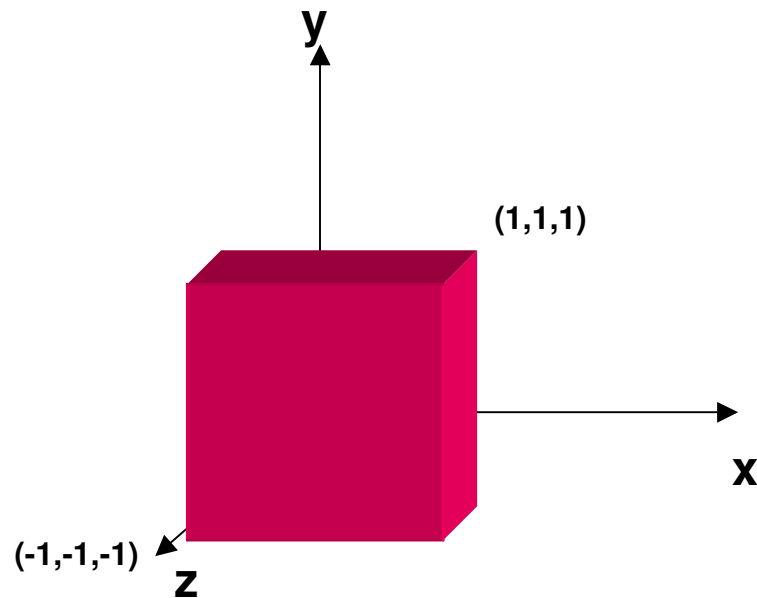
- Scene Setup
- Modeling transformations
- View setup
- Projections
- Scale to Screen (to pixels)
- Cross product
- Drawing surfaces

Scene Setup

- Model transformation
 - Arrange objects within viewing volume
- Viewing transformation
 - Camera position
 - Viewing direction
 - View-up vector
- Projection
 - Choose shape of viewing volume
- Viewport transformation
 - Project to display coordinates



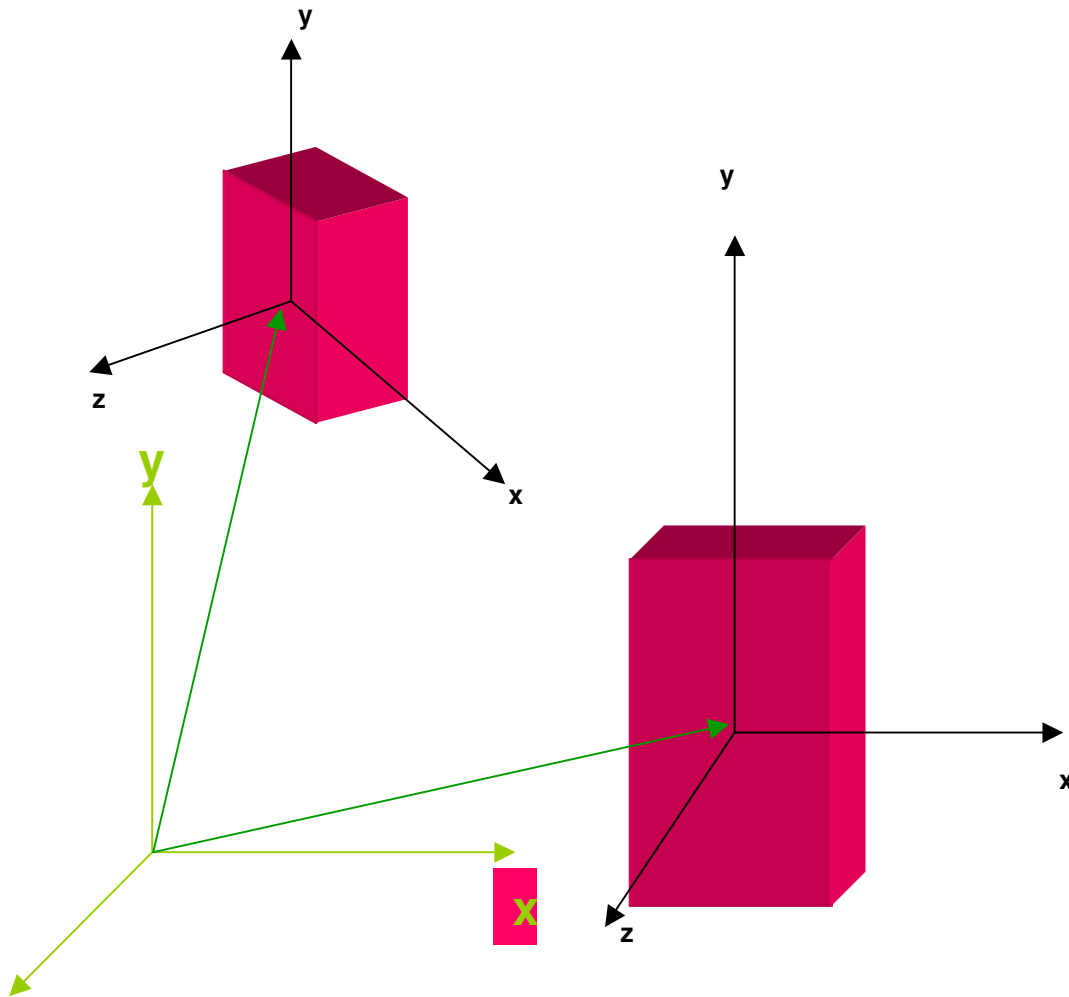
Drawing A Cube



**Object coordinates implied by
cube definition**

```
drawCube(GLfloat color[3]) {  
    glBegin(GL_QUAD_STRIP)  
        glColor3f(color[0],color[1],color[2]);  
        glVertex3f(-1,-1,-1);  
        glVertex3f(-1,1,-1);  
        glVertex3f(1,-1,-1);  
        glVertex3f(1,1,-1);  
        glVertex3f(1,-1,1);  
        glVertex3f(1,1,1);  
        glVertex3f(-1,-1,1);  
        glVertex3f(-1,1,1);  
        glVertex3f(-1,-1,-1);  
        glVertex3f(-1,1,-1);  
    glEnd()  
    glBegin(GL_QUADS)  
        glColor3f(color[0],color[1],color[2]);  
        glVertex3f(-1,-1,-1);  
        glVertex3f(1,-1,-1);  
        glVertex3f(1,-1,1);  
        glVertex3f(-1,-1,1);  
        glVertex3f(1,1,-1);  
        glVertex3f(1,1,1);  
        glVertex3f(1,1,1);  
        glVertex3f(1,1,1);  
    glEnd()  
}
```

Drawing 2 Cubes



z Object coordinates embedded within a **world** coordinate system

```
void renderScene()
{
    glClear(GL_COLOR_BUFFER_BIT |
            GL_DEPTH_BUFFER_BIT);
    // Draw the geometric primitives
    {
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();

        // DRAW CUBE 1

        glPushMatrix();
        glTranslatef(1,4,2);
        glRotated(45.0,0,1,0);
        drawCube( color1);
        glPopMatrix();

        //DRAW CUBE 2

        glPushMatrix();
        glTranslatef(5,2,1);
        glScalef(1.0,2.0,1.0);
        drawCube( color2);
        glPopMatrix();
    }
    glutSwapBuffers();
}
```

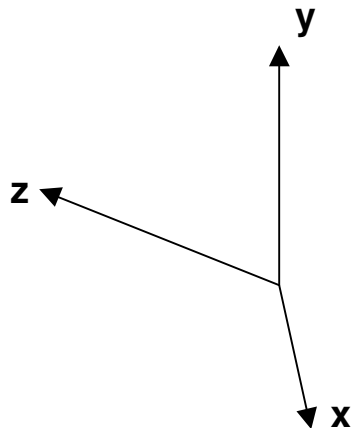
Modeling Transformation

- Rotation, translation, or scale

`glMatrixMode(GL_MODELVIEW)`

`glRotate*(angle,x,y,z), glTranslate*(x,y,z), glScale*(x,y,z)`

`glPushMatrix() & glPopMatrix()` to isolate transformations applied to different objects



Define Camera position

```
void renderScene()  
{
```

```
    glClear(GL_COLOR_BUFFER_BIT |  
           GL_DEPTH_BUFFER_BIT);  
    // Draw the geometric primitives  
    {
```

```
        glMatrixMode(GL_MODELVIEW);  
        glLoadIdentity();
```

```
        //Set up view w/ camera position &  
        orientation
```

```
        gluLookAt(3,3,-3,3,0,0,1,0,0);  
        // DRAW CUBE 1
```

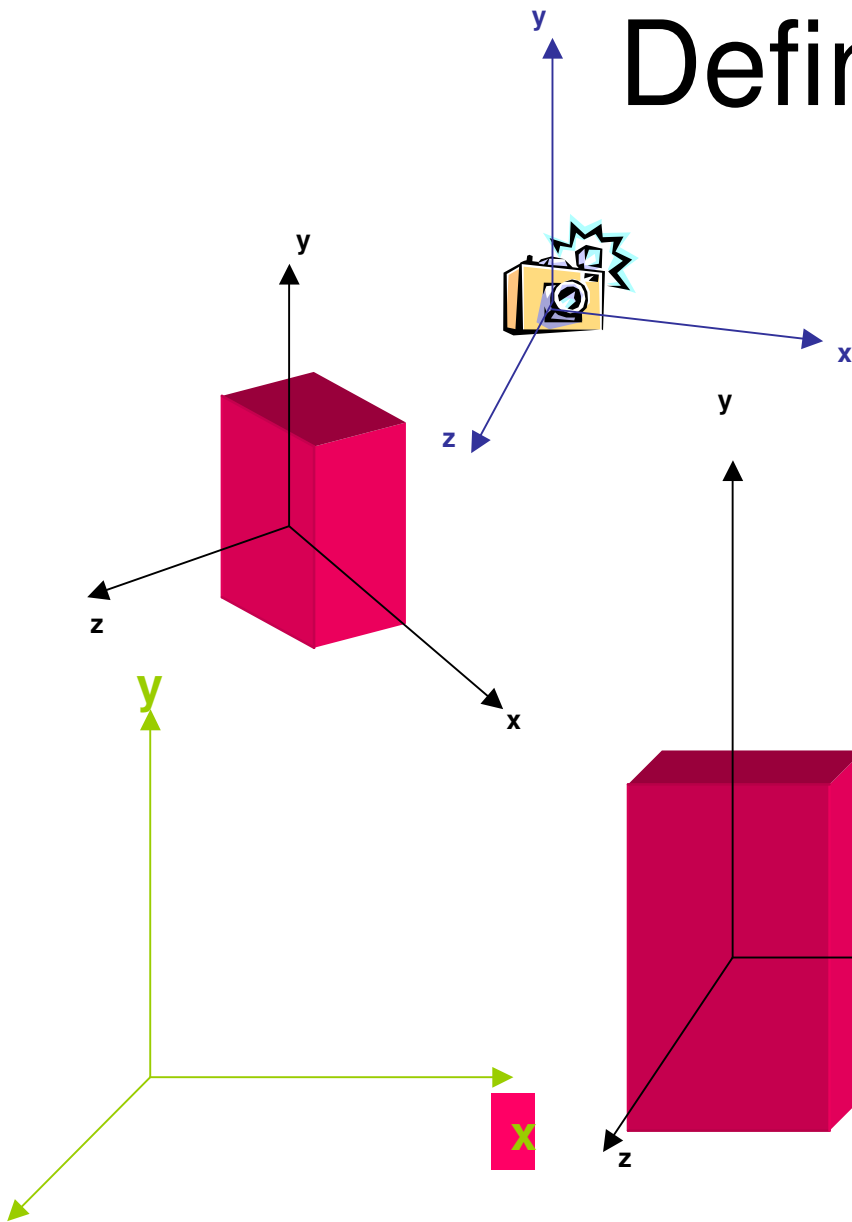
```
        glPushMatrix();  
        glTranslatef(1,4,2);  
        glRotated(45.0,0,1,0);  
        drawCube( color1);  
        glPopMatrix();
```

```
        //DRAW CUBE 2
```

```
        glPushMatrix();  
        glTranslatef(5,2,1);  
        glScalef(1.0,2.0,1.0);  
        drawCube( color2);  
        glPopMatrix();
```

```
    }  
    glutSwapBuffers();
```

```
}
```



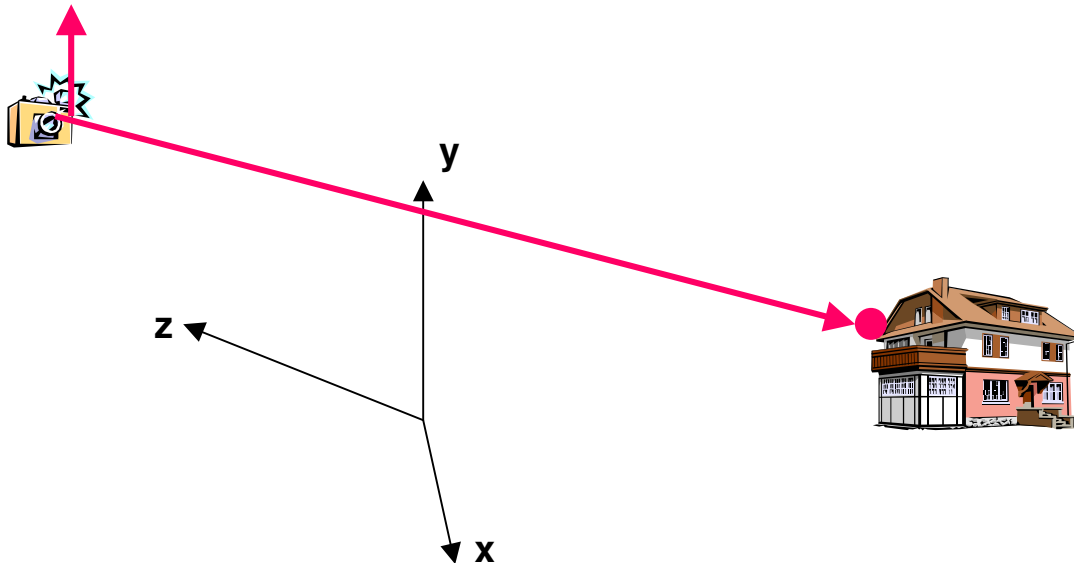
z Define a camera position & orientation within a **world** coordinate system

Viewing transformation

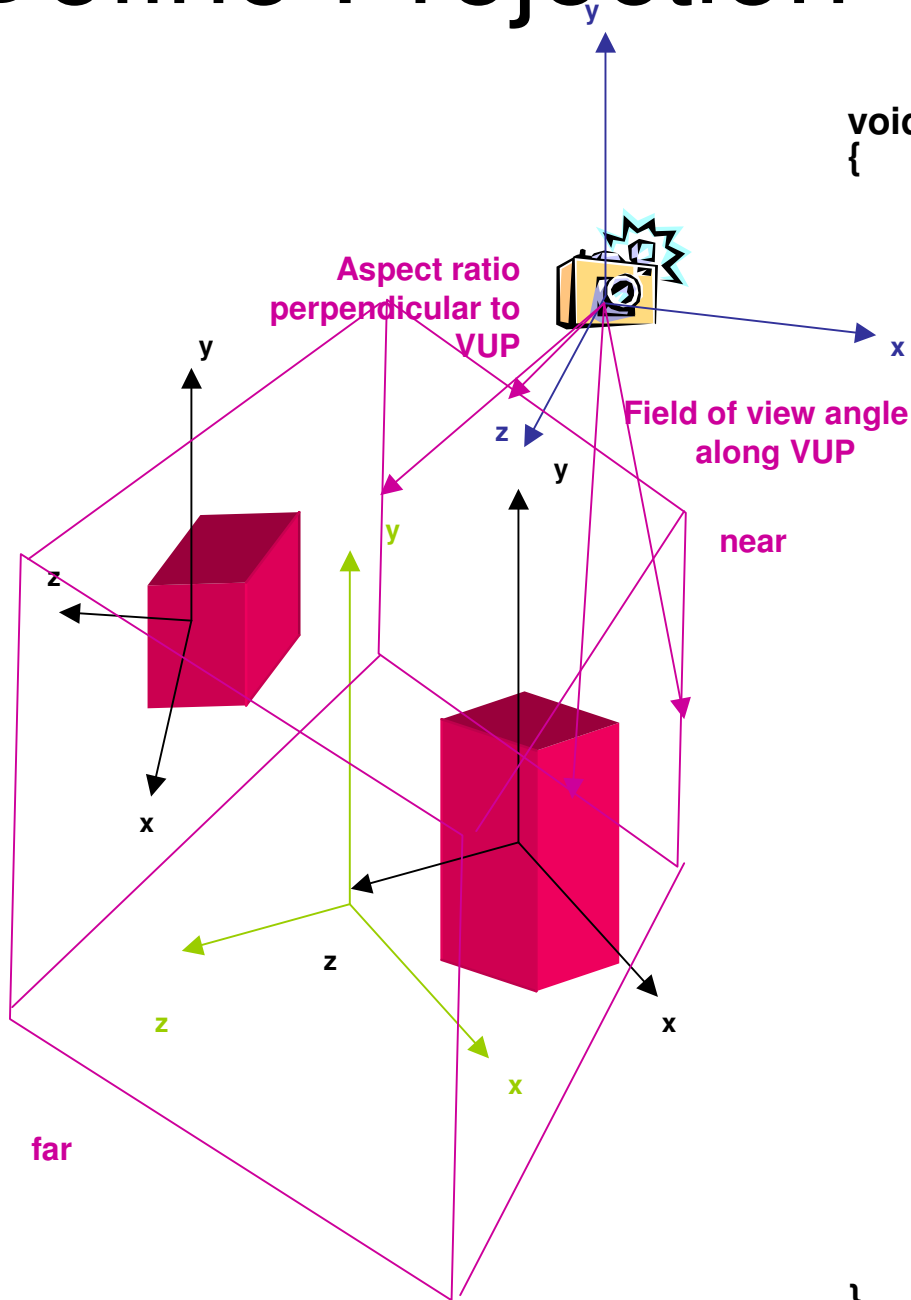
- Place camera w/ respect to world coordinates
- Specify point in object as “eye” point
 - This determines viewing direction
- Specify view-up vector

`glMatrixMode(GL_MODELVIEW)`

`gluLookAt(camera_x, camera_y, camera_z, object_x, object_y, object_z, VUP_x, VUP_y, VUP_z)`



Define Projection



```

void renderScene()
{
    glClear(GL_COLOR_BUFFER_BIT |
           GL_DEPTH_BUFFER_BIT);
    // Draw the geometric primitives
    {
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(90.0,1.0,0.50,6);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();

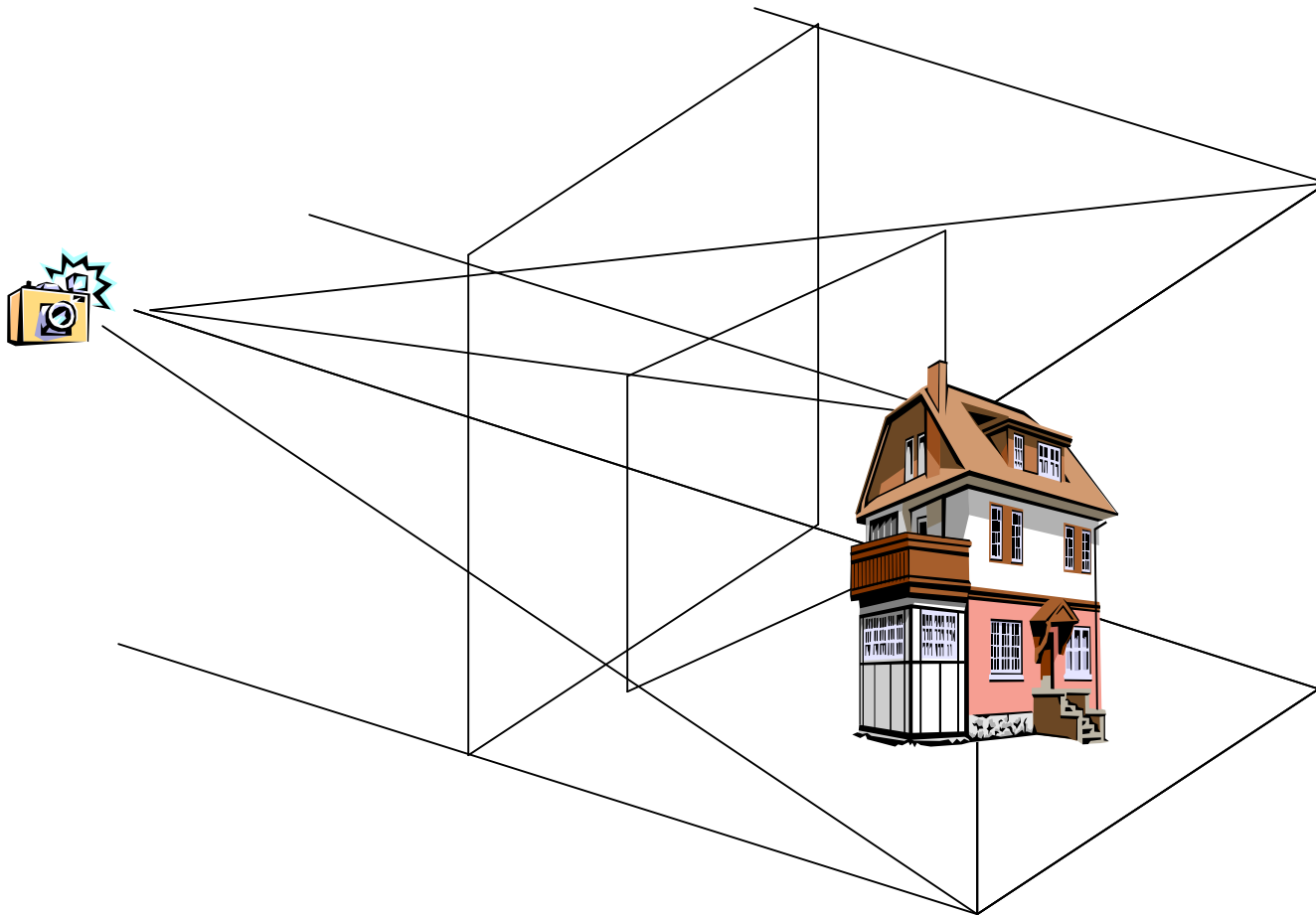
        //Set up view w/ camera position &
        orientation
        gluLookAt(3,3,-3.0,3,3,0.0,0.0,1.0,0.0);

        // DRAW CUBE 1
        glPushMatrix();
        glTranslatef(1,4,2);
        glRotated(45.0,0,1,0);
        drawCube( color1);
        glPopMatrix();

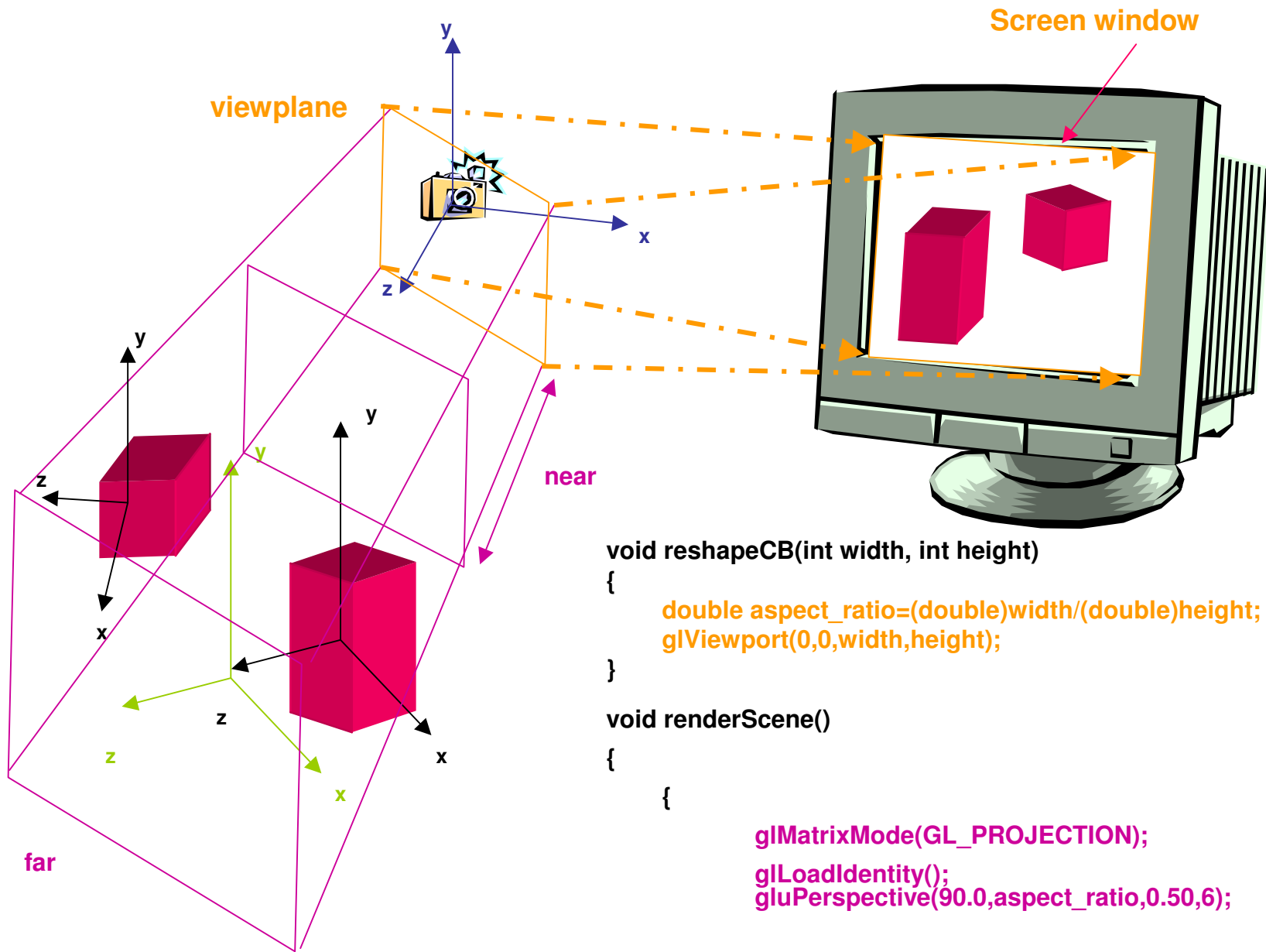
        //DRAW CUBE 2
        glPushMatrix();
        glTranslatef(5,2,1);
        glScalef(1.0,2.0,1.0);
        drawCube( color2);
        glPopMatrix();
    }
    glutSwapBuffers();
}
    
```

Projection

- Specify viewing volume shape & extents (clipping planes)
- Choose between perspective and orthographic
- (`glPerspective(field_of_view, aspect_ratio, near, far)` or `glOrtho(left, right, bottom, top, near, far)`)



Viewplane to Window



```
void reshapeCB(int width, int height)
{
    double aspect_ratio=(double)width/(double)height;
    glViewport(0,0,width,height);
}

void renderScene()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(90.0,aspect_ratio,0.50,6);
}
```

Viewport Transformation

- Defines the region on screen within which graphics will be rendered (`glViewport()`)
- Measured in window coordinates
- Corresponds to pixels on screen

