3D Graphics Programming Tools

OpenGL Camera and Transformations

Dr. Xianhui Cherry Che

x.che@qmul.ac.uk

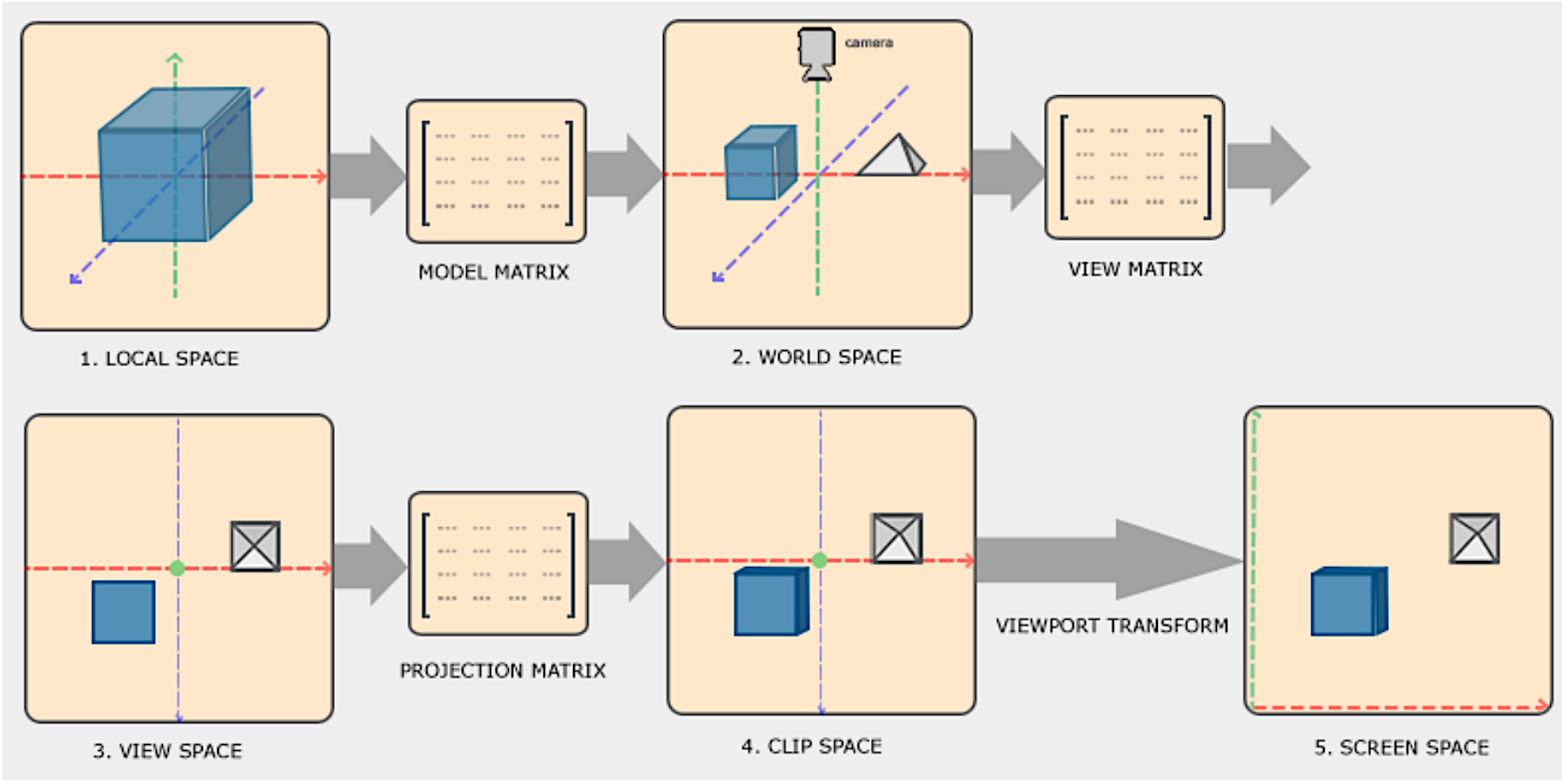
Learning Objectives

- Understand the transformation process in OpenGL development pipeline
- Get familiar with different types of camera settings and how they will affect the final viewing
- Gain further skills of transformation via the study of rotation and scaling

Topics

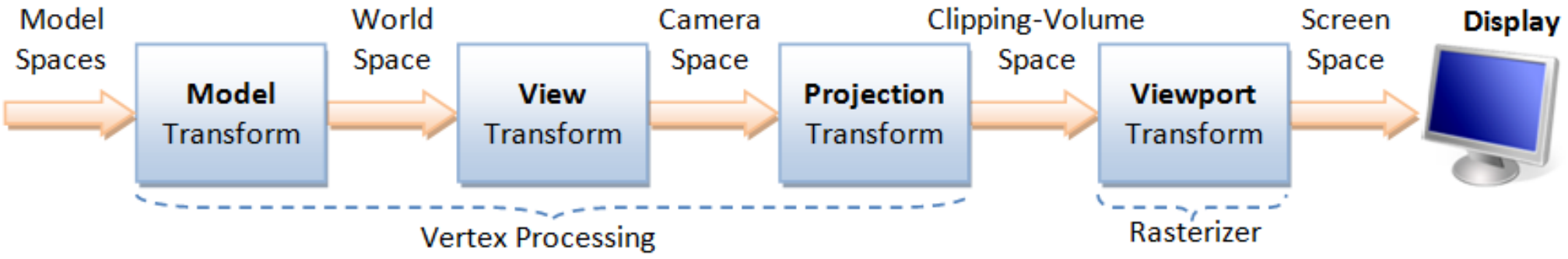
- OpenGL Transformations
- Camera
- Translation
- Rotation
- Scaling

Recap: The Global Picture



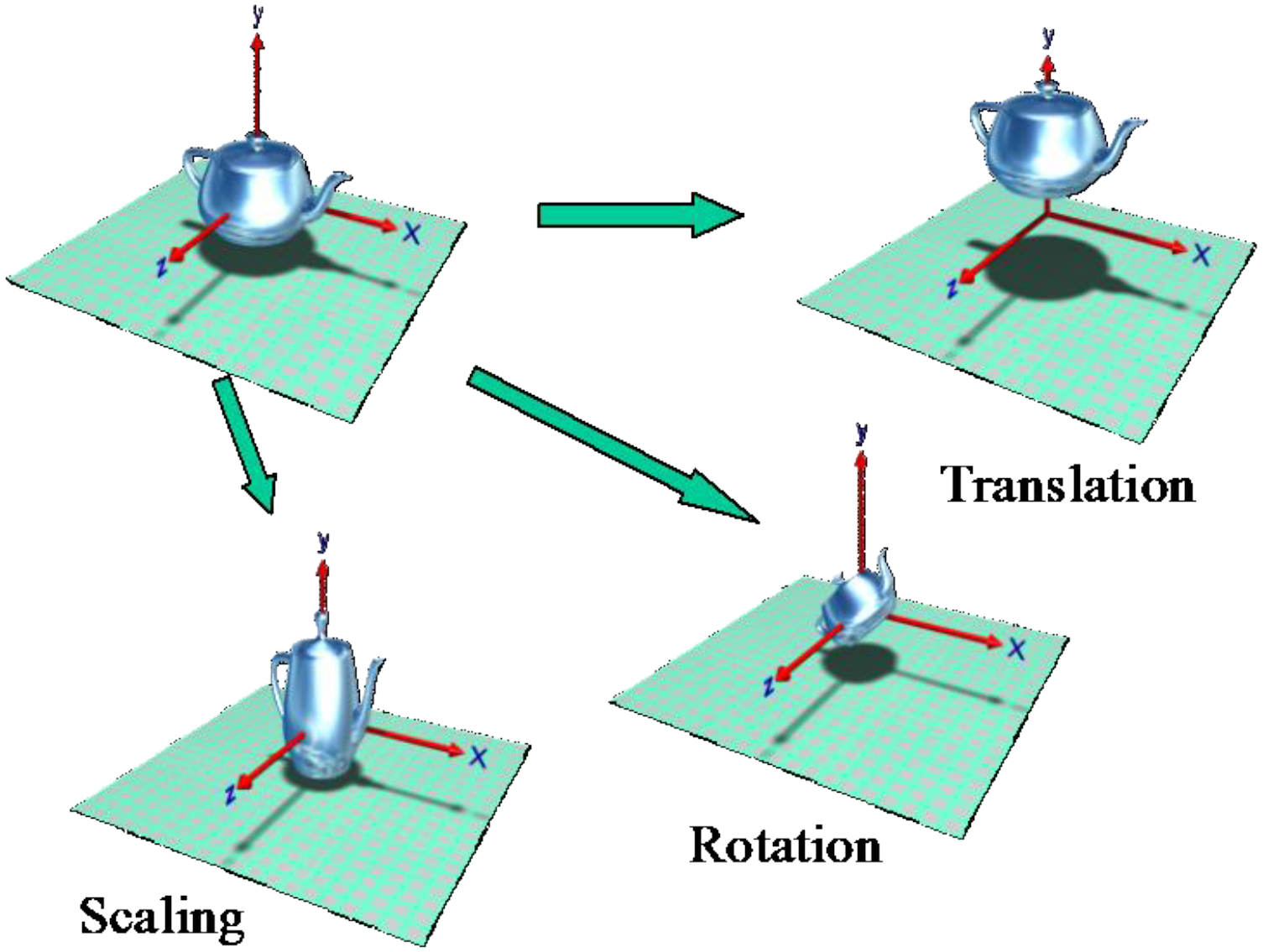
OpenGL Transformations

- Coordinates transform pipeline



Coordinates Transform Pipeline

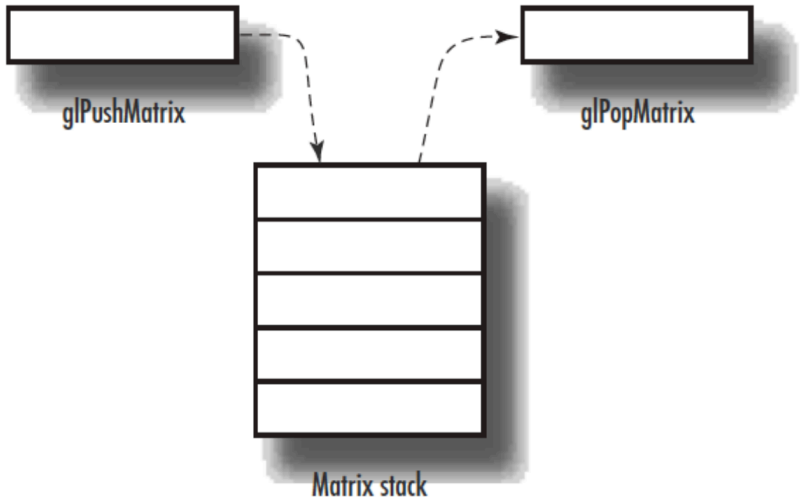
Geometric Transformations



Pushing & Popping Matrix Stacks

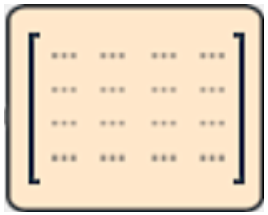
- OpenGL maintains a matrix stack for both the modelview an projection matrices.
- It may be necessary to save the current transformation state and then restore it after further transformation changes.
- Three steps:

1. Push the current matrix onto the stack with `glPushMatrix` to save it.
2. Make changes to the current matrix.
3. Popping the matrix off the stack with `glPopMatrix` then restores it.



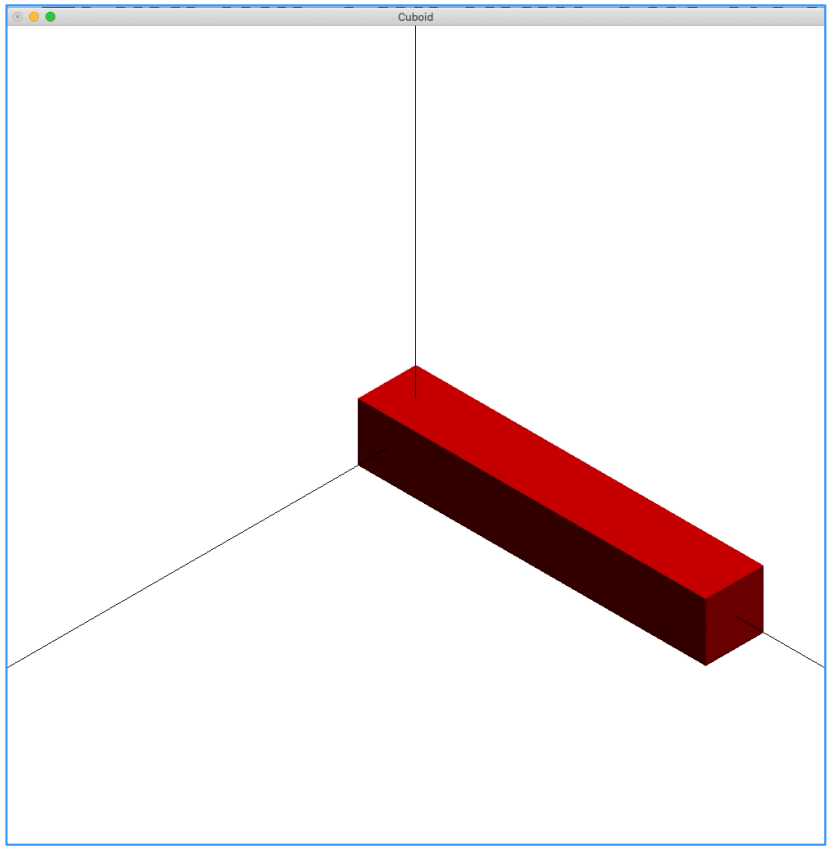
Pushing & Popping Matrix Stacks

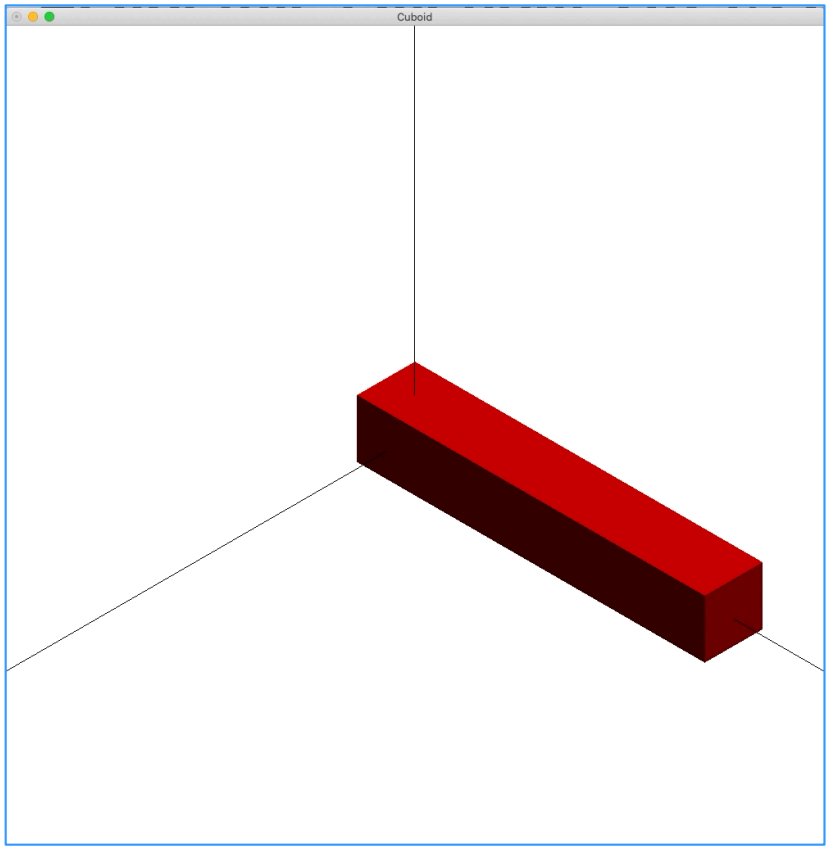
- In simple words:
  1. Put the current matrix in a safe to protect it.
  2. Make some changes (i.e. transformations)
  3. Take the matrix out.



Current Matrix

Pushing & Popping Matrix Stacks





Without pushing & popping

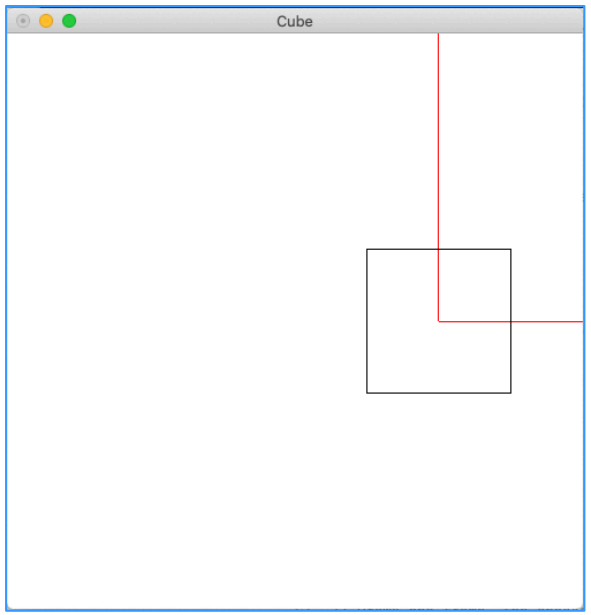
```
glTranslated(0, Y, 0);
drawCuboid(0, 0, 0, 1); //Self-defined
```

With pushing & popping

```
glPushMatrix();
glTranslated(0, Y, 0);
drawCuboid(0, 0, 0, 1); //Self-defined
glPopMatrix();
```

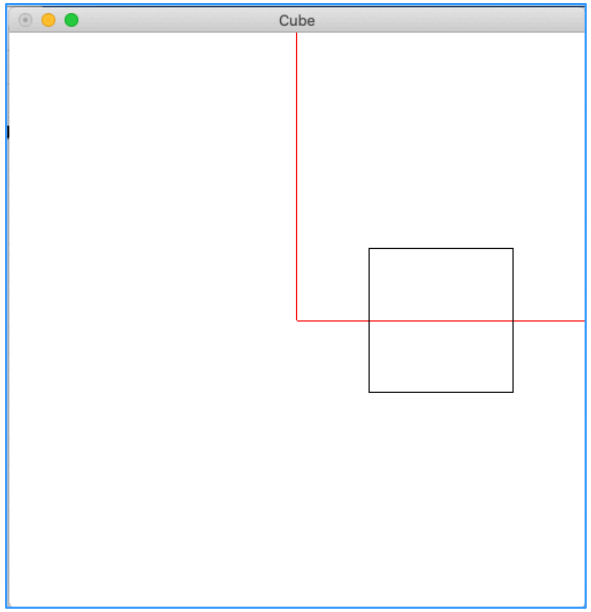
Pushing & Popping Matrix Stacks

```
//Draw cube and move +1 on x axis
glTranslatef(1, 0, 0);//Explain later
glutWireCube(1);

//Draw axis
glColor3f (1.0, 0.0, 0.0);
glBegin(GL_LINES);
glVertex3f(0, 0, 0);
glVertex3f(2, 0, 0);
glVertex3f(0, 0, 0);
glVertex3f(0, 2, 0);
glVertex3f(0, 0, 0);
glVertex3f(0, 0, 2);
glEnd();
```



Without preserving the current state, the changes affect the subsequent drawing, i.e. the drawing of the axis follows x+1 translation.

```
glPushMatrix();
glTranslatef(1, 0, 0);
glutWireCube(1);
glPopMatrix();

glColor3f (1.0, 0.0, 0.0);
glBegin(GL_LINES);
glVertex3f(0, 0, 0);
glVertex3f(2, 0, 0);
glVertex3f(0, 0, 0);
glVertex3f(0, 2, 0);
glVertex3f(0, 0, 0);
glVertex3f(0, 0, 2);
glEnd();
```



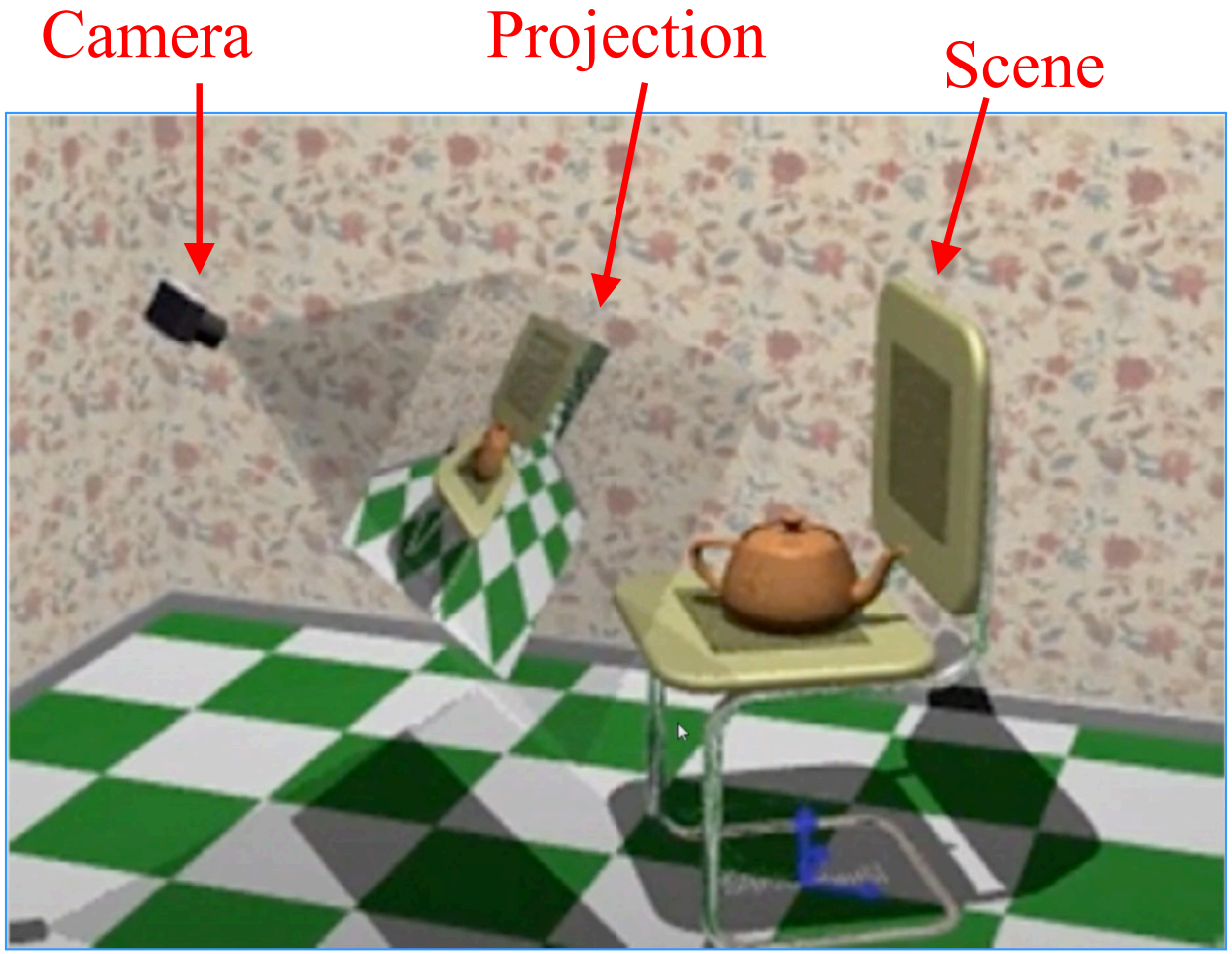
The current state is saved by pushing & popping matrix stacks, i.e. the drawing of the axis follows the original matrix before the cube was created and moved x+1.

Topics

- OpenGL Transformations
- Camera
- Translation
- Rotation
- Scaling

The `glLookAt` Function

- **`gluLookAt`** — places a camera relative to the scene.



The `glLookAt` Function

- **`gluLookAt`** — define a viewing transformation

```
void gluLookAt( GLdouble eyeX,
                GLdouble eyeY,
                GLdouble eyeZ,
                GLdouble centerX,
                GLdouble centerY,
                GLdouble centerZ,
                GLdouble upX,
                GLdouble upY,
                GLdouble upZ);
```

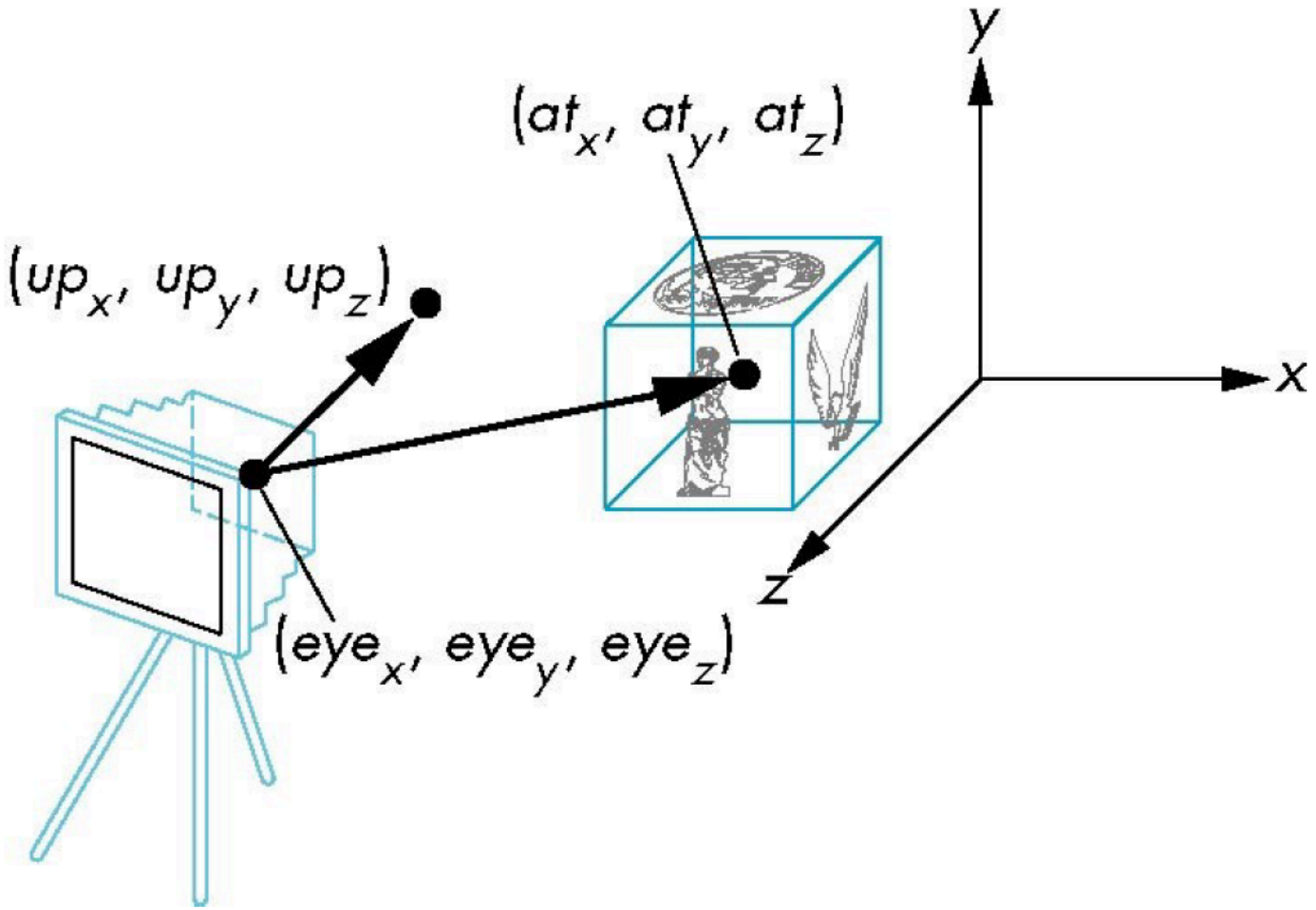
The position of the eye point

The position of the reference point

The direction of the up vector

The gluLookAt Function

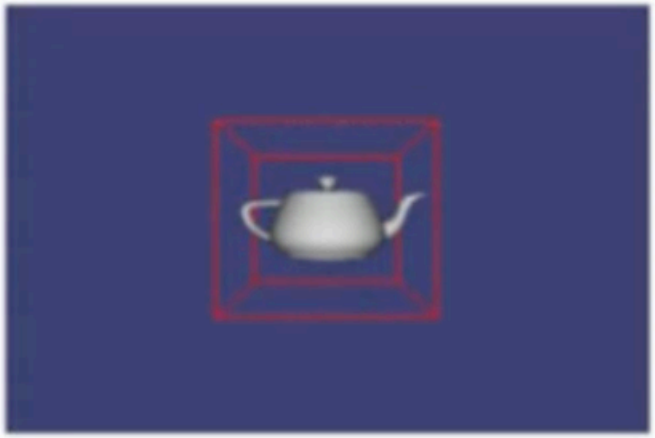
```
gluLookAt(eyex, eyey, eyez, atx, aty, atz, upx, upy, upz);
```



gluLookAt Examples – Eye Coordinates

By default the camera is pointing towards negative z direction.

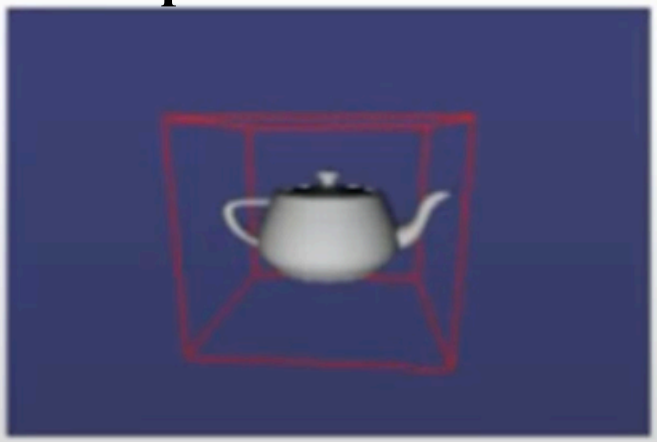
Example 1:



```
gluLookAt(0, 0, 15,     //eye(x, y, z)
          0, 0, 0,      //at (x, y, z)
          0, 1, 0);     //up (x, y, z)
```

In Example 1, the camera is moved towards the positive z direction so the teapot looks further away.

Example 2:



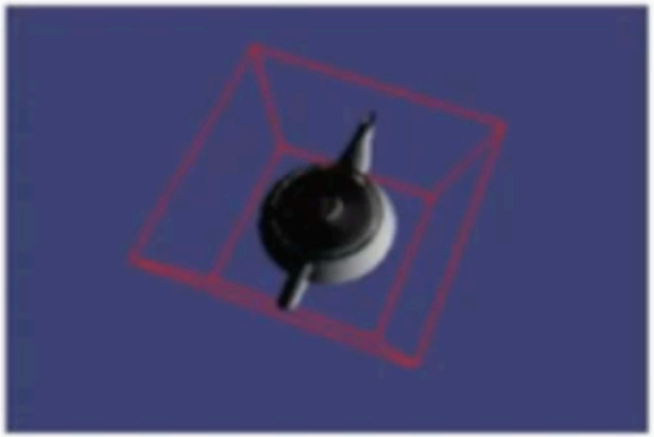
```
gluLookAt(1, 2, 11,     //eye(x, y, z)
          0, 0, 0,      //at (x, y, z)
          0, 1, 0);     //up (x, y, z)
```

In Example 2, the camera is moved to (1, 2, 11) to have a perturbed view of the teapot. Note that the teapot is still the centre of the picture because of the "at coordinates" are (0, 0, 0).

Queen Mary University of London

gluLookAt Examples – Eye Coordinates

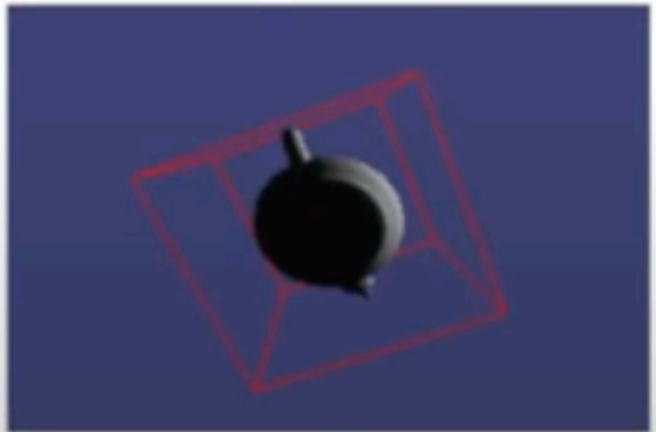
Example 3:



```
gluLookAt(-2, 11, 5,   //eye(x, y, z)
          0, 0, 0,     //at (x, y, z)
          0, 1, 0);    //up (x, y, z)
```

In Example 3, the y position of the camera is very high so the camera is well above the teapot.

Example 4:



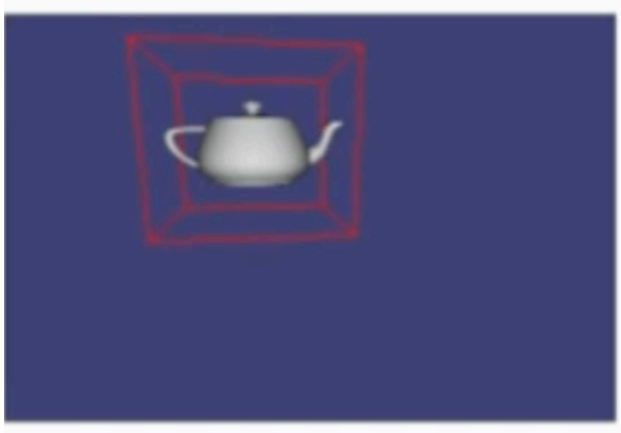
```
gluLookAt(-2, -11, 1,   //eye(x, y,
z)
          0, 0, 0,     //at (x, y, z)
          0, 1, 0);    //up (x, y, z)
```

In Example 4, the y position of the camera is very low so the picture shows the bottom of the teapot.

Queen Mary
University of London

gluLookAt Examples – At Coordinates

Example 5:



```
gluLookAt(0, 0, 14,   //eye(x, y, z)
          2, -3, 0,   //at (x, y, z)
          0, 1, 0);   //up (x, y, z)
```

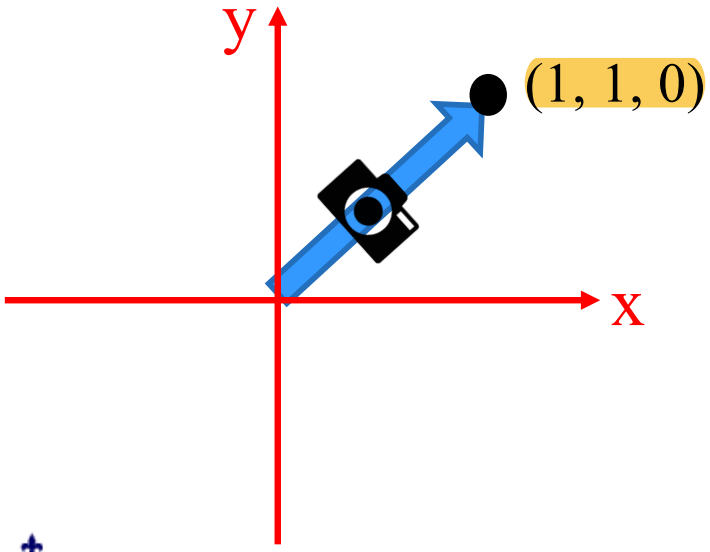
In Example 4, the teapot is not at the centre of the picture because the camera is pointing at (2, -3, 0).

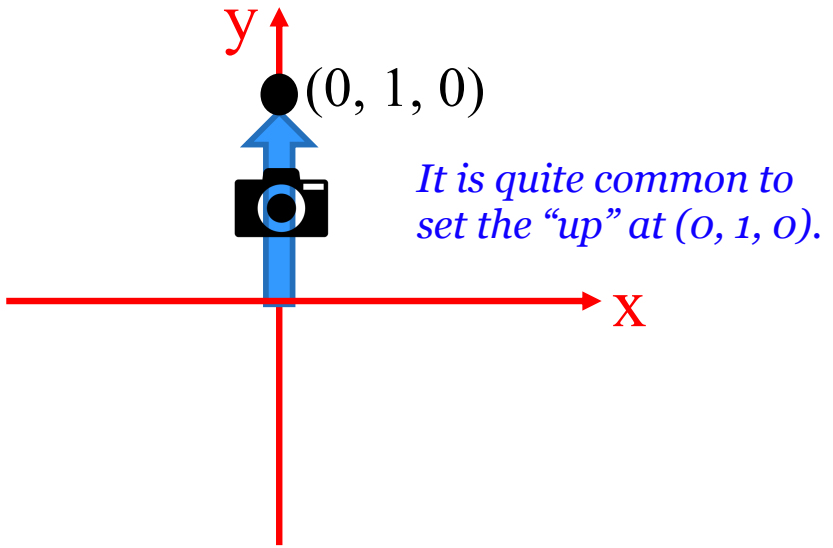
gluLookAt Examples – Up Coordinates

Example 6:

```
gluLookAt(0, 0, 14,   //eye(x, y, z)
          0, 0, 0,   //at (x, y, z)
          1, 1, 0);   //up (x, y, z)
```

In Example 6, the up direction of the camera is pointing at (1, 1, 0) so the image is tilted.





It is quite common to set the "up" at (0, 1, 0).

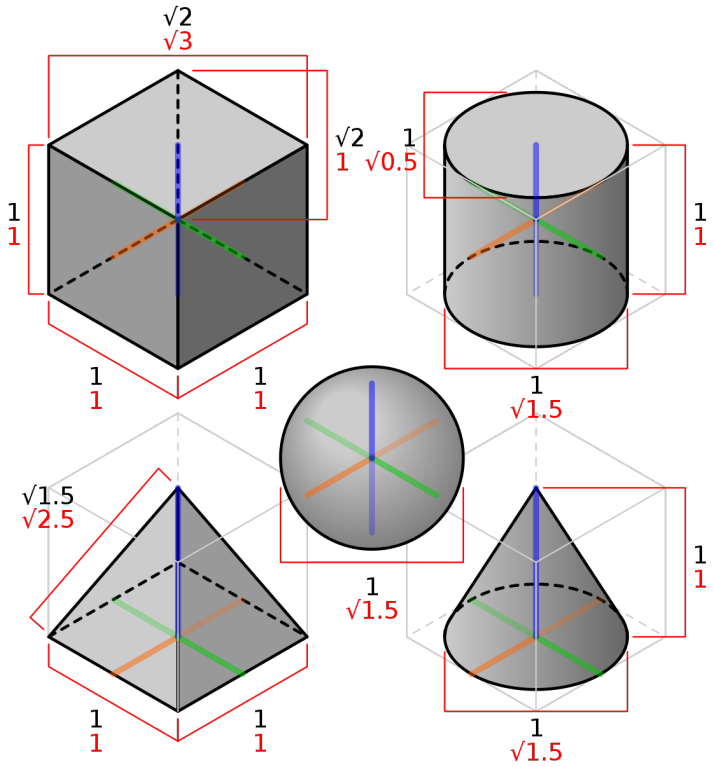
gluLookAt Examples – Demo

Queen Mary University of London

Isometric View with gluLookAt

- Isometric view can be achieved using **gluLookAt**.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(1.0, 1.0, 1.0 , 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
```

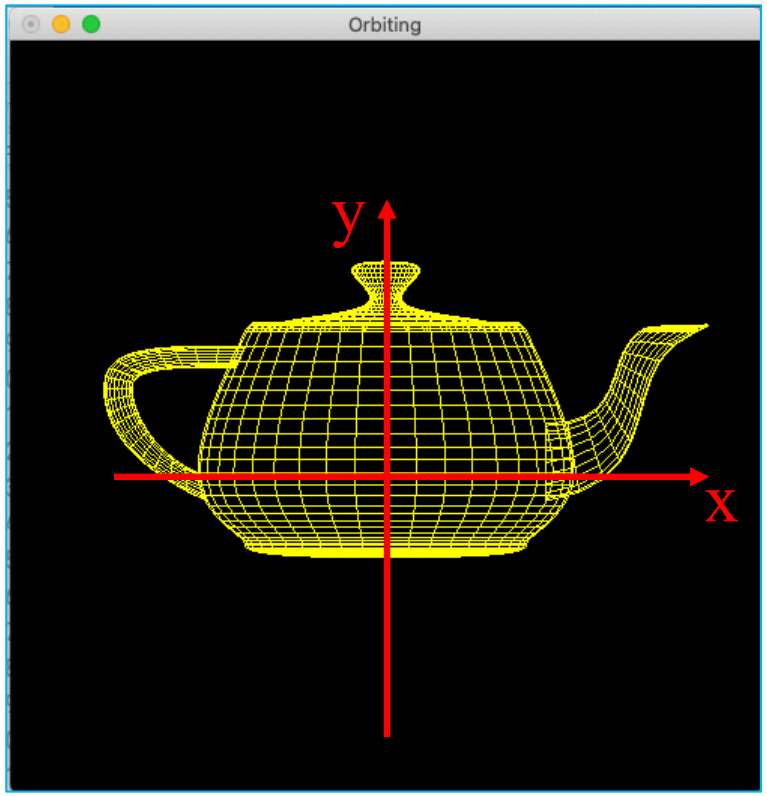


Camera Pseudocode

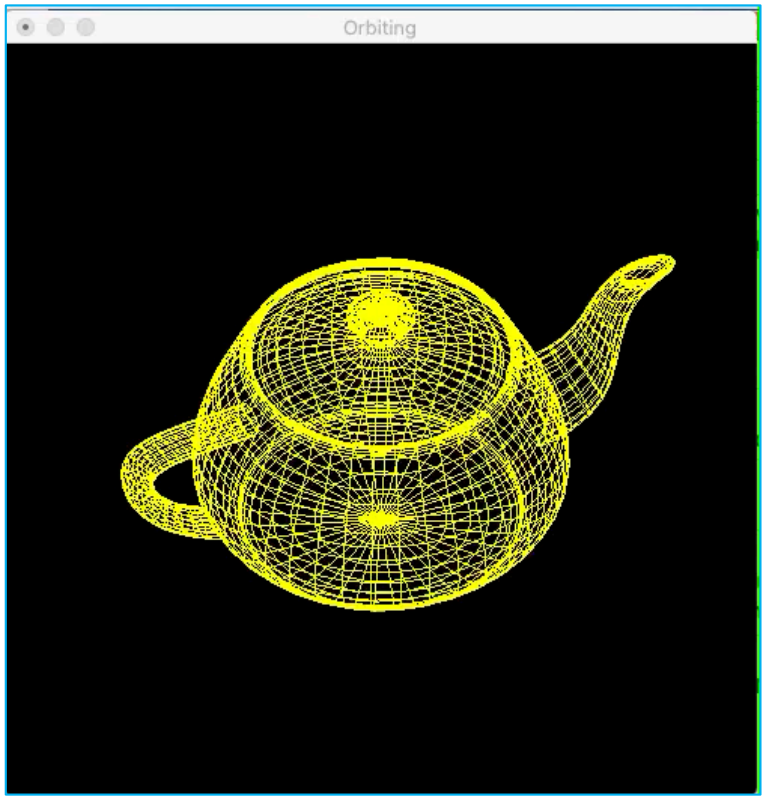
```
void display () {
        Set matrix in model view;
        gluLookAt in a proper viewing angle if needed;
        Draw things, transformation, etc;
}
void init () {
        set colours, etc;
        set projection matrix;
}
int main () {
        set callback functions;
}
```

Orbiting Camera

- Create a teapot where the camera is orbiting around on the top.

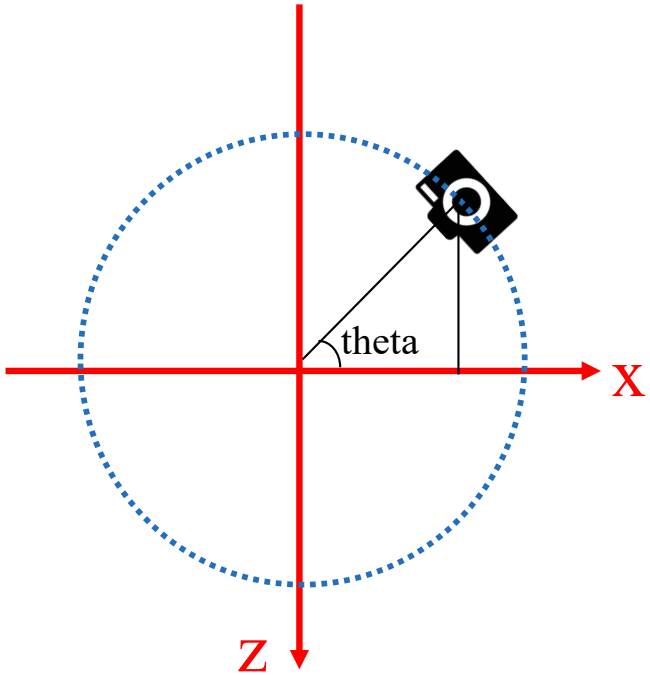


Original Still Image



Animation with Orbiting Camera on the Top

Orbiting Camera




```
#include <math.h>
GLdouble theta = 0;


void idle() {
     theta += 0.1;
     glutPostRedisplay();
}
```

```
gluLookAt(cos(theta), 1, sin(theta), 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
```

Orbiting Camera – Full Program

```
#include <GLUT/glut.h>
#include <math.h>
GLdouble theta = 0;
void idle() {
    theta += 0.1;
    glutPostRedisplay();
}
void display(){
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(cos(theta), 1, sin(theta), 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
    glColor3f (1, 1, 0);
    glutWireTeapot(1);
    glutSwapBuffers();
}
void init(void) {
    glClearColor(0.0, 0, 0, 0.0);
    glColor3f (0.0, 0.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-2.0, 2.0, -2.0, 2.0, -2.0, 100.0);
}
```

```
int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Orbiting");
    glutDisplayFunc(display);
    glutIdleFunc(idle);
    init();
    glEnable(GL_DEPTH_TEST);
    glutMainLoop();
}
```

Topics

- OpenGL Transformations
- Camera
- Translation
- Rotation
- Scaling

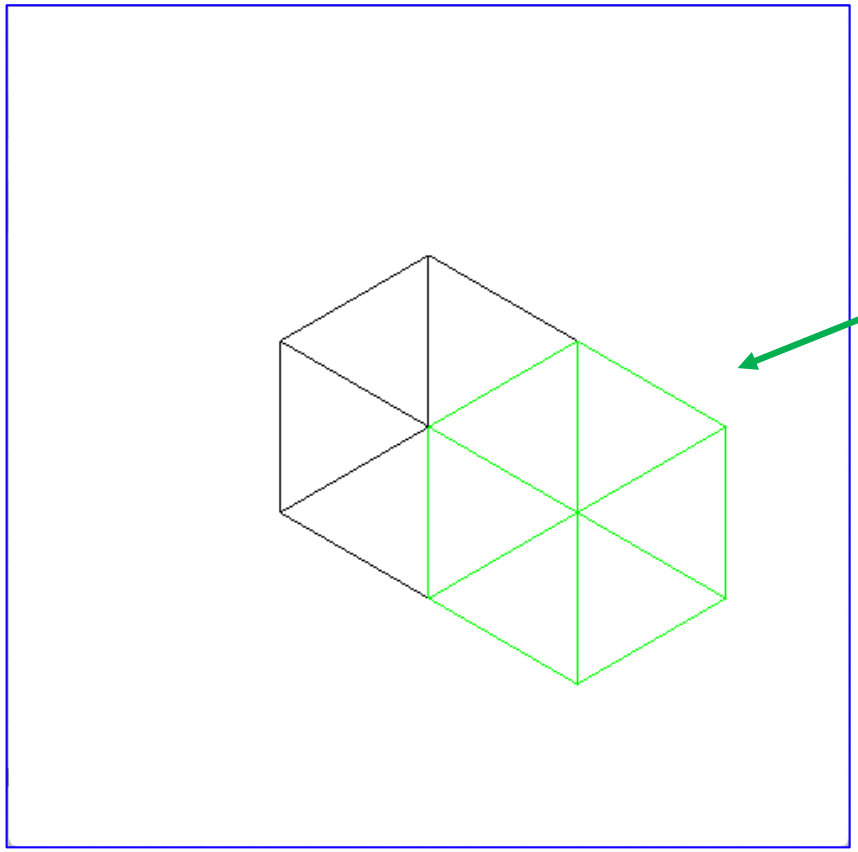
Translation

- **`glTranslate`** — multiply the current matrix by a translation matrix

```
void glTranslated( GLdouble x,
                   GLdouble y,
                   GLdouble z);

void glTranslatef( GLfloat x,
                   GLfloat y,
                   GLfloat z);
```

Two Cubes

- Display two adjacent cubes in isometric view.



The green cube is on the +1 position to the black cube along the x axis.

Two Cubes

```
#include <GLUT/glut.h>
void display(){
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(1.0, 1.0, 1.0 , 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f (0.0, 0.0, 0.0);
    glutWireCube(1);

    glPushMatrix();
    glTranslated(1, 0, 0);
    glColor3f (0.0, 1.0, 0.0);
    glutWireCube(1);
    glPopMatrix();

    glFlush();
}
```

```
void init(void) {
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f (0.0, 0.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-2.0, 2.0, -2.0, 2.0, -2.0, 100.0);
}
int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Two Cubes");
    glutDisplayFunc(display);
    init();
    glutMainLoop();
}
```

Queen Mary
University of London

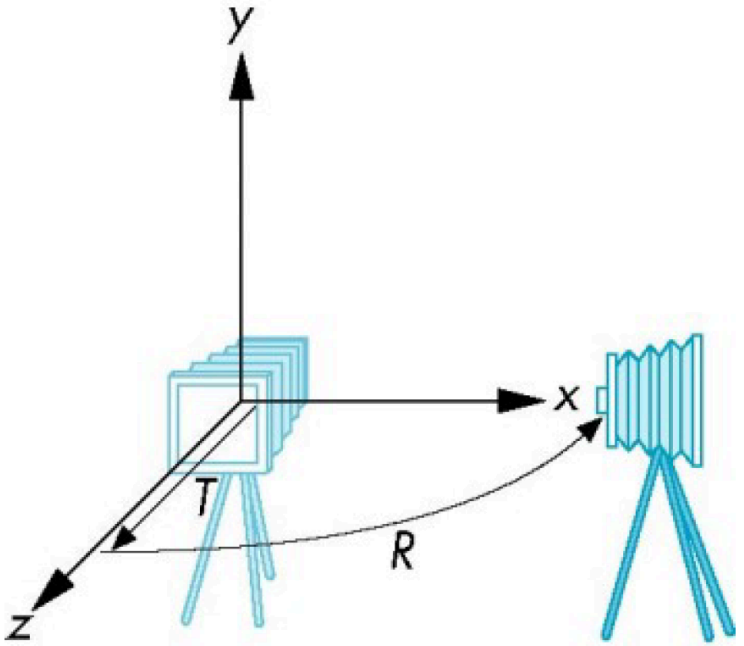
Topics

- OpenGL Transformations
- Camera
- Translation
- Rotation
- Scaling

Rotation

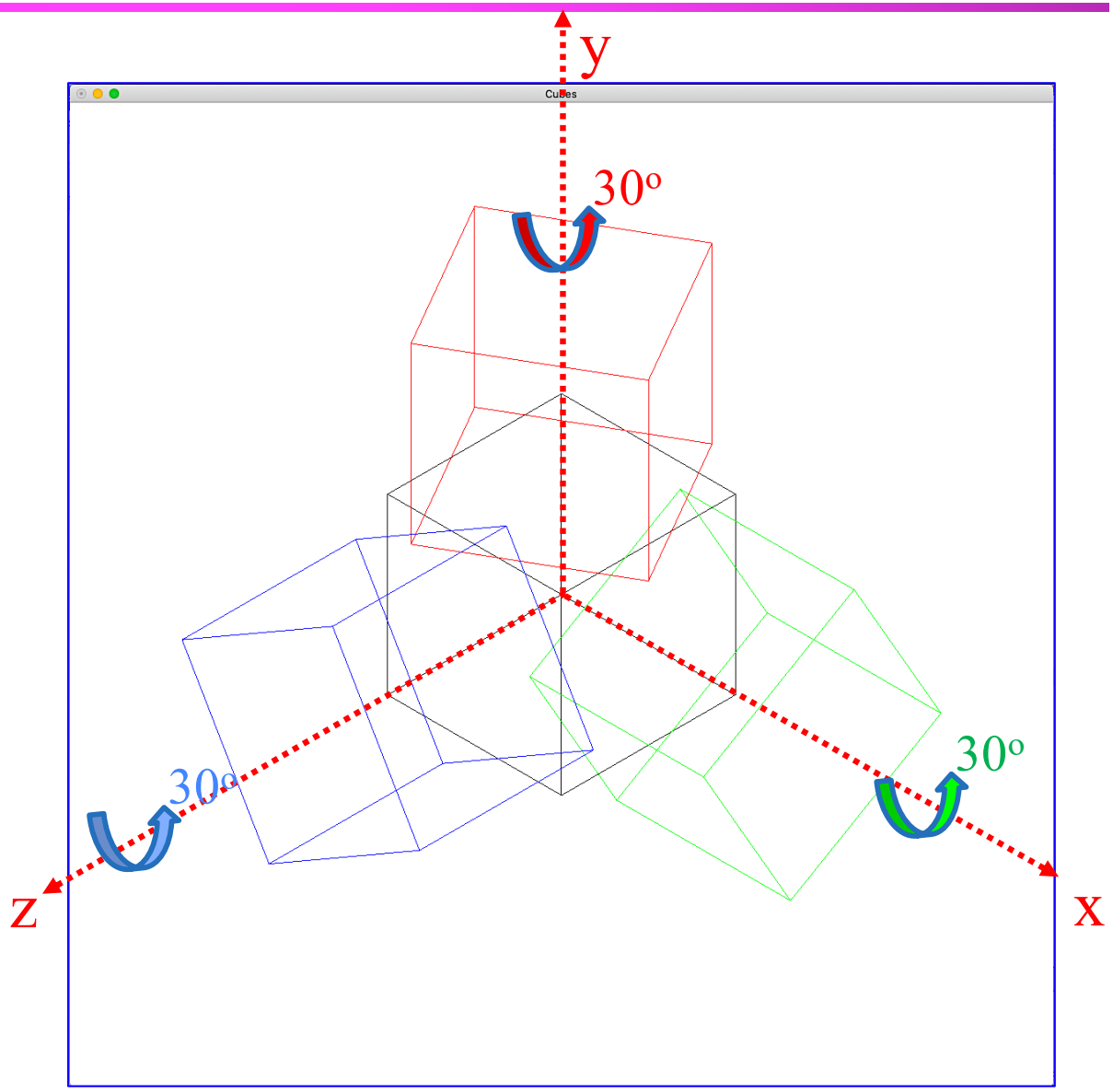
- **`glRotate`** — multiply the current matrix by a rotation matrix

```
void glRotated( GLdouble angle,
                GLdouble x,
                GLdouble y,
                GLdouble z);

void glRotatef( GLfloat angle,
                GLfloat x,
                GLfloat y,
                GLfloat z);
```



Rotation Examples

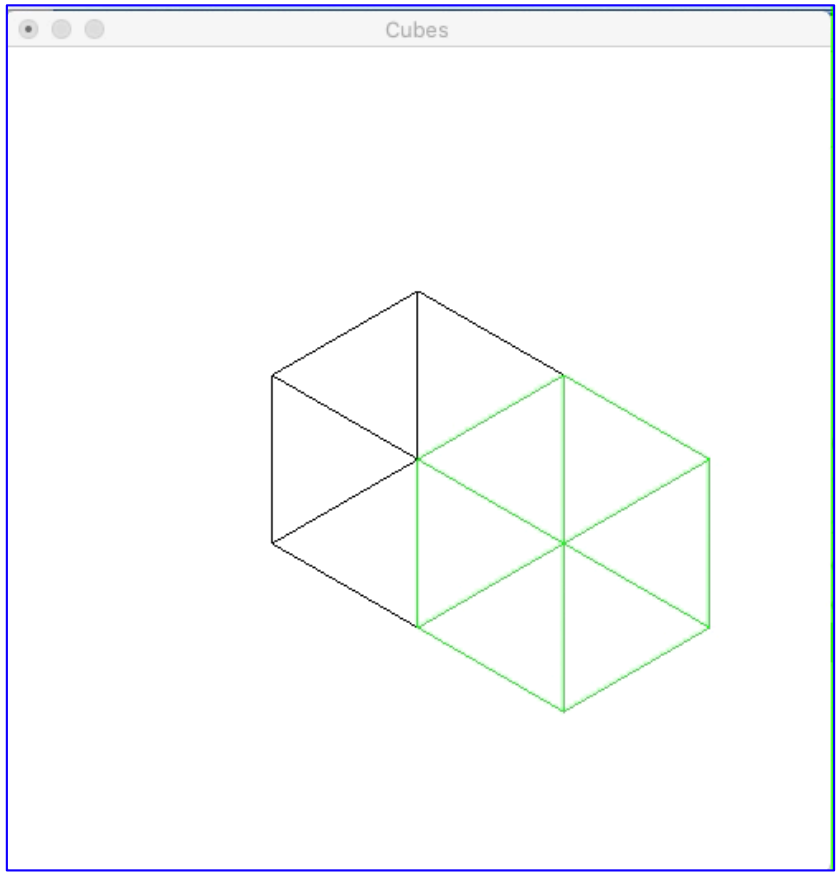
```
glColor3f (0.0, 0.0, 0.0);
glutWireCube(1);      ← Black cube

glPushMatrix();
glTranslated(1, 0, 0);
glRotated(30, 10, 0, 0);
glColor3f (0.0, 1.0, 0.0);
glutWireCube(1);      ← Green cube
glPopMatrix();

glPushMatrix();
glTranslated(0, 1, 0);
glRotated(30, 0, 10, 0);
glColor3f (1.0, 0.0, 0.0);
glutWireCube(1);      ← Red cube
glPopMatrix();

glPushMatrix();
glTranslated(0, 0, 1);
glRotated(30, 0, 0, 10);
glColor3f (0.0, 0.0, 1.0);
glutWireCube(1);      ← Blue cube
glPopMatrix();
```



Isometric view

Rotation Animation Example

- Rotate a cube around x axis. Draw the original cube as a reference point.



Rotation Animation Example

```
#include <GLUT/glut.h>
GLdouble angle=0;
void idle() {
    angle++;
    glutPostRedisplay();
}
void display(){
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(1.0, 1.0, 1.0 , 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
    glColor3f (0.0, 0.0, 0.0);
    glutWireCube(1);

    glPushMatrix();
    glTranslated(1, 0, 0);
    glRotated(angle, 10, 0, 0);
    glColor3f (0.0, 1.0, 0.0);
    glutWireCube(1);
    glPopMatrix();

    glutSwapBuffers();
}
```

```
void init(void) {
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f (0.0, 0.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-2.0, 2.0, -2.0, 2.0, -2.0, 100.0);
}
```

```
int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Cubes");
    glutDisplayFunc(display);
    glutIdleFunc(idle);
    init();
    glutMainLoop();
}
```

- Multiple translations can be put in the same line at the same time.
- Multiple rotations must be done one at a time

Topics

- OpenGL Transformations
- Camera
- Translation
- Rotation
- Scaling

Scaling

glScale — multiply the current matrix by a general scaling matrix

```
void glScaled( GLdouble x,
               GLdouble y,
               GLdouble z);

void glScalef( GLfloat x,
               GLfloat y,
               GLfloat z);
```

x, *y*, *z*

Specify scale factors along the *x*, *y*, and *z* axes, respectively.

Questions?

x.che@qmul.ac.uk