Programming with OpenGL: Three Dimensions

Dr. Marie-Luce Bourguet
(marie-luce.bourguet@qmul.ac.uk)

Slides adapted from Interactive Computer Graphics 4E © Addison-Wesley

Queen Mary
University of London

1

Objectives

- Develop a more sophisticated three-dimensional example
- Introduce hidden-surface removal

Queen Mary
University of London

2

Three-dimensional Applications

- In OpenGL, two-dimensional applications are a special case of three-dimensional graphics
- Going to 3D
  - Not much changes
  - Use **`glVertex3*( )`**
  - Have to worry about the order in which polygons are drawn or use hidden-surface removal
  - Polygons should be simple, convex, flat

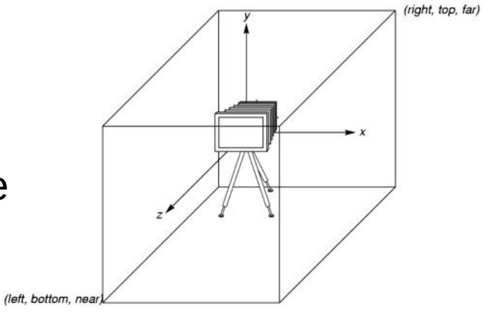
Queen Mary University of London

3

OpenGL Camera

- OpenGL places a camera at the origin in object space pointing in the negative z direction

- Remember: the default viewing volume is a box centered at the origin with a side of length 2



Queen Mary University of London

4

Transformations and Viewing

- In OpenGL, projection is carried out by a projection matrix (transformation)
- There is only one set of transformation functions so we must set the matrix mode first

```
glMatrixMode (GL_PROJECTION)
```

- Transformation functions are incremental so we start with an identity matrix and alter it with a projection matrix that gives the view volume

```
glLoadIdentity();
glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
```

Queen Mary
University of London

5

Colorcube

- We will go through the code of a program that creates a 3D cube and rotates it in an interactive way

Queen Mary
University of London

6

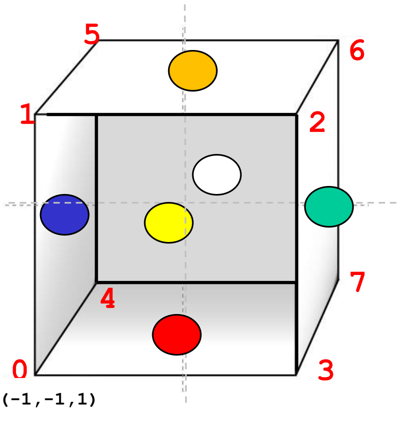
Colorcube 1 (1)

```
GLfloat X = 0.5;                /* A scaling factor */

GLfloat GlobalVertices[][3] = {{-1.0,-1.0,1.0},
       {-1.0,1.0,1.0}, {1.0,1.0,1.0}, {1.0,-1.0,1.0},
       {-1.0,-1.0,-1.0}, {-1.0,1.0,-1.0}, {1.0,1.0,-1.0},
       {1.0,-1.0,-1.0}};

// These will be the coordinates of the vertices of the cube
GLfloat CubeVertices[][3] = {{-1.0,-1.0,1.0},
       {-1.0,1.0,1.0}, {1.0,1.0,1.0}, {1.0,-1.0,1.0},
       {-1.0,-1.0,-1.0}, {-1.0,1.0,-1.0}, {1.0,1.0,-1.0},
       {1.0,-1.0,-1.0}};

GLfloat colors[][3] = {{1.0,1.0,0.0},{0.0,1.0,0.0},
               {1.0,0.0,0.0}, {1.0,0.5,0.0}, {0.9,0.9,0.9},
               {0.0,0.0,1.0}};
```

Queen Mary
University of London

Colorcube 1 (2)

```
void a3dpolygon(GLfloat vertices[][3], int a, int b, int c, int d) {
  /* draw a polygon/facet via list of vertices */
  glShadeModel(GL_FLAT);
  glBegin(GL_POLYGON);
  glVertex3fv(vertices[a]);
  glVertex3fv(vertices[b]);
  glVertex3fv(vertices[c]);
  glVertex3fv(vertices[d]);
  glEnd();
}

void colorcube()
{  /* map vertices to facets */
  glColor3fv(colors[0]);
  a3dpolygon(CubeVertices, 0,3,2,1);
  glColor3fv(colors[1]);
  a3dpolygon(CubeVertices, 2,3,7,6);
  glColor3fv(colors[2]);
  a3dpolygon(CubeVertices, 3,0,4,7);
  glColor3fv(colors[3]);
  a3dpolygon(CubeVertices, 1,2,6,5);
  glColor3fv(colors[4]);
  a3dpolygon(CubeVertices, 4,5,6,7);
  glColor3fv(colors[5]);
  a3dpolygon(CubeVertices, 5,4,0,1);
}
```



Queen Mary
University of London

Colorcube 1 (3)

```
void display()
{
  glClear(GL_COLOR_BUFFER_BIT);
  colorcube();
  glFlush();
}

void init()
{
  int i, j;
  glClearColor(1.0, 1.0, 1.0, 1.0);
    for (j = 0; j < 3; j++) {
    for (i = 0; i < 8; i++) {
      CubeVertices[i][j] = GlobalVertices[i][j] * X; /* Scale each vertex by X */
    }
  }
}

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

Queen Mary
University of London

Colorcube 2 (1)

```
GLfloat X = 0.5;          /* A scaling factor */

static GLfloat theta[] = {45.0,45.0,45.0}; //ROTATION ANGLES

GLfloat GlobalVertices[][3] = {{-1.0,-1.0,1.0},
       {-1.0,1.0,1.0}, {1.0,1.0,1.0}, {1.0,-1.0,1.0},
       {-1.0,-1.0,-1.0}, {-1.0,1.0,-1.0}, {1.0,1.0,-1.0},
       {1.0,-1.0,-1.0}};

// These will be the coordinates of the vertices of the cube
GLfloat CubeVertices[][3] = {{-1.0,-1.0,1.0},
       {-1.0,1.0,1.0}, {1.0,1.0,1.0}, {1.0,-1.0,1.0},
       {-1.0,-1.0,-1.0}, {-1.0,1.0,-1.0}, {1.0,1.0,-1.0},
       {1.0,-1.0,-1.0}};

GLfloat colors[][3] = {{1.0,1.0,0.0},{0.0,1.0,0.0},
              {1.0,0.0,0.0}, {1.0,0.5,0.0}, {0.9,0.9,0.9},
              {0.0,0.0,1.0}};
```

Queen Mary
University of London

Colorcube 2 (2)

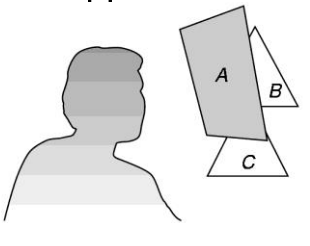
```
void display()
{

glClear(GL_COLOR_BUFFER_BIT);

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();

  glRotatef(theta[0], 1.0, 0.0, 0.0);
  glRotatef(theta[1], 0.0, 1.0, 0.0);
  glRotatef(theta[2], 0.0, 0.0, 1.0);

  colorcube();

  glFlush();
}
```

Queen Mary University of London

11

Hidden-Surface Removal

- We want to see only those surfaces in front of other surfaces
- OpenGL uses a *hidden-surface* method called the *z*-buffer algorithm that saves depth information as objects are rendered so that only the front objects appear in the image

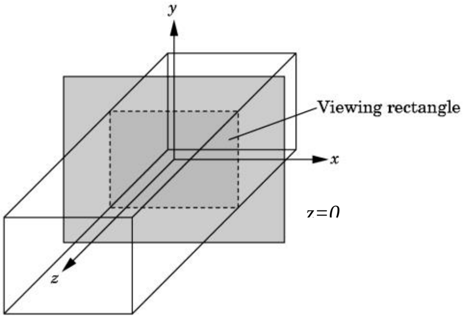


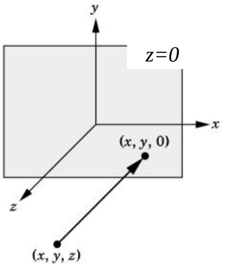
Queen Mary University of London

12

Orthographic Viewing

In the default orthographic view, points are projected forward along the z axis onto the plane $z=0$





13

Using the *z*-buffer algorithm

- The algorithm uses an extra buffer, the z-buffer, to store depth information as geometry travels down the pipeline
- It must be
  - Requested in **`main.c`**
    - **`glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH)`**
  - Enabled in **`init.c or main.c`**
    - **`glEnable(GL_DEPTH_TEST)`**
  - Cleared in the display callback
    - **`glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)`**

14

Colorcube 3 (1)

```
void display()
{

  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glRotatef(theta[0], 1.0, 0.0, 0.0);
  glRotatef(theta[1], 0.0, 1.0, 0.0);
  glRotatef(theta[2], 0.0, 0.0, 1.0);

  colorcube();

  glFlush();
}
```

15

Colorcube 3 (2)

```
void init()
{
  int i, j;
  glClearColor(1.0, 1.0, 1.0, 1.0);
    for (j = 0; j < 3; j++) {
    for (i = 0; i < 8; i++) {
      CubeVertices[i][j] = GlobalVertices[i][j]*X; //Scale each vertex by X
    }
  }
}

int main(int argc, char **argv)
{
  glutInit(&argc, argv);
    /* need both double buffering and z buffer */
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowSize(500, 500);
  glutCreateWindow("Colourcube");
  glutDisplayFunc(display);
  glEnable(GL_DEPTH_TEST); /* Enable hidden-surface removal */
  init();
  glutMainLoop();
}
```

16

Colorcube (1)

```
GLint axis = 2;

void spinCube()
{
/* idle callback, spin cube about selected axis */
 theta[axis] += 0.01;
 if( theta[axis] > 360.0 ) theta[axis] -= 360.0;
 glutPostRedisplay();
}

void specialkey(int key, int x, int y) {
  switch (key) {
   case GLUT_KEY_LEFT:
      axis = 0;
   break;
   case GLUT_KEY_UP:
      axis = 1;
   break;
  case GLUT_KEY_RIGHT:
     axis = 2;
   break;
  }
}
```

Queen Mary University of London

17

Colorcube (2)

```
void display()
{

  /* display callback, clear frame buffer and z buffer,
     rotate cube and draw, swap buffers */

  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glRotatef(theta[0], 1.0, 0.0, 0.0);
  glRotatef(theta[1], 0.0, 1.0, 0.0);
  glRotatef(theta[2], 0.0, 0.0, 1.0);
  colorcube();

  glutSwapBuffers();
}
```

Queen Mary University of London

18

Colorcube (3)

```
int main(int argc, char **argv)
{
  glutInit(&argc, argv);
    /* need both double buffering and z buffer */
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowSize(500, 500);
  glutCreateWindow("Colourcube");
  glutDisplayFunc(display);
  glutIdleFunc(spinCube);
  glutSpecialFunc(specialkey);
  glEnable(GL_DEPTH_TEST); /* Enable hidden-surface removal */
  init();
  glutMainLoop();
}
```

Queen Mary University of London

19

Glut Objects

GLUT objects:

```
glutWireCube(GLdouble size)
  centered at the origin, aligned with coordinate axes
glutWireSphere(GLdouble radius, GLint slices,
           GLint stacks) centered at the origin
glutWireCone(GLdouble base, GLdouble height,
  GLint slices, GLint stacks) base in plane z = 0
glutWireTorus(GLdouble inner, GLdouble outer,
   GLint sides, GLint slices) aligned with z axis
glutWireDodecahedron() vertices on sphere of radius 1
glutWireTeapot(GLdouble size)
```

Queen Mary University of London

20

The famous tea pot

Queen Mary
University of London

21

The famous tea pot

Queen Mary
University of London

22

The famous tea pot

Queen Mary
University of London

23

The famous tea pot

Queen Mary
University of London

24