LAB. 5

PROJECTIONS AND 3D VISUALIZATION

1. Learning goals
2. 3D transformations in OpenGL (revisited)
3. Example: The World Cube
4. Programming exercises

Lab. 5

PROJECTIONS AND 3D VISUALIZATION

In this lecture we are going to deal with 3D scenes and their visualization in 3D space. For this purpose, we use 3D geometric transformations to put 3D objects in the scene, projective transformations to project the scene on a virtual plane in the 3D space, and a viewer immersed in the scene that looks at the scene objects.
Thus, as usual in 2D engineering drafting, to draw an object on a paper sheet we need three entities:

- The viewer (you or me!) that looks at the object;
- The object (or a generic scene);
- The projection plane (paper sheet) where the object is projected.

1. Learning Goals

At the end of this chapter **you should be able to**:

1. To learn building 3D scenes up.
2. Master 3D transformations in computer graphics; in particular, you should be able to construct a scene together with objects moving around. These transformations are also used to move a bot or avatar in virtual environments such as, for example, in a First-Person Shooter (FPS) game.
3. Master the details behind the 3D viewing pipeline; in particular, you should be able to move the camera/viewer in the scene.

2. 3D Transformations in OpenGL (revisited)

Translation:

The **`glTranslated`** and **`glTranslatef`** functions multiply the current matrix by a translation matrix. Their prototypes are:

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

Rotation:

The **`glRotated`** and **`glRotatef`** functions multiply the current matrix by a rotation matrix. Their prototypes are:

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

The **`glRotate`** function computes a matrix that performs a counterclockwise rotation of `angle` degrees about the vector from the origin through the point (x, y, z).

Scaling:

The **glScaled** and **glScalef** functions multiply the current matrix by a scaling matrix. Their prototypes are:

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

All these transformations are essential in the modeling/construction of a scene because they allow us to construct an object using smaller building blocks and place it in the scene.

3. Example: The Cube World

The following program places 12 cubes on a large planar floor defined by a square with the following diagonal vertices: (-100,-100) and (100,100). This floor is in the plane XZ. Each cube has a different color.

```
#include <glut/glut.h>

void init(void);
void display(void);
void keyboard(unsigned char, int, int);
void resize(int, int);
void drawcube(int, int, int);

int is_depth; /* depth testing flag */

int main (int argc, char **argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowSize(600, 600);
   glutInitWindowPosition(40, 40);
   glutCreateWindow("The Cube World");
   init();
   glutDisplayFunc(display);
   glutKeyboardFunc(keyboard);

   /* this time we're going to keep the aspect ratio
   constant by trapping the window resizes */
   glutReshapeFunc(resize);

   glutMainLoop();
   return 0;
}

void init(void)
{
```

```
32    glClearColor(0.0, 0.0, 0.0, 0.0);
33    glEnable(GL_DEPTH_TEST);
34    is_depth = 1;
35    glMatrixMode(GL_MODELVIEW);
36 }
37
38 void display(void)
39 {
40    if (is_depth)
41       glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
42    else
43       glClear(GL_COLOR_BUFFER_BIT);
44    /* draw the floor */
45    glBegin(GL_QUADS);
46       glColor3f(0.2f, 0.2f, 0.2f);
47       glVertex3f(-100.0, 0.0, -100.0);
48       glColor3f(0.4f, 0.4f, 0.4f);
49       glVertex3f(-100.0, 0.0, 100.0);
50       glColor3f(0.6f, 0.6f, 0.6f);
51       glVertex3f(100.0, 0.0, 100.0);
52       glColor3f(0.8f, 0.8f, 0.8f);
53       glVertex3f(100.0, 0.0, -100.0);
54    glEnd();
55    /* draw 12 cubes with different colors */
56    drawcube(75, 57, 2);
57    drawcube(-65, -12, 3);
58    drawcube(50, -50, 1);
59    drawcube(-56, 17, 2);
60    drawcube(67, 12, 3);
61    drawcube(-87, 32, 1);
62    drawcube(-26, 75, 2);
63    drawcube(57, 82, 3);
64    drawcube(-3, 12, 1);
65    drawcube(46, 35, 2);
66    drawcube(37, -2, 3);
67    glutSwapBuffers();
68 }
69
70 void keyboard(unsigned char key, int x, int y)
71 {
72    /* This time the controls are:
73
74      "a": move left
75      "d": move right
76      "w": move forward
77      "s": move back
78      "t": toggle depth-testing
79
80    */
```

```
81    switch (key)
82    {
83       case 'a':
84       case 'A':
85                      glTranslatef(5.0, 0.0, 0.0);
86                      break;
87       case 'd':
88       case 'D':
89                      glTranslatef(-5.0, 0.0, 0.0);
90                      break;
91       case 'w':
92       case 'W':
93                      glTranslatef(0.0, 0.0, 5.0);
94                      break;
95       case 's':
96       case 'S':
97                      glTranslatef(0.0, 0.0, -5.0);
98                      break;
99       case 't':
100      case 'T':
101                     if (is_depth)
102                     {
103                        is_depth = 0;
104                        glDisable(GL_DEPTH_TEST);
105                     }
106                     else
107                     {
108                        is_depth = 1;
109                        glEnable(GL_DEPTH_TEST);
110                     }
111   }
112   display();
113 }
114
115 void resize(int width, int height)
116 {
117    if (height == 0) height = 1;
118
119    glMatrixMode(GL_PROJECTION);
120    glLoadIdentity();
121
122    /* we divide our width by our height to get the aspect ratio */
123    gluPerspective(45.0, width / height, 1.0, 400.0);
124
125    /* set initial position */
126    glTranslatef(0.0, -5.0, -150.0);
127
128    glMatrixMode(GL_MODELVIEW);
129
```

```
130     gluLookAt(0.0,5.0,10.0,0.0,0.0,0.0,0.0,1.0,0.0);
131
132  }
133
134  void drawcube(int x_offset, int z_offset, int color)
135  {
136     /* it draws a cube centerd at (x_offset, z_offset) x and z _big
137     are the back and rightmost points, x and z _small are
138     the front and leftmost points */
139     float x_big = (float)x_offset + 5;
140     float z_big = (float)z_offset + 5;
141     float x_small = (float)x_offset - 5;
142     float z_small = (float)z_offset - 5;
143     switch(color)
144     {
145     case 1:
146                       glColor3f(1.0,0.0,0.0);
147                       break;
148     case 2:
149                       glColor3f(0.0,1.0,0.0);
150                       break;
151     case 3:
152                       glColor3f(0.0,0.0,1.0);
153                       break;
154     }
155     glBegin(GL_QUADS);
156        glVertex3f(x_small,10.0,z_big); /* front */
157        glVertex3f(x_small,0.0,z_big);
158        glVertex3f(x_big,0.0,z_big);
159        glVertex3f(x_big,10.0,z_big);
160
161        glVertex3f(x_big,10.0,z_small); /* back */
162        glVertex3f(x_big,0.0,z_small);
163        glVertex3f(x_small,0.0,z_small);
164        glVertex3f(x_small,10.0,z_small);
165
166        glVertex3f(x_big,10.0,z_big); /* right */
167        glVertex3f(x_big,0.0,z_big);
168        glVertex3f(x_big,0.0,z_small);
169        glVertex3f(x_big,10.0,z_small);
170
171        glVertex3f(x_small,10.0,z_small); /* left */
172        glVertex3f(x_small,0.0,z_small);
173        glVertex3f(x_small,0.0,z_big);
174        glVertex3f(x_small,10.0,z_big);
175
176        glVertex3f(x_small,10.0,z_big); /* top */
177        glVertex3f(x_big,10.0,z_big);
178        glVertex3f(x_big,10.0,z_small);
```

```
179         glVertex3f(x_small,10.0,z_small);
180
181         glVertex3f(x_small,0.0,z_small); /* bottom */
182         glVertex3f(x_big,0.0,z_small);
183         glVertex3f(x_big,0.0,z_big);
184         glVertex3f(x_small,0.0,z_big);
185     glEnd();
186 }
```

Questions:

(1) Which are the objects of the scene?
(2) Which is the location of the viewer?
(3) Where is the projection plane?

4. Programming Exercises

1. Re-write the previous program to move a cube around the scene. The cube moves on the plane XZ. The allowed movements are: translation along x-axis; translation along z-axis; and rotation about the y-axis. Hint: use the keys x, y, and z for moving the cube interactively.
2. Re-write the previous program in a way that the moving object around in the scene is now the viewer. Use the arrow keys to move around the scene.
3. Build up a 3D house with a single door and no windows. Also, use with distinct color for each part of the house, namely: walls, roof, and door.
4. Enhance the previous program in order to rotate the house around the z-axis. The counterclockwise rotation is done using the mouse left button, while the clockwise rotation is done using the mouse right button.
5. Change the previous program in order to include a skyscraper (*arranha-céus*, in Portuguese) on the opposite side of the street. Do not use windows this time.
6. Change the previous program in order to include the door number on each building.