Tutorial 2

Coloured Cube

Ok, so far we have learned to open an OpenGL window and drawn a white triangle in it. In this tutorial we will learn how to use colours and draw 3D objects.

Here in this tutorial we will draw a coloured cube and its gradient version. We are using triangles and not quads to draw the cube, since quads and many other primitives were deprecated since OpenGL version 3.1.

First thing to start with, is to define the coordinates of the cube. Since we are using triangles to draw a cube, we need 12 triangles , 2 for each face and there are 6 faces for a cube. Each triangle is defined by 3 vertices, thus we need to define a total 36 vertices, 3 for each of 12 triangles. An array of float v_positions[] defines all the vertices, where each entry is x, y & z location of the corresponding vertex.

```
//6 faces, 2 triangles/face, 3 vertices/triangle
float v_positions[36][3] = {
//Face 1
//triangle 1
{-1.0,1.0,1.0},
{-1.0,-1.0,1.0},
{1.0,-1.0,1.0},
//triangle 2
{-1.0,1.0,1.0},
{1.0,-1.0,1.0},
{1.0,1.0,1.0},
...
};
```

Next we define the colour for each triangle using the v_colors[] array. I have used same colour for triangles on the same face, but you may try to give different colours. You may also give different colour to each vertex of a trian-

gle to get a gradient effect, which is demonstrated using drawGradientBox() function.

```
//6 faces, 2 triangles/face, RGB colors
float v_colors[12][3] = {
  //color for face 1
  {1.0, 0.0, 0.0},  //triangle 1
  {1.0, 0.0, 0.0},  //triangle 2
  //color for face 2
  {0.0, 1.0, 0.0},
  {0.0, 1.0, 0.0},
  ...
};
```

Rendering the coloured cube

The main() function is same as that of previous tutorial except that it call an extra init() function.

```
void init(void)
{
  // Use depth buffering for hidden surface elimination.
  glEnable(GL_DEPTH_TEST);

  //Other functions will be explained in later tutorials.
  ...
}
```

Here init() function is used to just initialize the view, we will learn more about modelling and viewing in upcoming tutorials. For now, we will only focus on depth and colouring.

We are enabling GL_DEPTH_TEST to do depth comparisons and update the depth buffer. This will automatically detect the hidden surfaces and will only draw visible faces of the cube. To enable the depth test capability of OpenGL we call glEnable() function.

void glEnable(GLenum cap);
Parameters :

- cap - a symbolic constant indicating a GL capability.

Moving on to our rendering function display(). The glClear() function resets the window colour and depth buffer to default value.

```
void display(void)
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  drawBox();
  //drawGradientBox();  //Uncomment this and comment call to drawBox()
  glutSwapBuffers();
}
```

```
void drawBox(void)
{
  for (int i=0;i<12;i++){
    glBegin(GL_TRIANGLES);
    glColor3fv(v_colors[i]);
    glVertex3fv(v_positions[3*i]);
    glVertex3fv(v_positions[3*i+1]);
    glVertex3fv(v_positions[3*i+2]);
    glEnd();
  }
}
```

drawBox() is the function which draws the coloured cube. It iterates through all the vertices of 12 triangles and applies the given colour to each triangle. The function glColor3fv() takes input as a pointer to a vector of three floats.

void glColor3fv(float* vecRGB)
Parameters :

- vecRGB - a vector(array) to 3 floats describing red, green and blue component of colour to be used.

void glVertex3fv(float* vecXYZ)
Parameters :

- vecXYZ a vector(array) to 3 floats describing x, y and z coordinates of a vertex.

You can also use different colour for each vertex of a triangle to get a gradient shade. Function drawGradientBox() demonstrates the same by applying different colour for each vertex of a triangle. Try to uncomment the drawGradientBox() and comment the drawBox() function to see the difference.

```
void display(void)
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  //drawBox();
  drawGradientBox();  //Uncommented
  glutSwapBuffers();
}
```

References

https://www.opengl.org/archives/resources/code/samples/glut_examples/examples/examples.html
https://www.opengl.org/sdk/docs/man2/xhtml/glEnable.xml