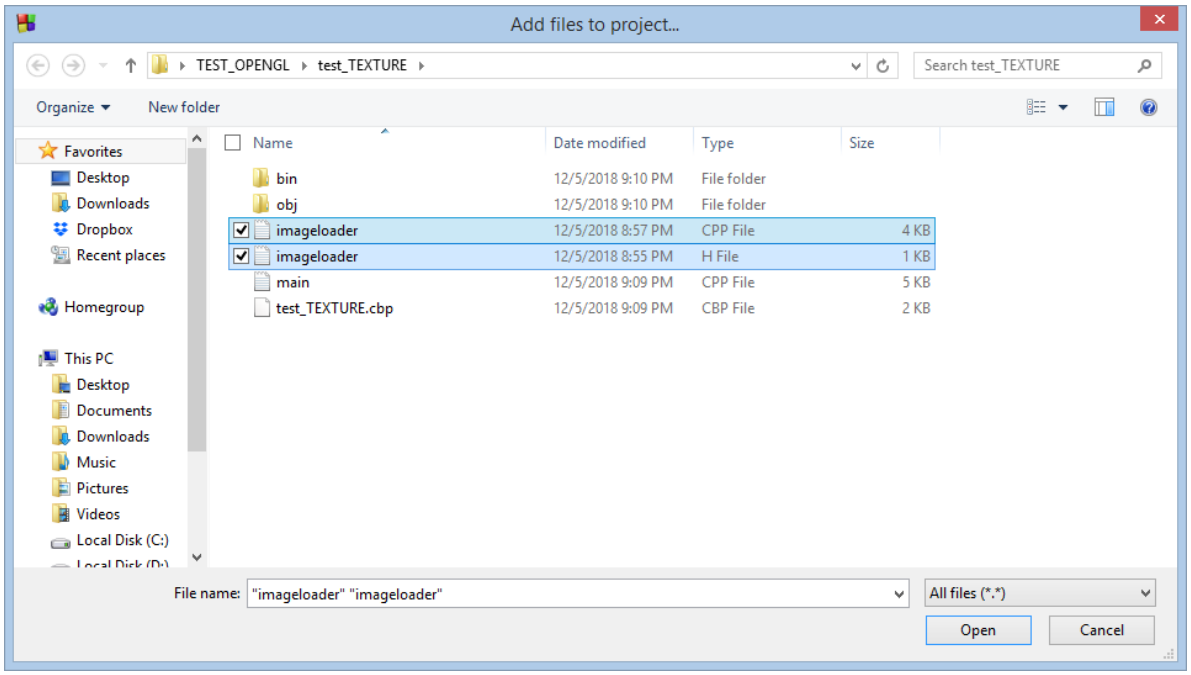
INSTRUCTIONS

This labsheet is divided into two parts as follows:

LAB 5 PART 1 : Simple Texture Mapping

1. Create a new GLUT project.
2. Download the library files (imageloader_library.zip) and image files (textures.zip) by clicking the respective URL provided in the website.
3. Copy two library files, "imageloader.cpp" and "imageloader.h", and paste them into folder where the GLUT project was created earlier in your computer.
4. Copy two image files, "crate.bmp" and "robot.bmp", and paste them into any specific location in your computer. For example, C:\zmisc\

6. Add the library files, “imageloader.cpp” and “imageloader.h” into the created GLUT project by:
    a. Right-clicking at the project name and select Add files…
    b. Selecting both files



7. Write down the following codes in the main.cpp file:

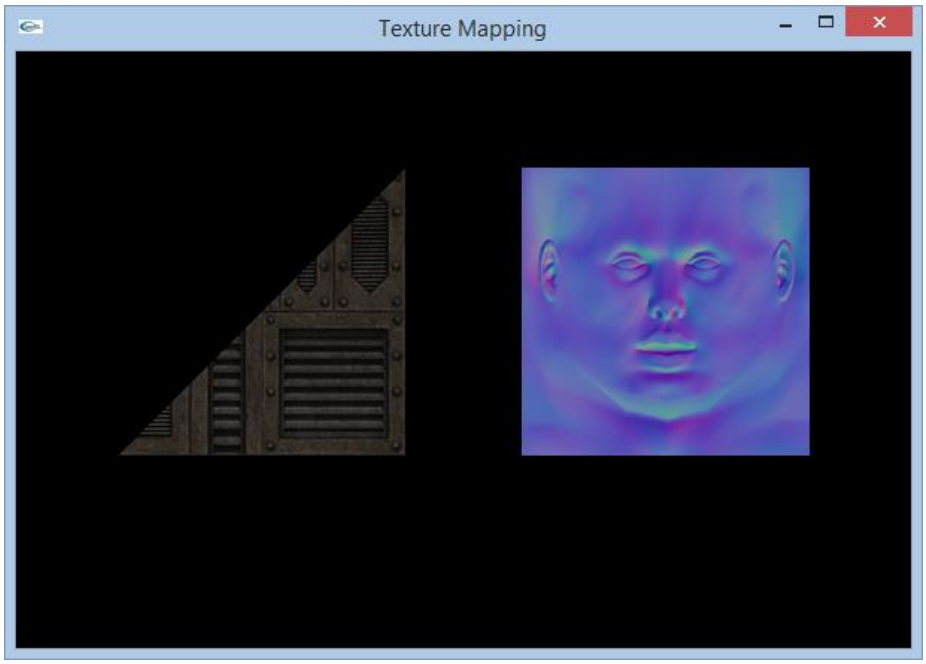
```
#include <windows.h>
#include <GL/glut.h>
#include "imageloader.h"

using namespace std;

void handleKeypress(unsigned char key, int x, int y)
{
    switch (key)
    {
        case 27: //Escape key
            exit(0);
    }
}

//Makes the image into a texture, and returns the id of the texture
GLuint loadTexture(Image* image)
{
    GLuint textureId;
    glGenTextures(1, &textureId); //Make room for our texture
    glBindTexture(GL_TEXTURE_2D, textureId); //Tell OpenGL which texture to edit
    //Map the image to the texture
    glTexImage2D(GL_TEXTURE_2D,                 //Always GL_TEXTURE_2D
                 0,                             //0 for now
                 GL_RGB,                        //Format OpenGL uses for image
                 image->width, image->height,   //Width and height
                 0,                             //The border of the image
                 GL_RGB, //GL_RGB, because pixels are stored in RGB format
                 GL_UNSIGNED_BYTE, //GL_UNSIGNED_BYTE, because pixels are stored
                                   //as unsigned numbers
                 image->pixels);                //The actual pixel data
    return textureId; //Returns the id of the texture
}

```

```
35  //The id of the texture
36  GLuint _textureId1;
37  GLuint _textureId2;
38
39  void initRendering()
40  {
41      glEnable(GL_DEPTH_TEST);
42      glEnable(GL_LIGHTING);
43      glEnable(GL_LIGHT0);
44      glEnable(GL_NORMALIZE);
45      glEnable(GL_COLOR_MATERIAL);
46
47      Image* image1 = loadBMP("C:\\zmisc\\texture-metalpanel.bmp");
48      _textureId1 = loadTexture(image1);
49      delete image1;
50
51      Image* image2 = loadBMP("C:\\zmisc\\texture-face.bmp");
52      _textureId2 = loadTexture(image2);
53      delete image2;
54  }
55
56  void handleResize(int w, int h)
57  {
58      glViewport(0, 0, w, h);
59      glMatrixMode(GL_PROJECTION);
60      glLoadIdentity();
61      gluPerspective(45.0, (float)w / (float)h, 1.0, 200.0);
62  }
63
64  void drawScene()
65  {
66      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
67
68      glMatrixMode(GL_MODELVIEW);
69      glLoadIdentity();
70
71      glTranslatef(0.0f, 1.0f, -6.0f);
72
73      GLfloat ambientLight[] = {0.2f, 0.2f, 0.2f, 1.0f};
74      glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLight);
75
76      GLfloat directedLight[] = {0.7f, 0.7f, 0.7f, 1.0f};
77      GLfloat directedLightPos[] = {-10.0f, 15.0f, 20.0f, 0.0f};
78      glLightfv(GL_LIGHT0, GL_DIFFUSE, directedLight);
79      glLightfv(GL_LIGHT0, GL_POSITION, directedLightPos);
80
```

```
81      //Draw a triangle with texture map
82      glPushMatrix();
83          glEnable(GL_TEXTURE_2D);
84          glBindTexture(GL_TEXTURE_2D, _textureId1);
85
86          glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
87          glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
88
89          glColor3f(1.0f, 1.0f, 1.0f);
90          glScaled(0.6, 0.6, 0.6);
91          glTranslated(-3.5, -1, 0);
92          glBegin(GL_TRIANGLES);
93              glNormal3f(0.0f, 0.0f, 1.0f);
94              glTexCoord2f(0.0f, 0.0f);   glVertex3f(-2.5f, -2.5f, -2.5f);
95              glTexCoord2f(1.0f, 1.0f);   glVertex3f(2.5f, 2.5f, -2.5f);
96              glTexCoord2f(1.0f, 0.0f);   glVertex3f(2.5f, -2.5f, -2.5f);
97          glEnd();
98      glPopMatrix();
99
100     //Draw a rectangle with texture map
101     glPushMatrix();
102         glEnable(GL_TEXTURE_2D);
103         glBindTexture(GL_TEXTURE_2D, _textureId2);
104
105         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
106         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
107
108         glColor3f(1.0f, 1.0f, 1.0f);
109         glScaled(0.6, 0.6, 0.6);
110         glTranslated(3.5,-1,0);
111         glBegin(GL_QUADS);
112             glNormal3f(0.0f, 0.0f, 1.0f);
113             glTexCoord2f(0.0f, 0.0f);   glVertex3f(-2.5f, -2.5f, -2.5f);
114             glTexCoord2f(0.0f, 1.0f);   glVertex3f(-2.5f, 2.5f, -2.5f);
115             glTexCoord2f(1.0f, 1.0f);   glVertex3f(2.5f, 2.5f, -2.5f);
116             glTexCoord2f(1.0f, 0.0f);   glVertex3f(2.5f, -2.5f, -2.5f);
117         glEnd();
118     glPopMatrix();
119
120     glutSwapBuffers();
121 }
122
123 int main(int argc, char** argv)
124 {
125     glutInit(&argc, argv);
126     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
127     glutInitWindowPosition(200, 100);
128     glutInitWindowSize(600, 400);
129
130     glutCreateWindow("Texture Mapping");
131     initRendering();
132
```

```
133      glutDisplayFunc(drawScene);
134      glutKeyboardFunc(handleKeypress);
135      glutReshapeFunc(handleResize);
136
137      glutMainLoop();
138      return 0;
139  }
```

8. Sample output:



LAB 5 PART 2 : Mapping Multiple Textures on Different Object Surfaces

Write an OpenGL program to display a 3D object with texture mapping applied on the surfaces. Please consider the following requirements:

1. Draw a box. Define a series of coordinates to represent the vertices.
2. Find six different images with resolution, 256 x 256. The image must be in BMP file format.
3. Wrap different image/texture onto each box's surface.
4. Create a box rotation around x-axis and y-axis separately.