COMSATS Institute of Information Technology Lahore

Terminal Examination Lab – Semester SP 21

Course Title:	Computer Graphics			Course Code:	CSD353	Credit Hours:	3(2,1)
Course Instructor/s:	Aamer Mehmood			Programme Name:	BS Computer Sciences		
Semester:	5th,7th	Batch:	SP17-BCS	Section:	A,B,C	Date:	
Time Allowed:	**3 Hours**			**Maximum Marks:**		**50**	
Student's Name:	**SCHAL HASNAIN**			Reg. No.	SP18-BCS-012		
.							

Important Instructions / Guidelines:

- All programs to be done using Visual Studio as the editor, notepad in worst case.
- You are allowed to access the offline version of documentation installed in the lab computers
- Anyone found using the internet will be disqualified immediately.
- Call your instructor whenever you finish a task so that it can be graded on the spot.

Question 1: Write a program to draw a 3D pyramid, sides with different colors. Centered at the origin. 15 marks

```
/*

 * OGL01Shape3D.cpp: 3D Shapes

 */

#include <windows.h>  // for MS Windows

#include <GL/glut.h>  // GLUT, include glu.h and gl.h



/* Global variables */

char title[] = "3D Shapes";



/* Initialize OpenGL Graphics */

void initGL() {

   glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Set background color to black and opaque

   glClearDepth(1.0f);                   // Set background depth to farthest
   glEnable(GL_DEPTH_TEST);   // Enable depth testing for z-culling

   glDepthFunc(GL_LEQUAL);    // Set the type of depth-test

   glShadeModel(GL_SMOOTH);   // Enable smooth shading

   glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);  // Nice perspective corrections

}



/* Handler for window-repaint event. Called back when the window first appears and

   whenever the window needs to be re-painted. */

void display() {

   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // Clear color and depth buffers

   glMatrixMode(GL_MODELVIEW);     // To operate on model-view matrix



   // Render a color-cube consisting of 6 quads with different colors

   glLoadIdentity();                 // Reset the model-view matrix

   glTranslatef(1.5f, 0.0f, -7.0f);  // Move right and into the screen



   glBegin(GL_QUADS);                // Begin drawing the color cube with 6 quads

      // Top face (y = 1.0f)

      // Define vertices in counter-clockwise (CCW) order with normal pointing out

      glColor3f(0.0f, 1.0f, 0.0f);     // Green

      glVertex3f( 1.0f, 1.0f, -1.0f);

      glVertex3f(-1.0f, 1.0f, -1.0f);
```

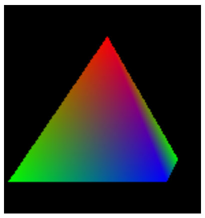
```
glVertex3f(-1.0f, 1.0f,  1.0f);

glVertex3f( 1.0f, 1.0f,  1.0f);



// Bottom face (y = -1.0f)

glColor3f(1.0f, 0.5f, 0.0f);     // Orange

glVertex3f( 1.0f, -1.0f,  1.0f);

glVertex3f(-1.0f, -1.0f,  1.0f);

glVertex3f(-1.0f, -1.0f, -1.0f);

glVertex3f( 1.0f, -1.0f, -1.0f);



// Front face  (z = 1.0f)

glColor3f(1.0f, 0.0f, 0.0f);     // Red

glVertex3f( 1.0f,  1.0f, 1.0f);

glVertex3f(-1.0f,  1.0f, 1.0f);

glVertex3f(-1.0f, -1.0f, 1.0f);

glVertex3f( 1.0f, -1.0f, 1.0f);



// Back face (z = -1.0f)

glColor3f(1.0f, 1.0f, 0.0f);     // Yellow

glVertex3f( 1.0f, -1.0f, -1.0f);

glVertex3f(-1.0f, -1.0f, -1.0f);

glVertex3f(-1.0f,  1.0f, -1.0f);
```

```
   glVertex3f( 1.0f,  1.0f, -1.0f);

   // Left face (x = -1.0f)
   glColor3f(0.0f, 0.0f, 1.0f);     // Blue
   glVertex3f(-1.0f,  1.0f,  1.0f);
   glVertex3f(-1.0f,  1.0f, -1.0f);
   glVertex3f(-1.0f, -1.0f, -1.0f);
   glVertex3f(-1.0f, -1.0f,  1.0f);

   // Right face (x = 1.0f)
   glColor3f(1.0f, 0.0f, 1.0f);     // Magenta
   glVertex3f(1.0f,  1.0f, -1.0f);
   glVertex3f(1.0f,  1.0f,  1.0f);
   glVertex3f(1.0f, -1.0f,  1.0f);
   glVertex3f(1.0f, -1.0f, -1.0f);
glEnd();  // End of drawing color-cube

// Render a pyramid consists of 4 triangles
glLoadIdentity();                  // Reset the model-view matrix
glTranslatef(-1.5f, 0.0f, -6.0f);  // Move left and into the screen

glBegin(GL_TRIANGLES);           // Begin drawing the pyramid with 4 triangles
```

```
// Front
glColor3f(1.0f, 0.0f, 0.0f);     // Red
glVertex3f( 0.0f, 1.0f, 0.0f);
glColor3f(0.0f, 1.0f, 0.0f);     // Green
glVertex3f(-1.0f, -1.0f, 1.0f);
glColor3f(0.0f, 0.0f, 1.0f);     // Blue
glVertex3f(1.0f, -1.0f, 1.0f);

// Right
glColor3f(1.0f, 0.0f, 0.0f);     // Red
glVertex3f(0.0f, 1.0f, 0.0f);
glColor3f(0.0f, 0.0f, 1.0f);     // Blue
glVertex3f(1.0f, -1.0f, 1.0f);
glColor3f(0.0f, 1.0f, 0.0f);     // Green
glVertex3f(1.0f, -1.0f, -1.0f);

// Back
glColor3f(1.0f, 0.0f, 0.0f);     // Red
glVertex3f(0.0f, 1.0f, 0.0f);
glColor3f(0.0f, 1.0f, 0.0f);     // Green
glVertex3f(1.0f, -1.0f, -1.0f);
glColor3f(0.0f, 0.0f, 1.0f);     // Blue
```

```
      glVertex3f(-1.0f, -1.0f, -1.0f);

      // Left
      glColor3f(1.0f,0.0f,0.0f);       // Red
      glVertex3f( 0.0f, 1.0f, 0.0f);
      glColor3f(0.0f,0.0f,1.0f);       // Blue
      glVertex3f(-1.0f,-1.0f,-1.0f);
      glColor3f(0.0f,1.0f,0.0f);       // Green
      glVertex3f(-1.0f,-1.0f, 1.0f);
   glEnd();   // Done drawing the pyramid

   glutSwapBuffers();  // Swap the front and back frame buffers (double buffering)
}

/* Handler for window re-size event. Called back when the window first appears and
   whenever the window is re-sized with its new width and height */
void reshape(GLsizei width, GLsizei height) {  // GLsizei for non-negative integer
   // Compute aspect ratio of the new window
   if (height == 0) height = 1;                // To prevent divide by 0
   GLfloat aspect = (GLfloat)width / (GLfloat)height;

   // Set the viewport to cover the new window
   glViewport(0, 0, width, height);


   // Set the aspect ratio of the clipping volume to match the viewport

   glMatrixMode(GL_PROJECTION);  // To operate on the Projection matrix

   glLoadIdentity();             // Reset

   // Enable perspective projection with fovy, aspect, zNear and zFar

   gluPerspective(45.0f, aspect, 0.1f, 100.0f);

}


/* Main function: GLUT runs as a console application starting at main() */

int main(int argc, char** argv) {

   glutInit(&argc, argv);            // Initialize GLUT

   glutInitDisplayMode(GLUT_DOUBLE); // Enable double buffered mode

   glutInitWindowSize(640, 480);   // Set the window's initial width & height

   glutInitWindowPosition(50, 50); // Position the window's initial top-left corner

   glutCreateWindow(title);          // Create window with the given title

   glutDisplayFunc(display);       // Register callback handler for window re-paint event

   glutReshapeFunc(reshape);       // Register callback handler for window re-size event

   initGL();                       // Our own OpenGL initialization

   glutMainLoop();                 // Enter the infinite event-processing loop

   return 0;

}
```

Output:



Question 2: Write a program to take House object and scale it half. Both houses should be shown but with different color. 15 marks

```
#include <windows.h>  // for MS Windows
#include <GL/glut.h>  // GLUT, include glu.h and gl.h

 // Initialize OpenGL Graphics
void initGL() {
        // Set "clearing" or background color
        glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Black and opaque
}

/* Handler for window-repaint event. Call back when the window first appears and
   whenever the window needs to be re-painted. */
void display() {
        glClear(GL_COLOR_BUFFER_BIT);   // Clear the color buffer with current clearing color

//1st House

        // Define shapes enclosed within a pair of glBegin and glEnd
        glBegin(GL_QUADS);           // Each set of 4 vertices form a quad
        glColor3f(1.0f, 0.0f, 0.0f);  // Red
        glVertex2f(-0.3f, -0.5f);     // Define vertices in counter-clockwise (CCW) order
        glVertex2f(0.3f, -0.5f);      // so that the normal (front-face) is facing you
        glVertex2f(0.3f, 0.1f);
        glVertex2f(-0.3f, 0.1f);
        glEnd();
        glBegin(GL_TRIANGLES);          // Each set of 3 vertices form a triangle
        glColor3f(0.0f, 0.0f, 1.0f);    // Blue
        glVertex2f(-0.4f, 0.1f);
        glVertex2f(0.4f, 0.1f);
        glVertex2f(0.0f, 0.5f);
        glEnd();

        glFlush();  // Render now

//2nd House

// Define shapes enclosed within a pair of glBegin and glEnd
        glBegin(GL_QUADS);              // Each set of 4 vertices form a quad
        glColor3f(0.0f, 0.0f, 1.0f);   // Blue
        glVertex2f(-0.3f, -0.5f);     // Define vertices in counter-clockwise (CCW) order
        glVertex2f(0.3f, -0.5f);      //  so that the normal (front-face) is facing you
        glVertex2f(0.3f, 0.1f);
        glVertex2f(-0.3f, 0.1f);
        glEnd();

        glBegin(GL_TRIANGLES);          // Each set of 3 vertices form a triangle

        glVertex2f(-0.4f, 0.1f);
        glColor3f(1.0f, 0.0f, 0.0f);  // Red
        glVertex2f(0.4f, 0.1f);
        glVertex2f(0.0f, 0.5f);
        glEnd();

        glFlush();  // Render now

}

glutSwapBuffers();  // Swap the front and back frame buffers (double buffering)

/* Main function: GLUT runs as a console application starting at main()  */
int main(int argc, char** argv) {
        glutInit(&argc, argv);                  // Initialize GLUT
        glutInitDisplayMode(GLUT_DOUBLE);                 // Enable double buffered mode
        glutCreateWindow("Vertex, Primitive & Color");  // Create window with the given title
        glutInitWindowSize(640, 480);                    // Set the window's initial width & height
        glutInitWindowPosition(50, 50);                  // Position the window's initial top-left corner
        glutDisplayFunc(display);                    // Register callback handler for window re-paint event
        initGL();                               // Our own OpenGL initialization
        glutReshapeFunc(reshape);                    // Register callback handler for window re-size event
        glutMainLoop();                              // Enter the event-processing loop
        return 0;
}
```

Question 3: Using the keyboard interaction, write a program which enable the user to draw circle, triangle, rectangle, polygon etc. just pressing the key.   20 marks

Hint: If user press the "T", triangle will be display. If "C" is pressed the circle will be drawn on the screen

```
// Include the GLEW header file

#include <GL/glut.h> // Include the GLUT header file

#include <stdio.h> // Include the GLUT header file

#include <math.h>


bool* keyStates = new bool[256]; // Create an array of boolean values of length 256 (0-255)


void initGL(void) {


  glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

  glMatrixMode(GL_PROJECTION);

  glLoadIdentity();

  gluOrtho2D(0, 400, 0, 400);

}
```

```
void keyOperations (void) {
if (keyStates['t']) { // If the 'a' key has been pressed


glClearColor(1.0f, 0.0f, 0.0f, 1.0f);
glClear(GL_COLOR_BUFFER_BIT);
glLoadIdentity();


glTranslatef(0.0f, 0.0f, -5.0f);
glColor3f(0.0f, 0.0f, 1.0f);


 glBegin(GL_TRIANGLES);
      glColor3f(1.0f, 0.0f, 0.0f);
      glVertex3f( 0.0f, 1.0f, 0.0f);
      glColor3f(0.0f, 1.0f, 0.0f);
      glVertex3f(-1.0f, -1.0f, 1.0f);
      glColor3f(0.0f, 0.0f, 1.0f);
      glVertex3f(1.0f, -1.0f, 1.0f);
   glEnd();
glFlush();


}
else if (keyStates['r']) {
glClearColor(1.0f, 0.0f, 1.0f, 1.0f);
glClear(GL_COLOR_BUFFER_BIT);
glLoadIdentity();
glTranslatef(0.0f, 0.0f, -5.0f);
glColor3f(0.0f, 0.0f, 1.0f);
```

```
    glBegin(GL_QUADS);

    glVertex3f(-2.0f, -1.0f, 0.0f);

    glVertex3f(-2.0f, 1.0f, 0.0f);

    glVertex3f(1.0f, 1.0f, 0.0f);

    glVertex3f(1.0f, -1.0f, 0.0f);

    glEnd();


glFlush();



}

else if (keyStates['c']) {

glClearColor(1.0f, 1.0f, 0.0f, 1.0f);

glClear(GL_COLOR_BUFFER_BIT);

   float theta;


   glColor3f(0.6f, 0.3f, 0.0f);

   glBegin(GL_POLYGON);

   for (int i = 0; i < 360; i ++) {

      theta = i * 3.142 / 180;

      glVertex2f(200 + 100 * cos(theta), 200 + 100 * sin(theta));

      glEnable(GL_BLEND);

   }

   glEnd();

   glFlush();


}
```

```
else if (keyStates['p']) {
glClearColor(1.0f, 0.0f, 0.0f, 1.0f);
glClear(GL_COLOR_BUFFER_BIT);
  float theta;
  glColor3f(0.6f, 0.3f, 0.0f);
  glBegin(GL_POLYGON);
  for (int i = 0; i < 360; i ++) {
    theta = i * 3.142 / 180;
    glVertex2f(200 + 100 * cos(theta), 200 + 100 * sin(theta));
    glEnable(GL_BLEND);
  }
  glEnd();
  glFlush();

}
}




void display (void) {
keyOperations();
}

void reshape (int width, int height) {
glViewport(0, 0, (GLsizei)width, (GLsizei)height); // Set our viewport to the size of our window
```

```
glMatrixMode(GL_PROJECTION); // Switch to the projection matrix so that we can manipulate how our scene is viewed

glLoadIdentity(); // Reset the projection matrix to the identity matrix so that we don't get any artifacts (cleaning up)

gluPerspective(60, (GLfloat)width / (GLfloat)height, 1.0, 100.0); // Set the Field of view angle (in degrees), the aspect ratio of our window, and the new and far planes

glMatrixMode(GL_MODELVIEW); // Switch back to the model view matrix, so that we can start drawing shapes correctly

}


void keyPressed (unsigned char key, int x, int y) {

keyStates[key] = true; // Set the state of the current key to pressed

}


void keyUp (unsigned char key, int x, int y) {

keyStates[key] = false; // Set the state of the current key to not pressed

}


int main (int argc, char **argv) {

glutInit(&argc, argv); // Initialize GLUT

glutInitDisplayMode (GLUT_SINGLE); // Set up a basic display buffer (only single buffered for now)

glutInitWindowSize (500, 500); // Set the width and height of the window

glutInitWindowPosition (100, 100); // Set the position of the window

glutCreateWindow ("Your first OpenGL Window"); // Set the title for the window


glutDisplayFunc(display); // Tell GLUT to use the method "display" for rendering


glutReshapeFunc(reshape); // Tell GLUT to use the method "reshape" for reshaping
```

```
glutKeyboardFunc(keyPressed);

glutKeyboardUpFunc(keyUp);


glutMainLoop();

}
```