Ahsanullah University of Science & Technology

Department of Computer Science and Engineering

CSE4204: Computer Graphics Lab

Student ID: 14.02.04.021

Student Name: Swapnil Biswas

Lab Group: A1

Date of Submission: 7/6/2018

1 Assignment 1

1.1 Task Name

Draw a scenario that contains basic openGL functions.

1.2 Description

It is a house in a greenland where there are some trees & river. There is a sun surrounded by some clouds in the sky.

1.3 Required Function

- GL_QUADS;
- glTranslatef();
- glColor3f();
- GL_TRIANGLES;
- GL_POLYGON

1.4 Code

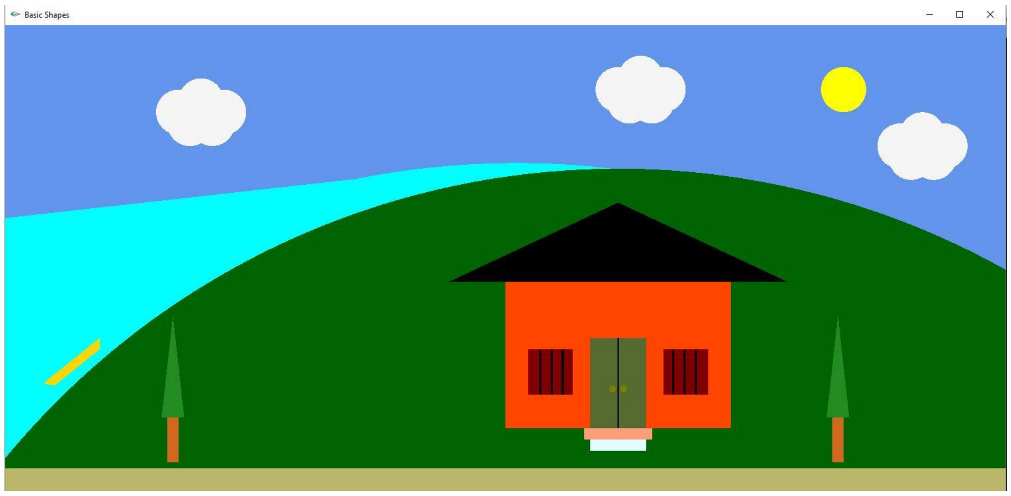
```
#include <bits/stdc++.h>
#include<windows.h>
#include <GL/glut.h>
using namespace std;

struct point{
    double x, y;
    point(double a, double b ){
        x = a; y = b;
    }
};
struct colour{
    double R, G, B;
    colour(double a, double b, double c ){
        R = a; G = b; B = c;
    }
};

//Called when a key is pressed
void handleKeypress(unsigned char key, //The key that was pressed
              int x, int y) {  //The current mouse coordinates
   switch (key) {
      case 27: //Escape key
         exit(0); //Exit the program
   }
}
//Initializes 3D rendering
void initRendering() {
   //Makes 3D drawing work when something is in front of something else
   glEnable(GL_DEPTH_TEST);
}
//Called when the window is resized
void handleResize(int w, int h) {
   //Tell OpenGL how to convert from coordinates to pixel values
   glViewport(0, 0, w, h);
```

```
  glMatrixMode(GL_PROJECTION); //Switch to setting the camera perspective
  //Set the camera perspective
  glLoadIdentity(); //Reset the camera
  gluPerspective(45.0,              //The camera angle
                 (double)w / (double)h, //The width-to-height ratio
                 1.0,                //The near z clipping coordinate
                 200.0);             //The far z clipping coordinate
}

void drawFilledCircle( double x, double y, double radius, double R, double G, double B ){
   float theta;
   glColor3f(R, G, B);
   glBegin(GL_POLYGON);
   for(int i = 0; i < 360; i++ )
   {
      theta = i*3.1416/180;
      glVertex3f(x+radius*cosf(theta),y+radius*sinf(theta),-5.0f);
   }
   glEnd();
}

void rectangle(point lowerLeft, point lowerRight, point upperRight, point upperLeft, colour c ){
   glBegin(GL_QUADS);

  glColor3f(c.R, c.G, c.B);
  glVertex3f(lowerLeft.x, lowerLeft.y, -5.0f);
  glVertex3f(lowerRight.x, lowerRight.y, -5.0f);
  glVertex3f(upperRight.x, upperRight.y, -5.0f);
  glVertex3f(upperLeft.x, upperLeft.y, -5.0f);

  glEnd(); //End quadrilateral coordinates
}

void triangle( point baseLeft, point baseRight, point up, colour c ){
   glBegin(GL_TRIANGLES);//Begin triangle coordinates

   glColor3f(c.R, c.G, c.B);
  glVertex3f(baseLeft.x, baseLeft.y, -5.0f);
  glVertex3f(baseRight.x, baseRight.y, -5.0f);
  glVertex3f(up.x, up.y, -5.0f);

  glEnd(); //End triangle coordinates
}

//Draws the 3D scene
void drawScene() {
  //Clear information from last draw
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glMatrixMode(GL_MODELVIEW); //Switch to the drawing perspective
  glLoadIdentity(); //Reset the drawing perspective

  glTranslatef(1.0f,0.0f,0.0f);

   drawFilledCircle(-.05, -1.15, .030, 0.502, 0.502, 0.000 ); // dorojar left handle
   drawFilledCircle(.05, -1.15, .030, 0.502, 0.502, 0.000 ); // dorojar right handle
   rectangle( point(-.005f, -1.50f), point(.005f, -1.5f), point(.005f, -.7f), point(-.005f, -.7f), colour(
       0.000f, 0.000f, 0.000f ) ); // dorojar majh
   rectangle( point(-.250f, -1.50f), point(.250f, -1.5f), point(.250f, -.7f), point(-.250f, -.7f), colour(
       0.333, 0.420, 0.184 ) ); // doroja
```

```
 rectangle( point(-.30f, -1.50f), point(.30f, -1.5f), point(.30f, -1.6f), point(-.30f, -1.6f), colour(
     1.000, 0.627, 0.478 ) ); // siri
 rectangle( point(-.250f, -1.60f), point(.25f, -1.6f), point(.25f, -1.7f), point(-.25f, -1.7f), colour(
     0.878, 1.000, 1.000 ) ); // siri

 rectangle( point(-.70f, -1.2f), point(-.68f, -1.2f), point(-.68f, -.8f), point(-.7f, -.8f), colour(
     0.000, 0.000, 0.000 ) ); // janala left grill
 rectangle( point(-.580f, -1.2f), point(-.60f, -1.2f), point(-.60f, -.8f), point(-.580f, -.8f), colour(
     0.000, 0.000, 0.000 ) ); // janala left grill
 rectangle( point(-.50f, -1.2f), point(-.48f, -1.2f), point(-.48f, -.8f), point(-.50f, -.8f), colour(
     0.000, 0.000, 0.000 ) ); // janala left grill
 rectangle( point(-.80f, -1.2f), point(-.40f, -1.2f), point(-.40f, -.8f), point(-.80f, -.8f), colour(
     0.502, 0.000, 0.000 ) ); // janala left

 rectangle( point(.68f, -.8f), point(.7f, -.8f),point(.70f, -1.2f), point(.68f, -1.2f), colour( 0.000,
     0.000, 0.000 ) ); // janala left grill
 rectangle( point(.60f, -.8f), point(.580f, -.8f), point(.580f, -1.2f), point(.60f, -1.2f), colour(
     0.000, 0.000, 0.000 ) ); // janala left grill
 rectangle( point(.48f, -.8f), point(.50f, -.8f), point(.50f, -1.2f), point(.48f, -1.2f), colour( 0.000,
     0.000, 0.000 ) ); // janala right grill
 rectangle( point(.40f, -.8f), point(.80f, -.8f), point(.80f, -1.2f), point(.40f, -1.2f), colour( 0.502,
     0.000, 0.000 ) ); // janala right

 rectangle( point(-1.0f, -1.5f), point(1.0f, -1.5f), point(1.0f, -.2f), point(-1.0f, -.2f), colour(
     1.000, 0.271, 0.000 ) ); // body

 triangle( point(-1.5, -.2), point( 1.5 ,-.2), point(0.0, .5), colour(0.000, 0.000, 0.000) );// roof

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

 triangle( point(-3.05, -1.4), point( -2.85 ,-1.4), point(-2.95, -.5), colour(0.133, 0.545, 0.133) ); //
     tree er matha
 rectangle( point(-3.0f, -1.8f), point(-2.90f, -1.8f), point(-2.90f, -1.0f), point(-3.0f, -1.0f), colour(
     0.824, 0.412, 0.118 ) );//tree er body

 triangle( point(3.05, -1.4), point( 2.85 ,-1.4), point(2.95, -.5), colour(0.133, 0.545, 0.133) ); //
     tree er matha
 rectangle( point(2.90f, -1.0f), point(3.0f, -1.0f), point(3.0f, -1.8f), point(2.90f, -1.8f), colour(
     0.824, 0.412, 0.118 ) );//tree er body

glTranslatef(.0f,-0.6f,0.0f);
 rectangle( point(-4.0f, -.52f), point(-3.6f, -.20f), point(-3.6f, -.10f), point(-4.1f, -.5f), colour(
     1.000, 0.843, 0.000 ) ); //nouka
 glTranslatef(.0f,0.5f,0.0f);

 rectangle( point(-5.0f, -5.0f), point(5.0f, -5.0f), point(5.0f, -1.75f), point(-5.0f, -1.75f), colour(
     0.741, 0.718, 0.420 ) ); //mati

 drawFilledCircle(3.5, 1.1, .20, 0.961, 0.961, 0.961 );//cloud
 drawFilledCircle(3.7, 1.2, .20, 0.961, 0.961, 0.961 );//cloud
 drawFilledCircle(3.9, 1.1, .20, 0.961, 0.961, 0.961 );//cloud
 drawFilledCircle(3.6, 1.0, .20, 0.961, 0.961, 0.961 );//cloud
 drawFilledCircle(3.8, 1.0, .20, 0.961, 0.961, 0.961 );//cloud

 drawFilledCircle(1.0, 1.6, .20, 0.961, 0.961, 0.961 );//cloud
 drawFilledCircle(1.2, 1.7, .20, 0.961, 0.961, 0.961 );//cloud
 drawFilledCircle(1.4, 1.6, .20, 0.961, 0.961, 0.961 );//cloud
 drawFilledCircle(1.1, 1.5, .20, 0.961, 0.961, 0.961 );//cloud
 drawFilledCircle(1.3, 1.5, .20, 0.961, 0.961, 0.961 );//cloud
```

```
    drawFilledCircle(-2.5, 1.4, .20, 0.961, 0.961, 0.961 );//cloud
    drawFilledCircle(-2.7, 1.5, .20, 0.961, 0.961, 0.961 );//cloud
    drawFilledCircle(-2.9, 1.4, .20, 0.961, 0.961, 0.961 );//cloud
    drawFilledCircle(-2.6, 1.3, .20, 0.961, 0.961, 0.961 );//cloud
    drawFilledCircle(-2.8, 1.3, .20, 0.961, 0.961, 0.961 );//cloud


    drawFilledCircle(3.0, 1.6, .20, 1.000, 1.000, 0.000 );//cloud//Sun

    drawFilledCircle(1.0, -6.15, 7.05, 0.000, 0.392, 0.000 ); // grass

    rectangle( point(-5.0f, -.8f), point(.90f, -.8f), point(-.50f, .90f), point(-5.0f, .4f), colour( 0.000,
        1.000, 1.000 ) ); // nodi
    drawFilledCircle(.1, -6.35, 7.3, 0.000, 1.000, 1.000 ); // nodi

    rectangle( point(-5.0, -5.0f), point(5.0f, -5.0f), point(5.0f, 5.0f), point(-5.0f, 5.0f), colour(0.392,
        0.584, 0.929 ) ); // akash

   glutSwapBuffers(); //Send the 3D scene to the screen
}

int main(int argc, char** argv) {
   //Initialize GLUT
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
   glutInitWindowSize(1500, 700); //Set the window size
   //Create the window
   glutCreateWindow("Basic Shapes");
   initRendering(); //Initialize rendering
   //Set handler functions for drawing, keypresses, and window resizes
   glutDisplayFunc(drawScene);
   glutKeyboardFunc(handleKeypress);
   glutReshapeFunc(handleResize);

   glutMainLoop(); //Start the main loop. glutMainLoop doesn't return.
   return 0; //This line is never reached
}
```

1.5 Output



2 Assignment 2

2.1 Task Name

Draw a rectangle object that can rotates towards X-axis, Y-axis, Z-axis & X-Y-Z axis.

2.2 Description

There are two rectangles. The background rectangle remains still but the four-ground rectangle is rotating towards axis's.

2.3 Required Function

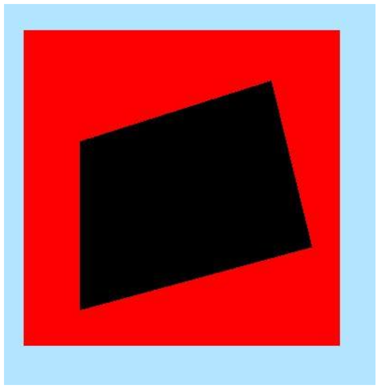
- GL_QUADS;
- glTranslatef();
- glColor3f();
- glutKeyboardFunc();
- glutPostRedisplay();
- glRotatef();
- glutTimerFunc

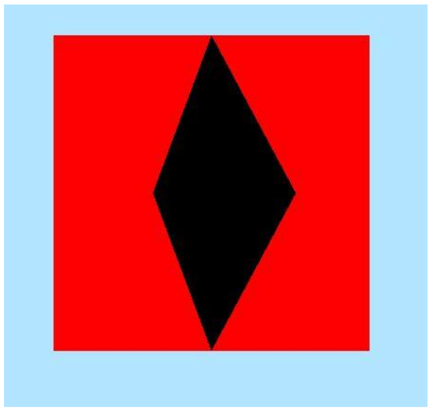
2.4 Code

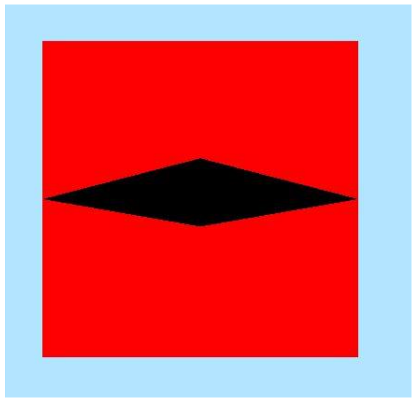
```
#include<windows.h>
#include <iostream>
#include <stdlib.h>
#ifdef __APPLE__
#include <OpenGL/OpenGL.h>
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif

using namespace std;

float xx = 1.0, yy = 0.0, zz = 0.0;

float _angle = 30.0f;
float _cameraAngle = 0.0f;

void inside( ){
    glRotatef(-_cameraAngle, 1.0f, 1.0f, 1.0f);
   glTranslatef(0.0f, 0.0f, -5.0f);

   glPushMatrix();
   glRotatef(_angle, xx, yy, zz);
   glBegin(GL_QUADS);
   //Trapezoid
   glColor3f(0.0f, 0.0f, 0.0f);
   glVertex3f(0.0f, 1.0f, 0.0f);
   glVertex3f(1.0f, 0.0f, 0.0f);
   glVertex3f(0.0f, -1.0f, 0.0f);
   glVertex3f(-1.0f, 0.0f, 0.0f);
   glEnd();
   glPopMatrix();
}

//Called when a key is pressed
void handleKeypress(unsigned char key, int x, int y) {
    cout << key << "\n";
   if( key == '1' ){
        xx = 1.0;
        yy = 0.0;
        zz = 0.0;
        _angle = 30.0f;
   }else if( key == '2' ){
       xx = 0.0;
       yy = 1.0;
       zz = 0.0;
       _angle = 30.0f;
   }else if( key == '3' ){
       xx = 0.0;
       yy = 0.0;
       zz = 1.0;
       _angle = 30.0f;
   }else if( key == '4' ){
       xx = 1.0;
       yy = 1.0;
       zz = 1.0;
       _angle = 30.0f;
   }

}

//Initializes 3D rendering
void initRendering() {
   glEnable(GL_DEPTH_TEST);
   glEnable(GL_COLOR_MATERIAL); //Enable color
   glClearColor(0.7f, 0.9f, 1.0f, 1.0f); //Change the background to sky blue
}

//Called when the window is resized
void handleResize(int w, int h) {
   glViewport(0, 0, w, h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(45.0, (double)w / (double)h, 1.0, 200.0);
}

//Draws the 3D scene
void drawScene() {
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();

    inside( );

   glPushMatrix();
   glBegin(GL_QUADS);
   glColor3f(1.0f, 0.0f, 0.0f);

   glVertex3f(-2.0f, -2.0f, -5.0f);
```

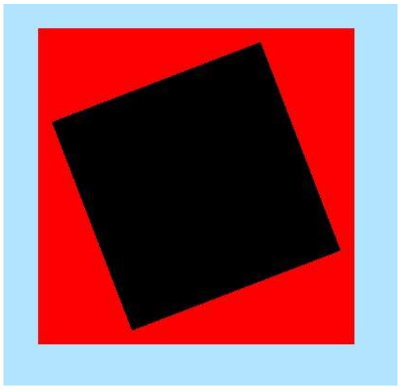
```
  glVertex3f(2.0f, -2.0f, -5.0f);
  glVertex3f(2.0f, 2.0f, -5.0f);
  glVertex3f(-2.0f, 2.0f, -5.0f);

  glEnd();
  glPopMatrix();




  glutSwapBuffers();
}

void update(int value) {
  _angle += 2.0f;
  if (_angle > 360) {
    _angle -= 360;
  }

  glutPostRedisplay();
  glutKeyboardFunc(handleKeypress);
  glutTimerFunc(25, update, 0);
}

int main(int argc, char** argv) {
  //Initialize GLUT
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowSize(1500, 700);

  //Create the window
  glutCreateWindow("Color");
  initRendering();

  //Set handler functions
  glutDisplayFunc(drawScene);
  glutKeyboardFunc(handleKeypress);
  glutReshapeFunc(handleResize);

  glutTimerFunc(25, update, 0); //Add a timer

  glutMainLoop();
  return 0;
}
```

2.5 Output









3 Assignment 3

3.1 Task Name

Draw any object where reflected lights falls on the side of it.

3.2 Description

It's a pyramid. Reflected lights fall on it from the right side.

3.3 Required Function

- GL_QUADS;
- glTranslatef();
- glColor3f();
- glutKeyboardFunc();
- glutPostRedisplay();
- glRotatef();
- glutTimerFunc()
- GL_TRIANGLES;
- glLightfv;
- glEnable(GL_DEPTH_TEST);
- glEnable(GL_COLOR_MATERIAL);
- glEnable(GL_LIGHTING);

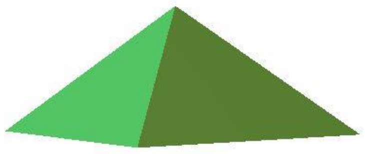
3.4 Code

```
#include <iostream>
#include <stdlib.h>
#include <windows.h>
#ifdef __APPLE__
#include <OpenGL/OpenGL.h>
#include <GLUT/glut.h>
```

```
#else
#include <GL/glut.h>
#endif

using namespace std;

//Called when a key is pressed
void handleKeypress(unsigned char key, int x, int y) {
   switch (key) {
      case 27: //Escape key
         exit(0);
   }
}
//Initializes 3D rendering
void initRendering() {
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
   glEnable(GL_DEPTH_TEST);
   glEnable(GL_COLOR_MATERIAL);
   glEnable(GL_LIGHTING); //Enable lighting
   //you can have upto 8 lighting
   glEnable(GL_LIGHT0); //Enable light #0
   glEnable(GL_LIGHT1); //Enable light #1
   glEnable(GL_NORMALIZE); //Automatically normalize normals
   //glShadeModel(GL_SMOOTH); //Enable smooth shading
}

//Called when the window is resized
void handleResize(int w, int h) {
   glViewport(0, 0, w, h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(45.0, (double)w / (double)h, 1.0, 200.0);
}

float _angle = -70.0f;

//Draws the 3D scene
void drawScene() {
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();

   glTranslatef(0.0f, -1.5f, -8.0f);

   //Add ambient light
   //sh that shines everywhere in our scene by the same amount
   //every face gets the same amount
   GLfloat ambientColor[] = {0.5f, 0.4f, 0.1f, 1.0f}; //Color (0.2, 0.2, 0.2) and intensity //can be greater
       than 1 so not like color
   glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientColor);

   //Add positioned light
   GLfloat lightColor0[] = {0.2f, 0.1f, 0.1f, 1.0f}; //Color (0.5, 0.5, 0.5)
   GLfloat lightPos0[] = {4.0f, 0.0f, 8.0f, 1.0f}; //Positioned at (4, 0, 8)
   glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColor0);
   glLightfv(GL_LIGHT0, GL_POSITION, lightPos0);

   //Add directed light
   GLfloat lightColor1[] = {0.2f, 0.5f, 0.4f, 1.0f}; //Color (0.5, 0.2, 0.2)
   //Coming from the direction (-1, 0.5, 0.5)
   // 0 because direced light source
   GLfloat lightPos1[] = {-1.0f, 0.5f, 0.5f, 0.0f};
   glLightfv(GL_LIGHT1, GL_DIFFUSE, lightColor1);
   glLightfv(GL_LIGHT1, GL_POSITION, lightPos1);

   glRotatef(_angle, 0.0f, 1.0f, 0.0f);
   glColor3f(0.5f, 1.0f, 1.0f);//cyan

   glBegin(GL_TRIANGLES);
    //front
   glNormal3f(-1.0f, 0.0f, 1.0f);
   glVertex3f(0.0f, 2.5f, 0.0f);
    glVertex3f(1.5f, 1.0f, 1.5f);
    glVertex3f(-1.5f, 1.0f, 1.5f);

    //right
    glNormal3f(1.0f, 0.0f, 0.0f);
    glVertex3f(0.0f, 2.5f, 0.0f);
    glVertex3f(1.5f, 1.0f, -1.5f);
    glVertex3f(1.5f, 1.0f, 1.5f);

    //Back
   glNormal3f(0.0f, 0.0f, -1.0f);
   glVertex3f(0.0f, 2.5f, 0.0f);
    glVertex3f(1.5f, 1.0f, -1.5f);
    glVertex3f(-1.5f, 1.0f, -1.5f);

   //Left
   glNormal3f(-1.0f, 0.0f, 0.0f);
    glVertex3f(0.0f, 2.5f, 0.0f);
    glVertex3f(-1.5f, 1.0f, -1.5f);
    glVertex3f(-1.5f, 1.0f, 1.5f);

    glEnd();
   glutSwapBuffers();
}

void update(int value) {
   _angle += 1.0f;
   if (_angle > 360) {
      _angle -= 360;
   }

   glutPostRedisplay();
   glutTimerFunc(25, update, 0);
}

int main(int argc, char** argv) {
   //Initialize GLUT
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
   glutInitWindowSize(600, 600);

   //Create the window
   glutCreateWindow("Lighting ");
   initRendering();

   //Set handler functions
   glutDisplayFunc(drawScene);
```

```
    glutKeyboardFunc(handleKeypress);
    glutReshapeFunc(handleResize);

    glutTimerFunc(25, update, 0); //Add a timer

    glutMainLoop();
    return 0;
}
```

3.5 Output



4 Online

4.1 Task Name

Draw an object that can move around on a particular area.

4.2 Description

The object is a rectangular shape and the are is a 'L' shape.

4.3 Required Function

- GL_QUADS;
- glTranslatef();
- glColor3f();
- glutKeyboardFunc();
- glutPostRedisplay();
- glutTimerFunc();

4.4 Code

```
#include<windows.h>
#include <iostream>
#include <stdlib.h>

#ifdef __APPLE__
#include <OpenGL/OpenGL.h>
```

```
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif
using namespace std;

float xx = 0.0, yy = 0.0, zz = 0.0;
float _angle = 30.0f;
float _cameraAngle = 0.0f;

bool okLeft( double x ){
    if( ( 1 + x ) < -1 ) return false;
    return true;
}
bool okDown( double x ){
    if( ( -.5 + x ) < -1 ) return false;
    return true;
}
bool okRight( double x ){
    if( ( 1.5 + x ) > 2 ) return false;
    if( yy > 0 && ( ( 1.5 + x ) > 0 ) ) return false;
    return true;
}
bool okUp( double x ){
    if( ( 0 + x ) > 2 ) return false;
    if( ( 1.5 + xx ) > 0 && ( x > 0 ) ) return false;
    return true;
}

//Called when a key is pressed
void handleKeypress(unsigned char key, int x, int y) {
    cout << key << "\n";
   if( key == '4' ){
        if( okLeft( xx - .5 ) ){
           xx -= .5;
        }
   }else if( key == '2' ){
       if( okDown( yy - .5 ) ){
            yy -= .5;
       }
   }else if( key == '6' ){
       if( okRight( xx + .5 ) ){
           xx += .5;
       }
   }else if( key == '8' ){
       if( okUp( yy + .5 ) ){
            yy += .5;
       }
   }
}
//Initializes 3D rendering
void initRendering() {
   glEnable(GL_DEPTH_TEST);
   glEnable(GL_COLOR_MATERIAL); //Enable color
   glClearColor(0.7f, 0.9f, 1.0f, 1.0f); //Change the background to sky blue
}
//Called when the window is resized
void handleResize(int w, int h) {
   glViewport(0, 0, w, h);
   glMatrixMode(GL_PROJECTION);
```

```
  glLoadIdentity();
  gluPerspective(45.0, (double)w / (double)h, 1.0, 200.0);
}

void object(){
   glTranslatef(xx, yy, 0.0f);

   glPushMatrix();
  glBegin(GL_QUADS);
  glColor3f(0.0f, 0.0f, 0.0f);

  glVertex3f(1.0f, 0.0f, 0.0f);
  glVertex3f(1.5f, 0.0f, 0.0f);
  glVertex3f(1.5f, -.5f, 0.0f);
  glVertex3f(1.0f, -.5f, 0.0f);

  glEnd();
  glPopMatrix();

  glTranslatef(-xx, -yy, 0.0f);
}

//Draws the 3D scene
void drawScene() {
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();

  glTranslatef(0.0f, 0.0f, -5.0f);
  object();

  glColor3f(1.0f, 1.0f, 1.0f);

  glPushMatrix();
  glBegin(GL_QUADS);

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

  glEnd();
  glPopMatrix();

  glPushMatrix();
  glBegin(GL_QUADS);

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

  glEnd();
  glPopMatrix();

  glutSwapBuffers();
}

void update(int value) {
  glutPostRedisplay();
```

```
   glutKeyboardFunc(handleKeypress);
   glutTimerFunc(25, update, 0);
}

int main(int argc, char** argv) {
   //Initialize GLUT
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
   glutInitWindowSize(1500, 700);

   //Create the window
   glutCreateWindow("Color");
   initRendering();

   //Set handler functions
   glutDisplayFunc(drawScene);
   glutKeyboardFunc(handleKeypress);
   glutReshapeFunc(handleResize);

   glutTimerFunc(25, update, 0); //Add a timer

   glutMainLoop();
   return 0;
}
```

4.5 Output