Ahsanullah University of Science and Technology

LAB REPORT

CSE 4204

Computer Graphics Lab

Date of Submission: 07.06.18

Submitted By

Ashraful Zaman Eashan

ID: 13.02.04.098, Section: A1

Assignment 1: Create a structure (such as a house, rocket, etc) which must contain the following functions: GL_QUADS, GL_TRIANGLES, GL_POLYGON, glTranslatef , glColor3f (1.0, 1.0, 0.0);

Source Code:

```
#include<windows.h>
#include<GL/glut.h>
void myinit(void)
{
    glClearColor(1.0,1.0,1.0,0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}

void hut(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    //
    glColor3f(1.0,1.0,0.0);
    glBegin(GL_QUADS);
    glVertex3f(0.2, 0.2,0.0);
    glVertex3f(0.8, 0.2,0.0);
    glVertex3f(0.6,0.4,0.0);
    glVertex3f(0.4,0.4,0.0);
    glEnd();


    glColor3f(1.0,0.4,0.2);
    glBegin(GL_POLYGON);
    glVertex3f (0.1, 0.1, 0.0);
    glVertex3f (0.4, 0.1, 0.0);
    glVertex3f (0.4, 0.5, 0.0);
    glVertex3f (0.1, 0.5, 0.0);
    glEnd();

    glColor3f(1.0,0.0,0.0);
    glBegin(GL_POLYGON);
    glVertex3f (0.10, 0.5, 0.0);
    glVertex3f (0.4, 0.5, 0.0);
    glVertex3f (0.25, 0.75, 0.0);
    glEnd();

    glColor3f(0.0,1.0,0.0);
    glBegin(GL_POLYGON);
    glVertex3f (0.4, 0.1, 0.0);
    glVertex3f (0.8, 0.4, 0.0);
    glVertex3f (0.8, 0.75, 0.0);
    glVertex3f (0.4, 0.5, 0.0);
    glEnd();

    glColor3f(0.0,0.0,1.0);
```

```
    glBegin(GL_POLYGON);
    glVertex3f (0.4, 0.5, 0.0);
    glVertex3f (0.8, 0.75, 0.0);
    glVertex3f (0.62, 0.93, 0.0);
    glVertex3f (0.25, 0.75, 0.0);
    glEnd();

    glColor3f(1.0,0.0,0.0);
    glBegin(GL_LINES);
    glVertex3f(0.1, 0.2, 0.0);
    glVertex3f(0.1, 0.75, 0.0);

    glEnd();

    glColor3f(1.0,0.0,0.0);
    glBegin(GL_TRIANGLES);
    glVertex3f(0.1, 0.75, 0.0);
    glVertex3f(0.3, 0.80, 0.0);
    glVertex3f(0.1, 0.80, 0.0);

    glEnd();
    glFlush();
}
int main(int argc,char** argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(50,100);
    glutInitWindowSize(640,480);
    glutCreateWindow("Polygon with viewport");
    myinit();
    glutDisplayFunc(hut);
    glutMainLoop();
}
```

Output:

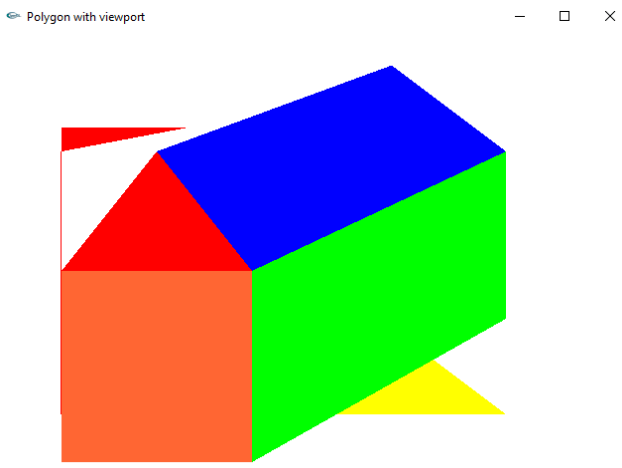

Fig 1: House GL_QUADS, GL_TRIANGLES, GL_POLYGON, glTranslatef , glColor3f

Assignment 2: Transform, Rotation, Controlling using Keyboard and mouse.

Description:
To transform the object vertically press 1, to transform the object horizontally press 2, and to rotate the object press 3, finally you can exit it by pressing 4.

Source Code:

```
#include<windows.h>
#include<stdio.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;

int angle = 0;
int flag = 0;

void update(int v)
{
    int b = 0;
    angle += 3;
    if(angle > 360)
        angle = 0;
        b++;
    glutPostRedisplay();
    glutTimerFunc(25, update, 0);
    int printf(b);
}


void handleKeypress(unsigned char key, //The key that was pressed
                            int x, int y) {    //The current mouse
coordinates
      switch (key) {
            case '4':
                  exit(0);
        case '1':
            flag = 1;
            break;
        case '2':
            flag = 2;
            break;
        case '3':
            flag = 3;
            break;
```

```
	}
}


void initRendering() {
	glEnable(GL_DEPTH_TEST);
}

void handleResize(int w, int h) {
	glViewport(0, 0, w, h);

	glMatrixMode(GL_PROJECTION);
	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 drawScene() {
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

	glMatrixMode(GL_MODELVIEW); //Switch to the drawing perspective
	glLoadIdentity(); //Reset the drawing perspective

    if(flag == 1) glRotated(angle, 1.0f, 0.0f, 0.0f);
    else if(flag == 2) glRotated(angle, 0.0f, 1.0f, 0.0f);
    else if(flag == 3) glRotated(angle, 0.0f, 0.0f, 1.0f);


    glBegin(GL_TRIANGLES);
    glColor3f(0.0f, 0.6f, 0.4f);
    glVertex3f(-0.5f, 0.5f, -5.0f); //x

    glVertex3f(0.5f, 0.5f, -5.0f); //z
    glVertex3f(0.0f, -1.0f, -5.0f);

    glEnd();

	glutSwapBuffers();
}

int main(int argc, char** argv) {
	//Initialize GLUT
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
	glutInitWindowSize(500, 500); //Set the window size

	//Create the window
	glutCreateWindow("Rotation");
```

```
    initRendering(); //Initialize rendering

    glutDisplayFunc(drawScene);
    glutKeyboardFunc(handleKeypress);
    glutReshapeFunc(handleResize);

  glutTimerFunc(25, update, 0);
    glutMainLoop(); //Start the main loop.  glutMainLoop doesn't return.
    return 0; //This line is never reached
}
```

Output:

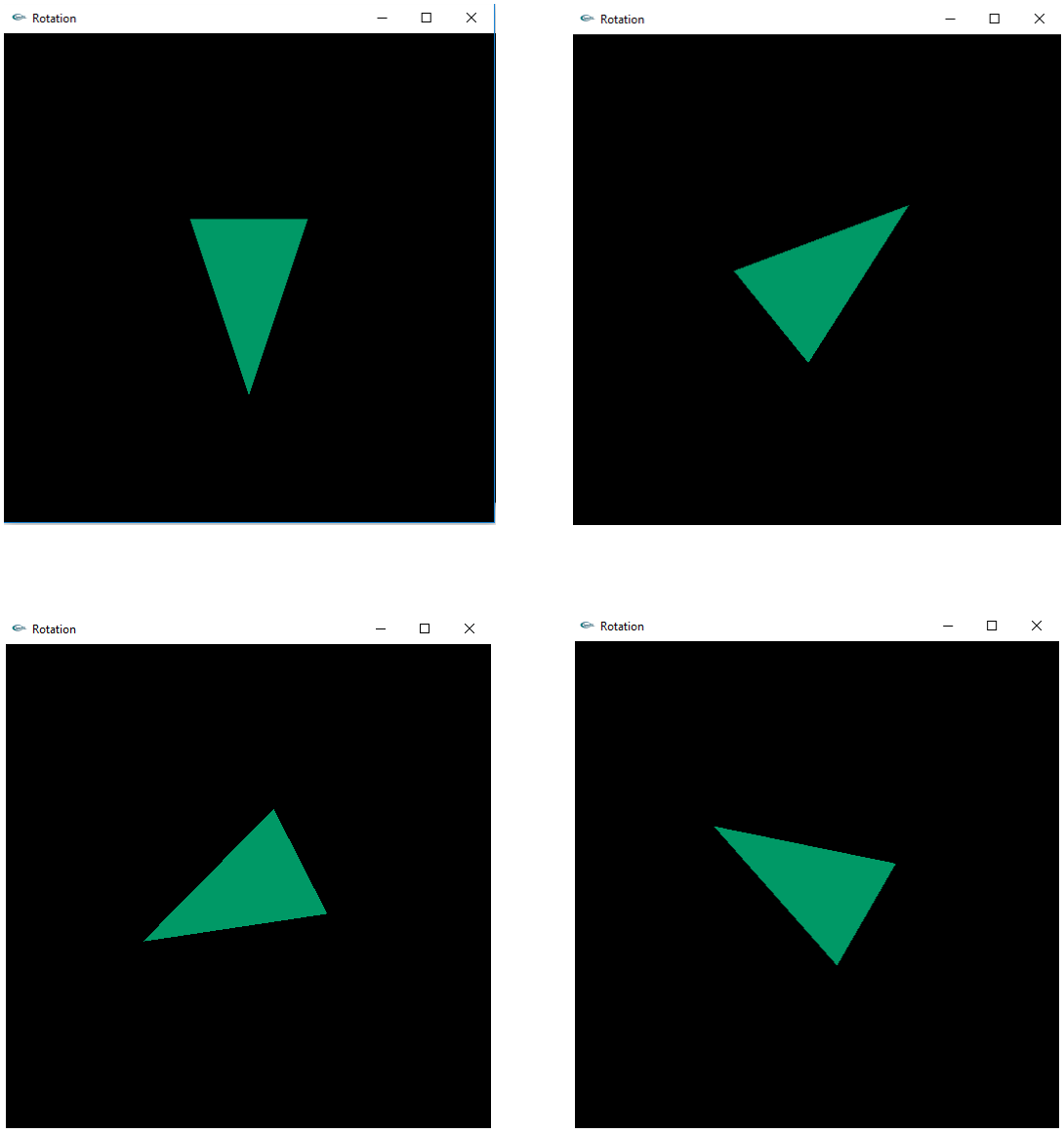

Fig 2: Rotation

Assignment 3: Create an object using lighting and shading.

Source 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() {
	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, 0.0f, -8.0f);

      //Add ambient light
      //sh that shines everywhere in our scene by the same amount
      //every face gets the same amount
      GLfloat ambientColor[] = {0.4f, 0.4f, 0.4f, 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[] = {1.0f, 0.0f, 0.0f, 1.0f}; //Color (0.5, 0.5,
0.5)
      GLfloat lightPos0[] = {2.0f, 0.0f, 4.0f, 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, 0.2f, 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.f, 0.4f, 0.0f};
      glLightfv(GL_LIGHT1, GL_DIFFUSE, lightColor1);
      glLightfv(GL_LIGHT1, GL_POSITION, lightPos1);

      glRotatef(_angle, 0.0f, 1.0f, 0.0f);
      glColor3f(1.0f, 1.0f, 1.0f);
      glBegin(GL_QUADS);

      //Front
      //normal is a vector perpendicular the face we are drawing
      //we need this because if the light source is directly opp to the
face then it will be light a lot
      //or if behind it won't be lit at all
      //they have to point outwards, so the back of the face don't get
light
      glNormal3f(0.0f, 0.0f, 0.5f);
      //glNormal3f(-1.0f, 0.0f, 1.0f);
      glVertex3f(-1.0f, -0.5f, 1.0f);
      //glNormal3f(1.0f, 0.0f, 1.0f);
      glVertex3f(1.0f, -0.5f, 1.0f);
      //glNormal3f(1.0f, 0.0f, 1.0f);
      glVertex3f(1.0f, 0.5f, 1.0f);
      //glNormal3f(-1.0f, 0.0f, 1.0f);
      glVertex3f(-1.0f, 0.5f, 1.0f);



      //Right
      glVertex3f(-1.0f,0.0f,0.0f);//left of window
      glVertex3f(0.0f,-1.0f,0.0f);//bottom of window
      glVertex3f(1.0f,0.0f,0.0f);//right of window
      glVertex3f(0.0f,1.0f,0.0f);//top of window
```

```
	//Back
	glNormal3f(0.0f, 0.0f, -0.5f);
	//glNormal3f(-1.0f, 0.0f, -1.0f);
	glVertex3f(-1.0f, -0.5f, -1.0f);
	//glNormal3f(-1.0f, 0.0f, -1.0f);
	glVertex3f(-1.0f, 0.5f, -1.0f);
	//glNormal3f(1.0f, 0.0f, -1.0f);
	glVertex3f(1.0f, 0.5f, -1.0f);
	//glNormal3f(1.0f, 0.0f, -1.0f);
	glVertex3f(1.0f, -0.5f, -1.0f);

	//Left
	glVertex3f(-1.0f,0.0f,0.0f);//left of window
	glVertex3f(0.0f,-1.0f,0.0f);//bottom of window
	glVertex3f(1.0f,0.0f,0.0f);//right of window
	glVertex3f(0.0f,1.0f,0.0f);//top of window

	glEnd();

	glutSwapBuffers();
}

void update(int value) {
	_angle += 1.5f;
	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(1000, 1000);

	//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;
}
```

Output:

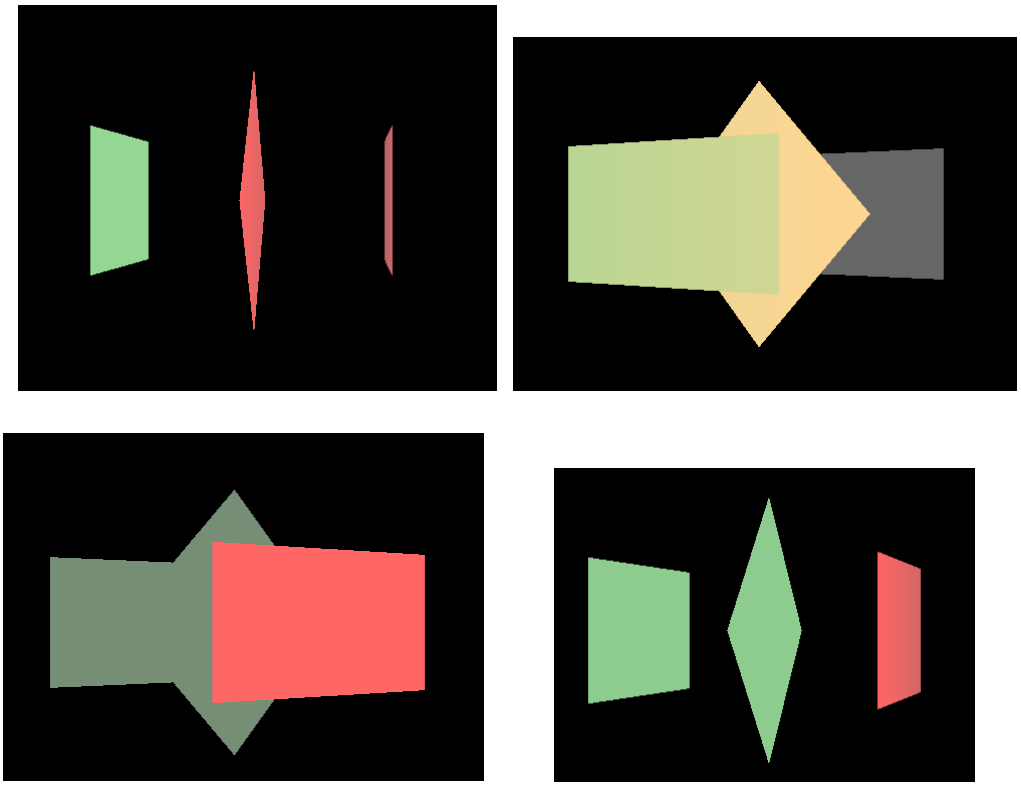

Fig 3: An object and using lighting and shading.