COMPUTER GRAPHICS ASSIGNMENT

Date:25/11/2020

Name- *FARHAN AHMAD*

Admission No.- *17JE003127*

Semester- *7th*

Submitted To- *Dr. Badam Singh Kushvah Sir*

Objective-

To make the following three objects using openGL Library.

1. Writing name using Bezier Curves
2. Drawing a 3D Clock Tower with different colors on each face
3. Drawing a Windmill with animated (rotating) wings

Methodology-

To accomplish the above mentioned objective we have to follow the following steps.

i. First we need to install an IDE (CodeBlocks) and afterwards we need to download the GLUT Library zip file and extract this in the same directory as CodeBlocks.

ii. After extracting the zip file we need to copy glut32.dll,glut.h and glut32.lib file in the required folder. After doing so our setup is completed and we are ready to go.

iii. Now we should write basic code which is common for all programs in openGL. This code includes initialization, display mode, window size, position and creation etc.

iv. To perform different tasks we need to follow different algorithms and write different pieces of code. Although there are many similarities in the code.

(a) **For writing names using bezier curves**: Draw the required Bezier curve. After doing so we should note down all the coordinates of the curves formed. Now by using the set of 4 points we will plot the Bazire curve in C++ language using openGL.

(b) **For drawing a 3D Clock Tower with different colors on each face**: To accomplish this task we need to use various inbuilt openGL functions Like glPushMatrix(), glRotatef(), glTranslatef(), glScalef(), glColor3f(), glutSolidOctahedron(),glPopMatrix() etc.We also declare user defined functions for drawing different types of objects like Tower, Stairs, Clock etc.

(c) **For drawing a Windmill with animated (rotating) wings**: To get this task done we will again be using various glut functions along with some of the functions used above. Some important functions used here are glMatrixMode(), glLoadIdentity(), glutTimerFunc() etc. Here also we make separate functions for different functionalities like Windmill and its frame speed etc.

Execution(Actual Code):

1. Writing name using Bezier Curves:

```
#include<windows.h>
#include<iostream>
#include<GL/glut.h>
#include<math.h>
using namespace std;

void myinit(void)
{
	glClearColor(0.0,0.0,0.0,1.0);
	glColor3f(0.5,0.75,1.0);
	glPointSize(3.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0,600,0.0,800);
}

void bezier(float b0_x,float b1_x,float b2_x,float b3_x,float b0_y,float
b1_y,float b2_y,float b3_y)
{
	float x,y,i;
	for(i=0.000;i<1.0;i+=0.001)
	{
	  x=b0_x*pow((1-i),3)+b1_x*3*i*pow((1-i),2)+b2_x*3*i*i*(1-i)+b3_x*i*i*i;
	  y=b0_y*pow((1-i),3)+b1_y*3*i*pow((1-i),2)+b2_y*3*i*i*(1-i)+b3_y*i*i*i;
	  glVertex2f(x,y);
	}
}


void display()
```

```
{
        glClear(GL_COLOR_BUFFER_BIT);
        glPointSize(3);
        glBegin(GL_POINTS);
        float m=5,m1=5;

       bezier(100+m*10.0, 100+m*14.26, 100+m*18.3, 100+m*20.0,
100+m1*19.5, 100+m1*19.5, 100+m1*19.5, 100+m1*19.5);
        bezier(100+m*15.0, 100+m*24.26, 100+m*48.3, 100+m*62.5,
100+m1*30.0, 100+m1*30.0, 100+m1*30.0, 100+m1*30.0);
        bezier(100+m*20.0, 100+m*20.0,  100+m*20.0, 100+m*20.0,
100+m1*0.00, 100+m1*16.8, 100+m1*20.8, 100+m1*30.0);
        bezier(100+m*60.0, 100+m*60.0,  100+m*60.0, 100+m*60.0,
100+m1*20.4, 100+m1*24.8, 100+m1*26.8, 100+m1*30.0);
        bezier(100+m*30.0, 100+m*30.0,  100+m*30.0, 100+m*30.0,
100+m1*30.0, 100+m1*28.8, 100+m1*26.8, 100+m1*24.6);
        bezier(100+m*47.7, 100+m*40.0,  100+m*36.0, 100+m*34.9,
100+m1*14.1, 100+m1*14.2, 100+m1*14.2, 100+m1*14.3);
        bezier(100+m*47.8, 100+m*47.8,  100+m*47.8, 100+m*47.8,
100+m1*29.8, 100+m1*28.8, 100+m1*16.8, 100+m1*0.00);

        bezier(100+m*1.30, 100+m*-2.4,  100+m*18.6, 100+m*9.8,
100+m1*22.0, 100+m1*33.5, 100+m1*33.4, 100+m1*19.7);
        bezier(100+m*-4.4, 100+m*4.6,   100+m*21.8, 100+m*9.7,
100+m1*10.4, 100+m1*-10.6,100+m1*12.8, 100+m1*19.6);
        bezier(100+m*39.9, 100+m*40.2,  100+m*39.85,100+m*34.9,
100+m1*30.3, 100+m1*13.2, 100+m1*-3.2, 100+m1*14.2);
        bezier(100+m*28.6, 100+m*14.2,  100+m*34.8,
100+m*32.85,100+m1*14.9, 100+m1*19.8, 100+m1*31.9, 100+m1*18.9);
        bezier(100+m*60.3, 100+m*47.0,  100+m*48.4, 100+m*60.0,
100+m1*20.1, 100+m1*22.8, 100+m1*-1.1, 100+m1*3.00);


        bezier(100+m*53.0, 100+m*56.1,  100+m*65.1, 100+m*60.0,
100+m1*-4.2, 100+m1*11.8, 100+m1*11.0, 100+m1*3.00);
```

```
      bezier(100+m*35.2, 100+m*32.2,
100+m*18.20,100+m*29.80,100+m1*6.8,  100+m1*25.0, 100+m1*8.2,
100+m1*2.00);


       glVertex2f(100+m*220,100+m1*440);
       glEnd();
       glFlush();

}

int main(int argc,char** argv)
{
       glutInit(&argc,argv);
       glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
       glutInitWindowSize(960,540);
       glutInitWindowPosition(0,0);
       glutCreateWindow("Farhan Ahmad 17JE003127");
       myinit();
       glutDisplayFunc(display);
       glutMainLoop();
       return EXIT_SUCCESS;
}
```

2. Drawing a 3D Clock Tower with different colors on each face

```
#include<windows.h>
#include <GL/glut.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#define XFORM_NONE    0
#define XFORM_ROTATE  1
#define XFORM_SCALE      2

struct tm *newtime;
time_t ltime;
float x_angle = 0.0, y_angle = 0.0, scale_size = 1;
int xform_mode = 0, ani = 0, timer = 75, release_x, release_y, press_x,
press_y;

void initLighting() {
        GLfloat lightColor0[] = { 0.6f, 0.6f, 0.6f, 1.0f };
        GLfloat lightPos0[] = { -0.5f, 0.8f, 1.0f, 0.0f };
        GLfloat ambientColor[] = { 0.4f, 0.4f, 0.4f, 1.0f };

        glEnable(GL_DEPTH_TEST);
        glEnable(GL_COLOR_MATERIAL);
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glEnable(GL_NORMALIZE);
        glShadeModel(GL_SMOOTH);

        glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientColor);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColor0);
        glLightfv(GL_LIGHT0, GL_POSITION, lightPos0);

}
```

```
//Tower
void Tower()
{
	// TOP
	glPushMatrix();
	glRotatef(45, 0, 1, 0);
	glTranslatef(0, 22.7, 0);
	glScalef(4.5, 4, 4.5);
	glColor3f(0.82, 0, 0.05);
	glutSolidOctahedron();
	glPopMatrix();

	//Tower
	glPushMatrix();
	glTranslatef(0, 11, 0);
	glScalef(5, 25, 5);
	glColor3f(0.65, 0.32, 0.05);
	glutSolidCube(1.0);
	glPopMatrix();

	//Cuts
	glPushMatrix();
	glColor3f(0.27, 0.13, 0);
	glTranslatef(0, 15, 0);
	glScaled(5.5, 0.5, 5.5);
	glutSolidCube(1);
	glPopMatrix();
	glPushMatrix();
	glColor3f(0.27, 0.13, 0);
	glTranslatef(0, 20, 0);
	glScaled(5.5, 0.5, 5.5);
	glutSolidCube(1);
	glPopMatrix();
	//Door
	glPushMatrix();
```

```
        glColor3f(0, 0, 0);
        glTranslatef(-0., 1.0, 2.5);
        glScalef(20, 24, 0.1);
        glutSolidCube(0.1);
        glPopMatrix();
        glPushMatrix();
        glColor3f(0, 0, 0);
        glTranslatef(-0., 1.8, 2.5);
        glScalef(10.1, 24, 0.01);
        glutSolidSphere(0.1, 222, 22);
        glPopMatrix();
}
//Stairs
void Stairs() {
        glPushMatrix();
                glColor3f(0.5, 0.5, 0.5);
        glPushMatrix();
                glTranslatef(0, -0.4, 0);
                glScaled(10, 0.5, 10);
                glutSolidCube(1);
        glPopMatrix();
        glPushMatrix();
                glTranslatef(0, -0.9, 0);
                glScaled(15, 0.5, 15);
                glutSolidCube(1);
        glPopMatrix();
        glPushMatrix();
                glTranslatef(0, -1.3, 0);
                glScaled(20, 0.5, 20);
                glutSolidCube(1);
        glPopMatrix();
        glPopMatrix();
}
//Land
/*
```

```
void Land() {

        glPushMatrix();
        glTranslatef(0, -1.5, 0);
        glScalef(500, 0.1, 500);
        glColor3f(0, 0.56, 0);
        glutSolidCube(1.0);
        glPopMatrix();
}
*/
//Clock
void Clock()
{
        glPushMatrix();
        glScaled(0.2, 0.2, 0.2);
        glTranslatef(0, 88, 13.55);
        glRotatef(180, 1.0, 0.0, 0.0);

        //Clock Frame
        glPushMatrix();
        glColor3f(0, 0, 0);
        glTranslatef(0.0, 0.0, 0.0);
        glScaled(1, 1, 0.01);
        glutSolidTorus(0.5, 7.5, 22, 55);
        glPopMatrix();

        //Clock Background
        glPushMatrix();
        glTranslatef(0, 0, 1.0);
        glColor3f(1.0, 1.0, 1.0);
        glScaled(7, 7, 0.01);
        glutSolidSphere(1, 22, 22);
        glPopMatrix();

        glPushMatrix();// Draw hour hand
        glColor3f(1.0, 0.5, 0.5);
        glRotatef((360 / 12) * newtime->tm_hour + (360 / 60) * (60 /
(newtime->tm_min + 1)), 0.0, 0.0, 1.0);
        glScaled(0.6, 4, 0.2);
        glTranslated(0, -0.45, 0);
        glutSolidCube(1);
        glPopMatrix();

        glPushMatrix();// Draw minute hand
        glColor3f(1.0, 0.5, 1.0);
        glRotatef((360 / 60) * newtime->tm_min, 0.0, 0.0, 1.0);
        glScaled(0.4, 5, 0.2);
        glTranslated(0, -0.45, 0);
        glutSolidCube(1);
        glPopMatrix();

        glPushMatrix();// Draw second hand
        glColor3f(1.0, 0.0, 0.5);
        glRotatef((360 / 60) *newtime->tm_sec, 0.0, 0.0, 1.0);
        glScaled(6, 0.2, 0.2);
        glTranslated(-0.45, 0, 0);
        glutSolidCube(1);
        glPopMatrix();


        for (int hour_ticks = 0; hour_ticks < 12; hour_ticks++)
        {
                glPushMatrix();// Draw next arm axis.
                glColor3f(0.0, 0, 0); // give it a color
                glTranslatef(0.0, 0.0, 0.0);
                glRotatef((360 / 12) * hour_ticks, 0.0, 0.0, 1.0);
                glTranslatef(6.0, 0.0, 0.0);
                glutSolidCube(1.0);

                glPopMatrix();
        }
        for (int sec_ticks = 0; sec_ticks < 60; sec_ticks++)
        {
                glPushMatrix();
                glTranslatef(0.0, 0.0, 0.0);
                glRotatef((360 / 60) * sec_ticks, 0.0, 0.0, 1.0);
                glTranslatef(6.0, 0.0, 0.0);
                glutSolidCube(0.25);
                glPopMatrix();
        }


        glPopMatrix();

}

void display()
{
        time(&ltime); // Get time
        newtime = localtime(&ltime); // Convert to local time
        glClearColor(0.4, 0.61, 0.97, 1);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(60, 1, 1, 900);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        gluLookAt(0.0, 15.0, 32.0, 0.0, 10.0, 0.0, 0.0, 30.0, 0.0);
        glRotatef(x_angle, 0, 1, 0);
        glRotatef(y_angle, 1, 0, 0);
        glScalef(scale_size, scale_size, scale_size);

        Tower();
        Clock();
        Stairs();
        //Land();
        glutSwapBuffers();
}


void mymouse(int button, int state, int x, int y)
{
        if (state == GLUT_DOWN) {
                press_x = x; press_y = y;
                if (button == GLUT_LEFT_BUTTON)
                        xform_mode = XFORM_ROTATE;
                else if (button == GLUT_RIGHT_BUTTON)
                        xform_mode = XFORM_SCALE;
        }
        else if (state == GLUT_UP) {
                xform_mode = XFORM_NONE;
        }
}

void myidle(int val) {
        int angle = 1;
        if (ani == 0) glutTimerFunc(timer, myidle, 0);

        angle = (angle + 10) % 360;
        glutPostRedisplay();

}

void mymotion(int x, int y)
{
        if (xform_mode == XFORM_ROTATE) {
                x_angle += (x - press_x) / 10.0;
                if (x_angle > 180) x_angle -= 360;
                else if (x_angle < -180) x_angle += 360;
                press_x = x;
```

```
		y_angle += (y - press_y) / 10.0;
		if (y_angle > 180) y_angle -= 360;
		else if (y_angle < -180) y_angle += 360;
		press_y = y;
	}
	else if (xform_mode == XFORM_SCALE) {
		float old_size = scale_size;
		scale_size *= (1 + (y - press_y) / 600.0);
		if (scale_size < 0) scale_size = old_size;
		press_y = y;
	}
	glutPostRedisplay();
}

void mykey(unsigned char key, int x, int y)
{
	if (key == '-') {
		scale_size--;
	}
	if (key == '+') {
		scale_size++;
	}
	if (key == 'd') {
		x_angle--;
	}
	if (key == 'a') {
		x_angle++;
	}
	if (key == 's') {
		y_angle--;
	}
	if (key == 'w') {
		y_angle++;
	}
```

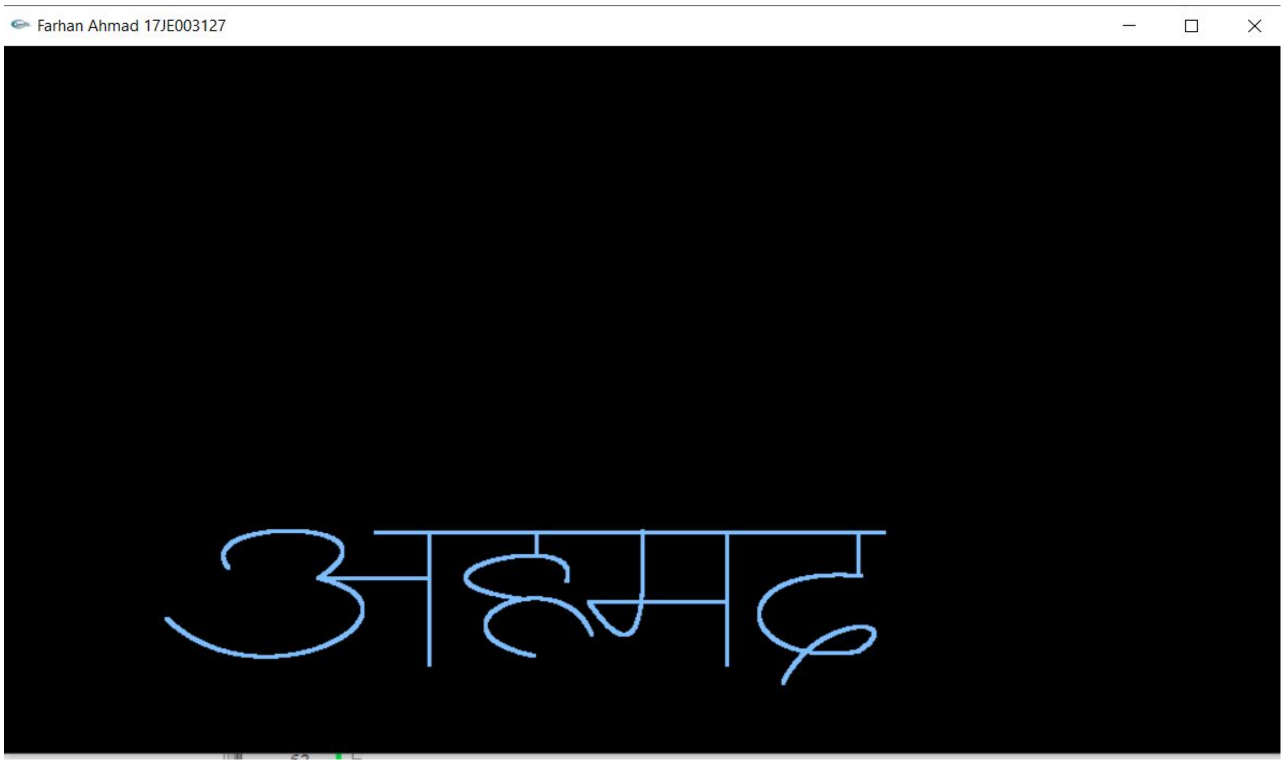
```
    if (key == ' ')
       {
           x_angle = 0.0, y_angle = 0.0, scale_size = 1 ;
       }
}

int main(int argc, char** argv)
{
         glutInit(&argc, argv);
         glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
         glutInitWindowSize(1400, 800);      //800 600
         glutCreateWindow("Clock Tower");
         glutDisplayFunc(display);
         glutMouseFunc(mymouse);
         glutMotionFunc(mymotion);
         initLighting();
         glutKeyboardFunc(mykey);
         glutTimerFunc(timer, myidle, 0);
         glutMainLoop();
}
```

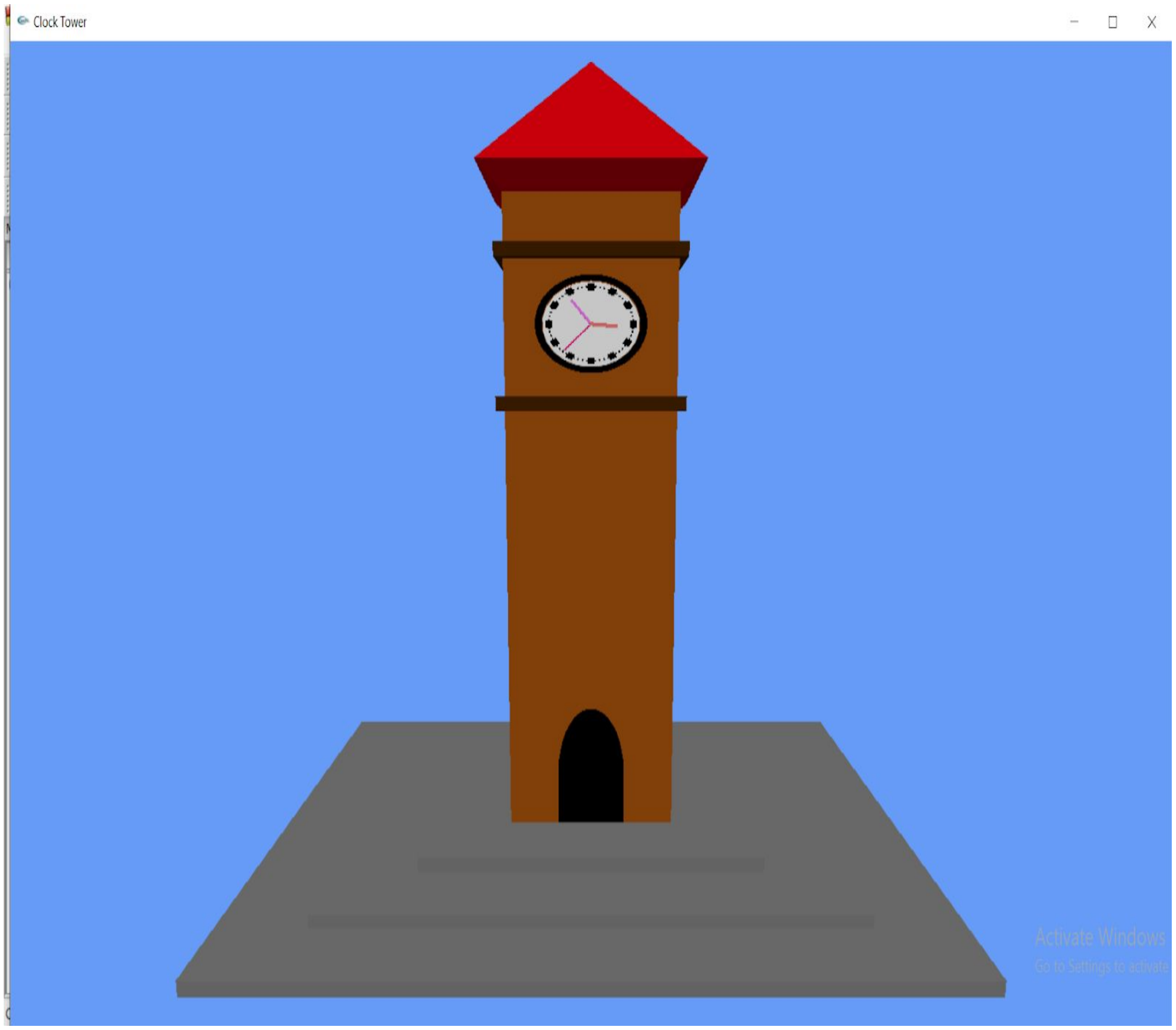
3.Drawing a Windmill with animated (rotating) wings

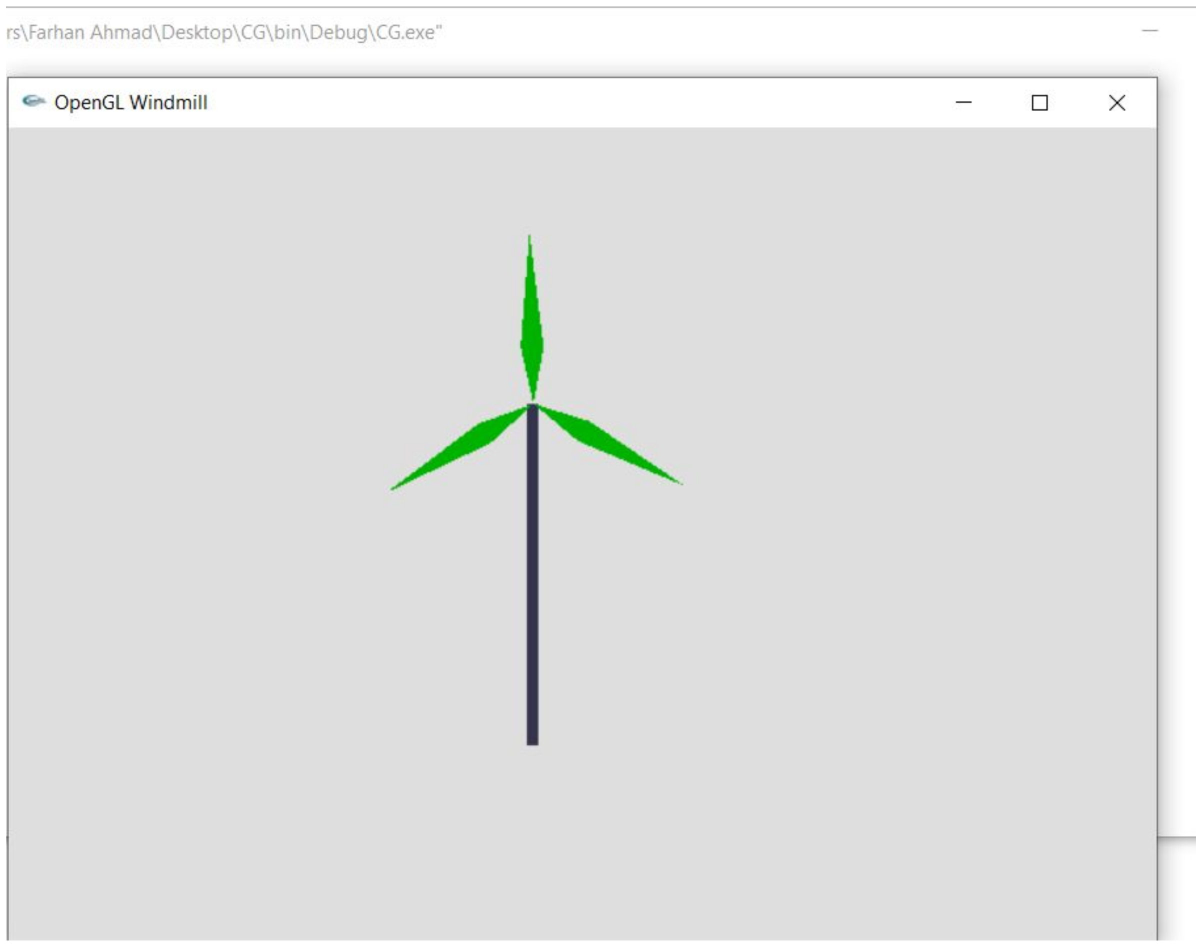
```
#include<windows.h>
#include <GL/gl.h>
#include <GL/glut.h>
#include <math.h>

const double PI = 3.141592654;

int frameNumber = 0;
```

```
void drawWindmill()
{
        int i;
        glColor3f(0.2f, 0.2f, 0.3f);      //0.8 0.8 .9
        glBegin(GL_POLYGON);
        glVertex2f(-0.05f, 0);
        glVertex2f(0.05f, 0);
        glVertex2f(0.05f, 3);
        glVertex2f(-0.05f, 3);
        glEnd();
        glTranslatef(0, 3, 0);
        glRotated(frameNumber * (180.0/46), 0, 0, 1);
        glColor3f(0.0f, 0.7f, 0.0f);
        for (i = 0; i < 3; i++)
       {
                glRotated(120, 0, 0, 1);  // Note: These rotations accumulate.
                glBegin(GL_POLYGON);
                glVertex2f(0,0);
                glVertex2f(0.5f, 0.1f);
                glVertex2f(1.5f,0);
                glVertex2f(0.5f, -0.1f);
                glEnd();
        }
}

void display()
{

        glClear(GL_COLOR_BUFFER_BIT); // Fills the scene with blue.
        glLoadIdentity();


        glPushMatrix();
        glTranslated(3.2,0.2,0);
        glScaled(0.7,0.7,1);
        drawWindmill();
        glPopMatrix();

        glutSwapBuffers();

}  // end display

void doFrame(int v)
 {
    frameNumber++;
    glutPostRedisplay();
    glutTimerFunc(30,doFrame,0);
}

void init() {
        glClearColor(0.87f, 0.87f, 0.87f, 1);
                // The next three lines set up the coordinates system.
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(0, 7, -1, 4, -1, 1);
        glMatrixMode(GL_MODELVIEW);
}


int main(int argc, char** argv)
 {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE);
    glutInitWindowSize(700,500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("OpenGL Windmill");

    init();
    glutDisplayFunc(display);
    glutTimerFunc(200,doFrame,0);
    glutMainLoop();
    return 0;
}
```

Results(Output):




Clock Tower
Activate Windows
Go to Settings to activate



Conclusion:

Thus we can make the above objects using openGL. We can easily modify the color, rotation, translation etc with slight changes in the code.