Chittagong University of Engineering & Technology

Assignment 8

Title: Draw a shape using Koch curve.

Name: Syed Tajir Hasnain

Id: 1604081

Session: 2020-21

Course code: 458

Course Title: Computer Graphics

Submitted to: Ms. Sabiha Anan,

Lecturer, Department of Computer Science & Engineering, CUET

Submission Date: October 05, 2021

GitHub Repository:

https://github.com/tajirhas9/opengl-practice/tree/main/assignment_8

Source Code:

- Files Structure
  - src/
    - main.cpp
      - Contains the main program
    - glib.h
      - Contains all the GLUT drawing utilities and algorithms.
      - Implements Korch Algorithm

`glib.h`

```
/**
* @author:                    Syed Tajir Hasnain
* @date:                      05/10/2021
* @project_details:           A GLUT utils header file
* @supported_operations:      1. initializes GLUT
*                             2. Koch Curve
*/

#include <GL/glut.h>
#include <cmath>
#include <stdio.h>
#include <iostream>
#include <vector>
#include "geometry.h"

namespace glib
{
	/**
	*
	* @utility: Initializes GLUT library
	* @params: takes the params supplied in the main() function
	*
	*/
	void init(int argc, char **argv)
	{
		glutInit(&argc, argv);
		glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
		glutInitWindowSize(500, 500);
		glutInitWindowPosition(100, 100);
		glutCreateWindow("");
		glClear(GL_COLOR_BUFFER_BIT);
		glClearColor(0, 0, 0, 0);
		glMatrixMode(GL_PROJECTION);
		glLoadIdentity();
		gluOrtho2D(-100, 100, -100, 100);
	}
	/**
	* @utility: takes the drawing callback and executes it
	*/
	void display(void (*callback)(void))
	{
		glutDisplayFunc(callback);
		glutMainLoop();
	}

	/**
```

```
        * @algorithm: Koch Curve Algorithm
        */

        void koch_curve(double x, double y, double lent, double alpha, int n, double
        &_x, double &_y )
        {
                if (n > 0)
                {
                        lent = lent / 3;
                        koch_curve(x, y, lent, alpha, n - 1, _x, _y);
                        x += lent * cos(alpha * M_PI / 180);
                        y += lent * sin(alpha * M_PI / 180);
                        koch_curve(x, y, lent, alpha - 60, n - 1, _x, _y);
                        x += lent * cos((alpha - 60) * M_PI / 180);
                        y += lent * sin((alpha - 60) * M_PI / 180);
                        koch_curve(x, y, lent, alpha + 60, n - 1, _x, _y);
                        x += lent * cos((alpha + 60) * M_PI / 180);
                        y += lent * sin((alpha + 60) * M_PI / 180);
                        koch_curve(x, y, lent, alpha, n - 1, _x, _y);
                }

                else
                {
                        glBegin(GL_LINES);
                        glVertex2f(x, y);
                        _x = x + (lent * cos(alpha * M_PI / 180));
                        _y = y + (lent * sin(alpha * M_PI / 180));
                        glVertex2f(x + (lent * cos(alpha * M_PI / 180)), y + (lent *
                        sin(alpha * M_PI / 180)));
                        glEnd();
                        glFlush();
                }
        }


        // make sure to flush everytime
        inline void close()
        {
                glFlush();
        }
}
```

`main.cpp`

```
#include <GL/glut.h>
#include <stdlib.h>
#include <stdio.h>
#include <bits/stdc++.h>
#include "glib.h"
using namespace std;

void input(double &x, double &y, double &alpha, double &length)
{
	printf("Enter starting co-ordinate: ");
	scanf("%lf %lf", &x, &y);
	printf("Enter rotation and length : ");
	scanf("%lf %lf", &alpha, &length);
}

void drawShape(void)
{
	double x, y, alpha, length;
	double _x, _y;
	int cnt = 0;

	input(x, y, alpha, length);

	_x = x;
	_y = y;

	while (cnt != 3)
	{
		glib::koch_curve(_x, _y, length, alpha, 1, _x, _y);
		alpha += 120;
		cnt++;
	}
}

int main(int argc, char **argv)
{
	glib::init(argc, argv);
	glib::display(drawShape);

	return 0;
}
```

Sample input:

Enter starting co-ordinate: 0 0

Enter rotation and length : 60 50

Sample Output: