**Practical 7**

Generate fractal patterns using i) Bezier ii) Koch Curve

**Program Code:-**

**Code for Koch Curve -**

#include <GL/glut.h>

#include <math.h>

GLfloat oldx=-0.7,oldy=0.5;

void drawkoch(GLfloat dir,GLfloat len,GLint iter) {

GLdouble dirRad = 0.0174533 \* dir;

GLfloat newX = oldx + len \* cos(dirRad);

GLfloat newY = oldy + len \* sin(dirRad);

if (iter==0) {

glVertex2f(oldx, oldy);

glVertex2f(newX, newY);

oldx = newX;

oldy = newY;

}

else {

iter--;

//draw the four parts of the side \_/\\_

drawkoch(dir, len, iter);

dir += 60.0;

drawkoch(dir, len, iter);

dir -= 120.0;

drawkoch(dir, len, iter);

dir += 60.0;

drawkoch(dir, len, iter);

}

}

void display(){

glClear( GL\_COLOR\_BUFFER\_BIT );

glBegin(GL\_LINES);

glColor3f(0.0, 1.0, 0.0);

drawkoch(0.0,0.04,3);

drawkoch(-120.0, 0.04, 3);

drawkoch(120.0,0.04,3);

glEnd();

glFlush();

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("Koch Curve");

glutDisplayFunc(display);

glutMainLoop();

}

**Code for Bezier Curve -**

#include <iostream>

#include <stdlib.h>

#include <GL/glut.h>

#include <math.h>

using namespace std;

//Point class for taking the points

class Point {

public:

float x, y;

void setxy(float x2, float y2)

{

x = x2; y = y2;

}

//operator overloading for '=' sign

const Point & operator=(const Point &rPoint)

{

x = rPoint.x;

y = rPoint.y;

return \*this;

}

};

int factorial(int n)

{

if (n<=1)

return(1);

else

n=n\*factorial(n-1);

return n;

}

float binomial\_coff(float n,float k)

{

float ans;

ans = factorial(n) / (factorial(k)\*factorial(n-k));

return ans;

}

Point abc[20];

int SCREEN\_HEIGHT = 500;

int points = 0;

int clicks = 4;

void myInit() {

glClearColor(1.0,1.0,1.0,0.0);

glColor3f(0.0,0.0,0.0);

glPointSize(3);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0,640.0,0.0,500.0);

}

void drawDot(int x, int y) {

glBegin(GL\_POINTS);

glVertex2i(x,y);

glEnd();

glFlush();

}

void drawLine(Point p1, Point p2) {

glBegin(GL\_LINES);

glVertex2f(p1.x, p1.y);

glVertex2f(p2.x, p2.y);

glEnd();

glFlush();

}

//Calculate the bezier point

Point drawBezier(Point PT[], double t) {

Point P;

P.x = pow((1 - t), 3) \* PT[0].x + 3 \* t \* pow((1 -t), 2) \* PT[1].x + 3 \* (1-t) \* pow(t, 2)\* PT[2].x + pow (t, 3)\* PT[3].x;

P.y = pow((1 - t), 3) \* PT[0].y + 3 \* t \* pow((1 -t), 2) \* PT[1].y + 3 \* (1-t) \* pow(t, 2)\* PT[2].y + pow (t, 3)\* PT[3].y;

return P;

}

//Calculate the bezier point [generalized]

Point drawBezierGeneralized(Point PT[], double t) {

Point P;

P.x = 0; P.y = 0;

for (int i = 0; i<clicks; i++)

{

P.x = P.x + binomial\_coff((float)(clicks - 1), (float)i) \* pow(t, (double)i) \* pow((1 - t), (clicks - 1 - i)) \* PT[i].x;

P.y = P.y + binomial\_coff((float)(clicks - 1), (float)i) \* pow(t, (double)i) \* pow((1 - t), (clicks - 1 - i)) \* PT[i].y;

}

//cout<<P.x<<endl<<P.y;

//cout<<endl<<endl;

return P;

}

void myMouse(int button, int state, int x, int y) {

// If left button was clicked

if(button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN) {

// Store where mouse was clicked, Y is backwards.

abc[points].setxy((float)x,(float)(SCREEN\_HEIGHT - y));

points++;

// Draw the red dot.

drawDot(x, SCREEN\_HEIGHT - y);

// If (click-amout) points are drawn do the curve.

if(points == clicks)

{

glColor3f(0.2,1.0,0.0);

// Drawing the control lines

for(int k=0;k<clicks-1;k++)

drawLine(abc[k], abc[k+1]);

Point p1 = abc[0];

/\* Draw each segment of the curve.Make t increment in smaller amounts for a more detailed curve.\*/

for(double t = 0.0;t <= 1.0; t += 0.02)

{

Point p2 = drawBezierGeneralized(abc,t);

cout<<p1.x<<" , "<<p1.y<<endl;

cout<<p2.x<<" , "<<p2.y<<endl;

cout<<endl;

drawLine(p1, p2);

p1 = p2;

}

glColor3f(0.0,0.0,0.0);

points = 0;

}

}

}

void myDisplay() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glFlush();

}

int main(int argc, char \*argv[]) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(640,500);

glutInitWindowPosition(100,150);

glutCreateWindow("Bezier Curve");

glutMouseFunc(myMouse);

glutDisplayFunc(myDisplay);

myInit();

glutMainLoop();

return 0;

}