Experiment No 7

Title Generate fractal patterns using i) Bezier Koch Curve

CODE FOR Bezier Curves

#include <iostream>

#include <math.h>

#include <time.h>

#include <GL/glut.h>

using namespace std;

int x[4],y[4];

void init(){

glClearColor(1.0,1.0,1.0,0.0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0,640,0,480);

glClear(GL\_COLOR\_BUFFER\_BIT);

}

void putpixel(double xt,double yt )

{

glColor3f(1,0,0);

glBegin(GL\_POINTS);

glVertex2d(xt,yt);

glEnd();

glFlush();

}

void Algorithm(){

glColor3f(0,1,0);

glBegin(GL\_LINES);

glVertex2i(x[0],y[0]);

glVertex2i(x[1],y[1]);

glVertex2i(x[1],y[1]);

glVertex2i(x[2],y[2]);

glVertex2i(x[2],y[2]);

glVertex2i(x[3],y[3]);

glEnd();

glFlush();

double t;

for (t = 0.0; t < 1.0; t += 0.0005)

{

double xt = pow(1-t, 3) \* x[0] + 3 \* t \* pow(1-t, 2) \* x[1] + 3 \* pow(t, 2) \* (1-t) \* x[2] +

pow(t, 3) \* x[3];

double yt = pow(1-t, 3) \* y[0] + 3 \* t \* pow(1-t, 2) \* y[1] + 3 \* pow(t, 2) \* (1-t) \* y[2] +

pow(t, 3) \* y[3];

putpixel(xt, yt);

}

}

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

cout<<"\n \t Enter The Four Points x space y ";

for(int i=0;i<4;i++){

cout<<"\n \t Enter x and y for "<<i<<" = ";

cin>>x[i]>>y[i];

}

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(640,480);

glutInitWindowPosition(200,200);

glutCreateWindow("Bezier 4 point");

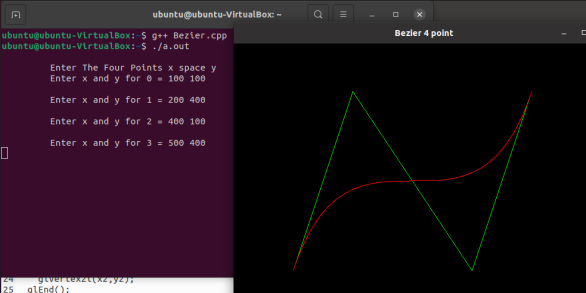
init();

glutDisplayFunc(Algorithm);

glutMainLoop();

return 0;

}



CODE FOR Koch Curve

#include <iostream>

#include <math.h>

#include <time.h>

#include <GL/glut.h>

using namespace std;

double x,y,len,angle;

int it;

void init(){

glClearColor(1.0,1.0,1.0,0.0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0,640,0,480);

glClear(GL\_COLOR\_BUFFER\_BIT);

}

void line1(int x1, int y11, int x2,int y2){

glColor3f(0,1,0);

glBegin(GL\_LINES);

glVertex2i(x1,y11);

glVertex2i(x2,y2);

glEnd();

glFlush();

}

void k\_curve(double x, double y, double len, double angle, int it){

if(it>0){

len /=3;

k\_curve(x,y,len,angle,(it-1));

x += (len \* cosl(angle \* (M\_PI)/180));

y += (len \* sinl(angle \* (M\_PI)/180));

k\_curve(x,y, len, angle+60,(it-1));

x += (len \* cosl((angle + 60) \* (M\_PI)/180));

y += (len \* sinl((angle + 60) \* (M\_PI)/180));

k\_curve(x,y, len, angle-60,(it-1));

x += (len \* cosl((angle - 60) \* (M\_PI)/180));

y += (len \* sinl((angle - 60) \* (M\_PI)/180));

k\_curve(x,y,len,angle,(it-1));

}

else

{

line1(x,y,(int)(x + len \* cosl(angle \* (M\_PI)/180) + 0.5),(int)(y + len \* sinl(angle \*

(M\_PI)/180) + 0.5));

}

}

void Algorithm(){

k\_curve(x,y,len,angle,it);

}

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

cout<<"\n Enter Starting Point x space y ";

cin>>x>>y;

cout <<"\n Lenght of line and space angle of line";

cin>>len>>angle;

cout<<"\n No. of ittration ";

cin>>it;

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(640,480);

glutInitWindowPosition(200,200);

glutCreateWindow("Koch");

init();

glutDisplayFunc(Algorithm);

glutMainLoop();

return 0;

}

OUTPUT

