**HÌNH HỌC FRACTAL**

**Họ và tên sinh viên:**Phan Trần Nhật Hạ

**Mã sinh viên:**102210159**Nhóm:** 21Nh12

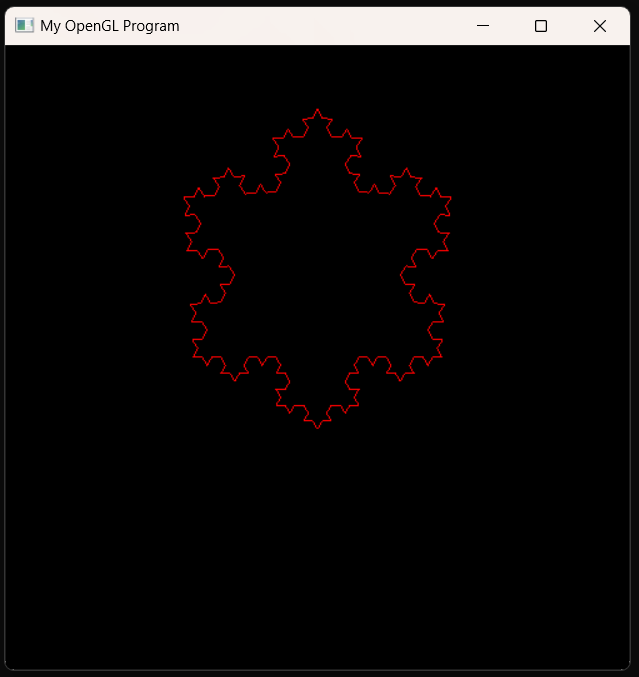
[**1.**     **Đường cong Kock**](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-HinhHocFractal.htm#_Toc103140896)

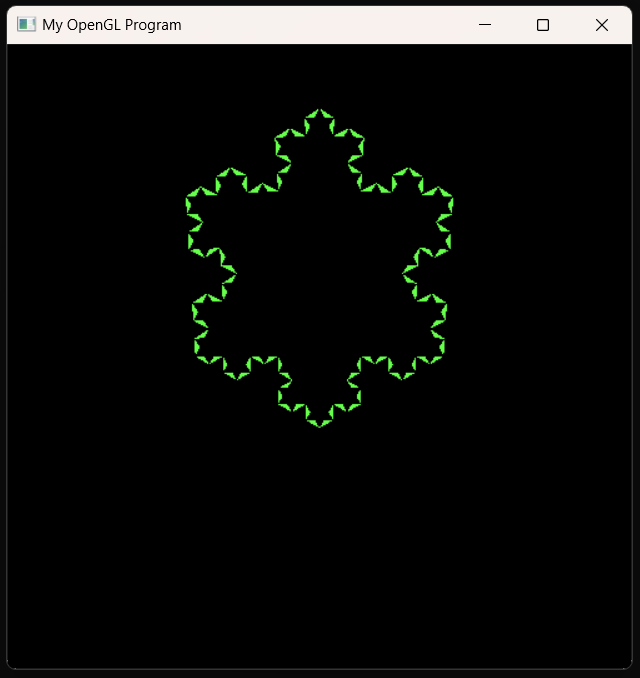
[**2.**     **Chương trình Mandelbrotset01.cpp**](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-HinhHocFractal.htm#_Toc103140897)

[**3.**     **Chương trình Mandelbrotset02.cpp**](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-HinhHocFractal.htm#_Toc103140898)

[**4.**     **Bài tập**](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-HinhHocFractal.htm#_Toc103140899)

**1.    Đường cong Kock**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**2.    Chương trình Mandelbrotset01.cpp**

**A screenshot of a computer

Description automatically generated**

**3.    Chương trình Mandelbrotset02.cpp**

A red and black fractal

Description automatically generated with low confidence

**4.    Bài tập**

1)     Viết chương trình nhập *n* để vẽ đường cong Kock bậc *n*.

#include <GL/glut.h>

#include <cmath>

#include<stdio.h>

int n;

void KochCurve(int n, double x1, double y1, double x2, double y2)

{

if (n == 0) {

glBegin(GL\_LINES);

glVertex2d(x1, y1);

glVertex2d(x2, y2);

glEnd();

}

else

{

double x3 = (2\*x1 + x2) / 3;

double y3 = (2\*y1 + y2) / 3;

double x4 = (x1 + x2) / 2 + (y1 - y2) / (2\*sqrt(3));

double y4 = (y1 + y2) / 2 + (x2 - x1) / (2\*sqrt(3));

double x5 = (x1 + 2\*x2) / 3;

double y5 = (y1 + 2\*y2) / 3;

KochCurve(n-1, x1, y1, x3, y3);

KochCurve(n-1, x3, y3, x4, y4);

KochCurve(n-1, x4, y4, x5, y5);

KochCurve(n-1, x5, y5, x2, y2);

}

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 0.5, 1.0);

glLoadIdentity();

KochCurve(n, -0.8, 0.0, 0.8, 0.0);

glFlush();

}

void reshape(int width, int height)

{

glViewport(0, 0, width, height);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);

glMatrixMode(GL\_MODELVIEW);

}

void init()

{

glClearColor(0.0, 0.0, 0.0, 0.0);

}

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

{

printf("----------BAI TAP 1----------\n");

printf("Nhap bac n : ");

scanf("%d",&n);

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("Koch Curve");

init();

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}

A screenshot of a computer

Description automatically generated

2)     Viết chương trình nhập *n* và vẽ đường cong C cấp *n*.

#include <iostream>

#include <math.h>

#include <GL/glut.h>

#include<stdio.h>

#define PI 3.14159265358979323846

int n;

void drawLine(float x1, float y1, float x2, float y2)

{

glBegin(GL\_LINES);

glVertex2f(x1, y1);

glVertex2f(x2, y2);

glEnd();

}

void KochCurve(int n, float x1, float y1, float x2, float y2)

{

if (n == 0)

{

drawLine(x1, y1, x2, y2);

}

else

{

float dx = x2 - x1;

float dy = y2 - y1;

float dist = sqrt(dx\*dx + dy\*dy);

float angle = atan2(dy, dx);

float segment = dist / 3.0;

float xA = x1 + cos(angle) \* segment;

float yA = y1 + sin(angle) \* segment;

float xB = xA + cos(angle - PI / 3.0) \* segment;

float yB = yA + sin(angle - PI / 3.0) \* segment;

float xC = xB + cos(angle + PI / 3.0) \* segment;

float yC = yB + sin(angle + PI / 3.0) \* segment;

float xD = x1 + 2 \* cos(angle) \* segment;

float yD = y1 + 2 \* sin(angle) \* segment;

KochCurve(n - 1, x1, y1, xA, yA);

KochCurve(n - 1, xA, yA, xB, yB);

KochCurve(n - 1, xB, yB, xC, yC);

KochCurve(n - 1, xC, yC, xD, yD);

KochCurve(n - 1, xD, yD, x2, y2);

}

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 0.2, 0.0);

KochCurve(n, -0.8, 0.0, 0.8, 0.0);

glFlush();

}

void init()

{

glClearColor(0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-1.0, 1.0, -1.0, 1.0);

}

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

{

printf("----------BAI TAP 2----------\n");

printf("Nhap bac n : ");

scanf("%d",&n);

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("Koch Curve - C Curve - Dragon Curve");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

A screenshot of a computer

Description automatically generated

3)     Viết chương trình nhập *n* và vẽ đường cong Rồng cấp *n*.

#include <GL/glut.h>

#include <math.h>

#include<stdio.h>

using namespace std;

int n;

const float PI = 3.14159265358979323846;

void DragonCurve(int n, float x1, float y1, float x2, float y2)

{

if (n == 0)

{

glVertex2f(x1, y1);

glVertex2f(x2, y2);

} else

{

float x3 = (x1 + x2 + (y2 - y1)) / 2;

float y3 = (y1 + y2 + (x1 - x2)) / 2;

DragonCurve(n-1, x1, y1, x3, y3);

DragonCurve(n-1, x2, y2, x3, y3);

}

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.3, 1.0, 1.0);

glBegin(GL\_LINES);

DragonCurve(n, -0.5, 0, 0.5, 0);

glEnd();

glFlush();

}

void init()

{

glClearColor(0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-1.0, 1.0, -1.0, 1.0);

}

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

{

printf("----------BAI TAP 3----------\n");

printf("Nhap bac n : ");

scanf("%d",&n);

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(1000, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("Dragon Curve");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

A screen shot of a computer

Description automatically generated with medium confidence

4)     Viết chương trình sinh tập Mandelbrot.

#include <GL/glut.h>

#include <cmath>

const int WIDTH = 600;

const int HEIGHT = 500;

const int MAX\_ITER = 1000;

int is\_in\_mandelbrot(float real, float imag)

{

float z\_real = 0, z\_imag = 0;

for (int i = 0; i < MAX\_ITER; i++)

{

float z\_real\_new = z\_real\*z\_real - z\_imag\*z\_imag + real;

float z\_imag\_new = 2\*z\_real\*z\_imag + imag;

z\_real = z\_real\_new;

z\_imag = z\_imag\_new;

if (z\_real\*z\_real + z\_imag\*z\_imag > 4)

{

return i;

}

}

return -1;

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glPointSize(1.0);

glBegin(GL\_POINTS);

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

{

for (int j = 0; j < HEIGHT; j++)

{

float x = -2.0 + 3.0\*i/WIDTH;

float y = -1.5 + 3.0\*j/HEIGHT;

int n = is\_in\_mandelbrot(x, y);

if (n == -1)

{

glColor3f(1.0, 1.0, 0.4);

glVertex2f(i, j);

}

else

{

float c = n - log(log(sqrt(x\*x + y\*y)))/log(2);

float r = 0.5 + 0.5\*cos(c\*0.05 + 0.1);

float g = 0.5 + 0.5\*cos(c\*0.1 + 1.2);

float b = 0.5 + 0.5\*cos(c\*0.15 + 2.3);

glColor3f(r, g, b);

glVertex2f(i, j);

}

}

}

glEnd();

glFlush();

}

void reshape(int w, int h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, w, 0, h);

glMatrixMode(GL\_MODELVIEW);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(WIDTH, HEIGHT);

glutCreateWindow("Mandelbrot Set");

glClearColor(1.0, 1.0, 1.0, 1.0);

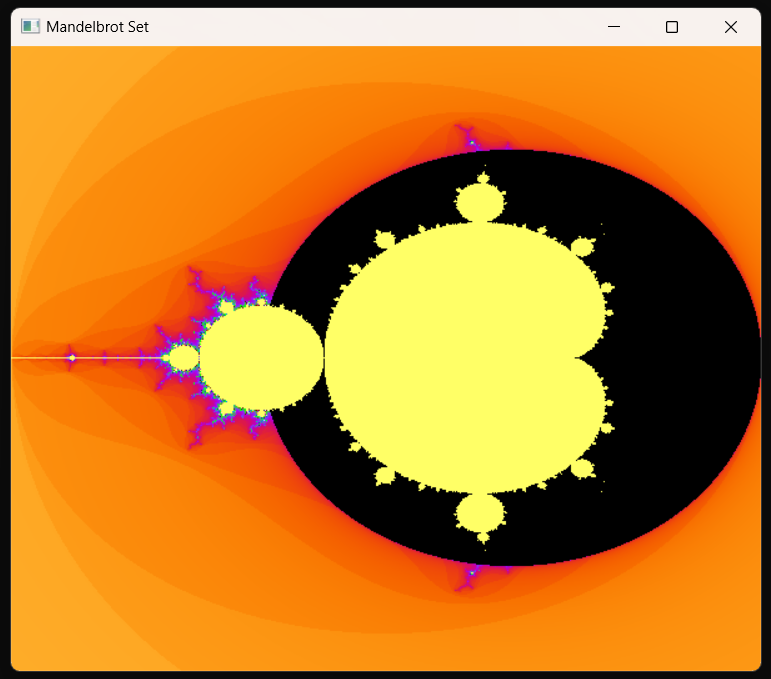
glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}



5)     Viết chương trình sinh đường Pythagoras.

#include <GL/glut.h>

#include <cmath>

#define WIDTH 800

#define HEIGHT 800

int level = 3;

void draw\_triangle(float x, float y, float size, float angle)

{

glBegin(GL\_TRIANGLES);

glVertex2f(x, y);

glVertex2f(x + size \* cos(angle), y + size \* sin(angle));

glVertex2f(x + size \* cos(angle + M\_PI / 2.0), y + size \* sin(angle + M\_PI / 2.0));

glEnd();

}

void draw\_pythagoras(float x, float y, float size, float angle, int level)

{

if (level == 0)

{

return;

}

draw\_triangle(x, y, size, angle);

float x1 = x + size \* cos(angle);

float y1 = y + size \* sin(angle);

float x2 = x1 + size \* cos(angle - M\_PI / 2.0);

float y2 = y1 + size \* sin(angle - M\_PI / 2.0);

float x3 = x + size \* cos(angle + M\_PI / 2.0);

float y3 = y + size \* sin(angle + M\_PI / 2.0);

draw\_pythagoras(x1, y1, size / sqrt(2.0), angle - M\_PI / 4.0, level - 1);

draw\_pythagoras(x2, y2, size / sqrt(2.0), angle - M\_PI / 4.0, level - 1);

draw\_pythagoras(x3, y3, size / sqrt(2.0), angle + M\_PI / 4.0, level - 1);

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glColor3f(0.0, 0.0, 0.0);

draw\_pythagoras(0, 0, WIDTH, HEIGHT, level);

glFlush();

}

void reshape(int w, int h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, w, 0, h);

glMatrixMode(GL\_MODELVIEW);

}

void keyboard(unsigned char key, int x, int y)

{

if (key == '+')

{

level++;

glutPostRedisplay();

}

else

if (key == '-')

{

level--;

if (level < 0)

{

level = 0;

}

glutPostRedisplay();

}

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(WIDTH, HEIGHT);

glutCreateWindow("Pythagoras Fractal");

glClearColor(1.0, 1.0, 1.0, 0.0);

glutDisplayFunc(display);

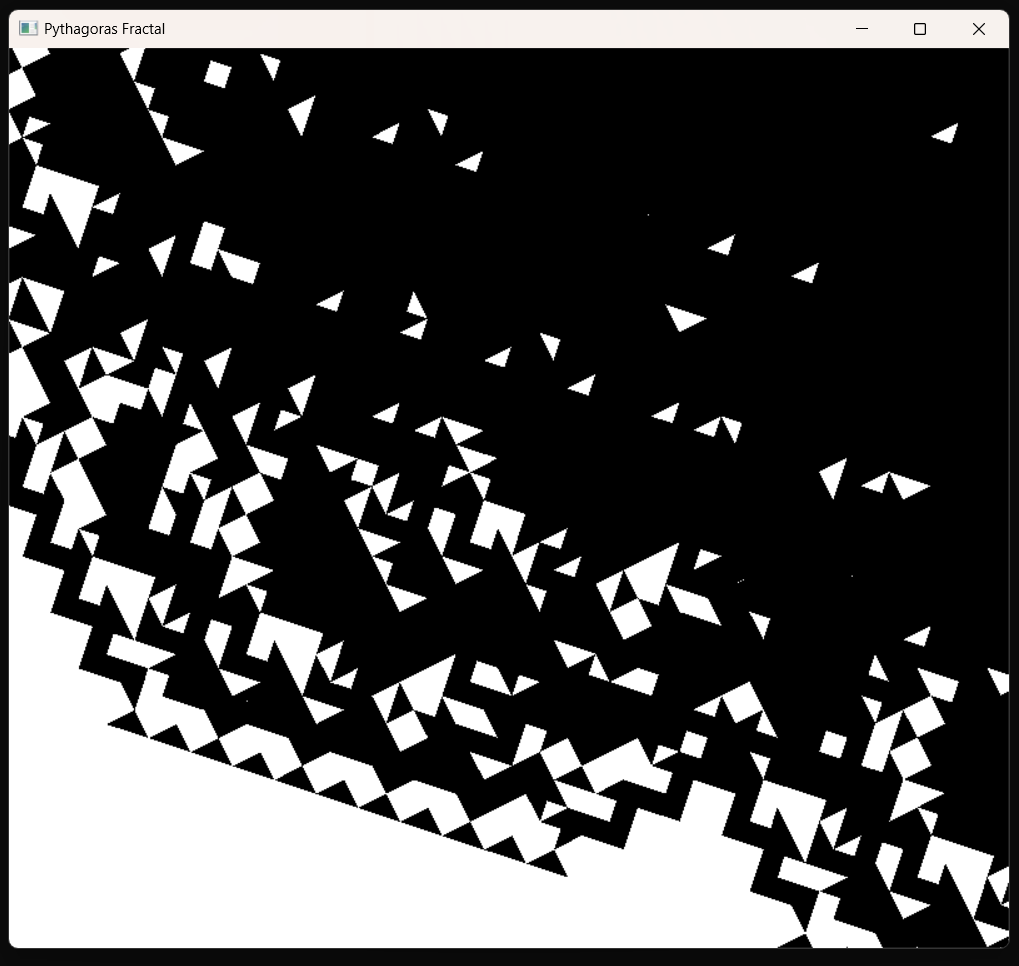
glutReshapeFunc(reshape);

glutKeyboardFunc(keyboard);

glutMainLoop();

return 0;

}



6)     Viết chương trình sinh tập Julia.

#include <GL/glut.h>

#include <complex>

using namespace std;

#define WIDTH 800

#define HEIGHT 800

const complex<float> c(-0.8, 0.156);

void color(float r, float g, float b, float radius, int iter)

{

if (iter == 0)

{

glColor3f(0.0, 0.0, 0.0);

}

else

{

float t = (float) iter / 100.0;

glColor3f(r \* t, g \* t, b \* t);

}

}

void draw\_julia()

{

glBegin(GL\_POINTS);

for (int y = 0; y < HEIGHT; y++)

{

for (int x = 0; x < WIDTH; x++)

{

complex<float> z((float) x / (float) WIDTH \* 4.0 - 2.0, (float) y / (float) HEIGHT \* 4.0 - 2.0);

int iter = 0;

float radius = 0.0;

while (iter < 100 && radius < 4.0)

{

z = z \* z + c;

iter++;

radius = abs(z);

}

color(0.5, 0.5, 1.0, radius, iter);

glVertex2i(x, y);

}

}

glEnd();

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

draw\_julia();

glFlush();

}

void reshape(int w, int h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, w, 0, h);

glMatrixMode(GL\_MODELVIEW);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(WIDTH, HEIGHT);

glutCreateWindow("Julia Set");

glClearColor(0.0, 0.0, 0.0, 0.0);

glPointSize(1.0);

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}

A screenshot of a computer

Description automatically generated

7)     Viết chương trình vẽ các đường cong tô vùng: Phoenix, Hilbert, Sierpinxki.

8)     Khai thác các chức năng của phần mềm Ultra Fractal (https://www.ultrafractal.com/)

----------------------------------------------------------