МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет автоматики и вычислительной техники

Кафедра АСУ

**ОТЧЕТ**

по расчётному графическому заданию

Дисциплина «Компьютерная графика», 3 семестр

Преподаватель: Павенко Евгений Николаевич

Группа: АВТ-113

Студент:

Кашаев Аскер

г. Новосибирск

2022 год

**Цель работы**

Повторить фрактал «Закрученный квадрат», аналогичный проекту по ссылке <http://opita.net/node/853>

**Аннотация:**

По итогу оказалось, что приложенный программный код не имеет никакого отношения к приложенному скриншоту в статье, аналогичное происходит и в приложенном архиве на сайте. Из-за чего мне пришлось самому составить программу по референсу с картинки.

**Реализация:**

Используя уравнение Архимедовой спирали и перевод полярных координат в декартовы, раскрасив закручивающиеся квадраты в контрастные цвета получаем необходимый фрактал.

Добавил **движение** (углубление к центру сцены) в итоговой программе.

В препроцессорных директивах в исходном файле .cpp есть возможность изменить количество итераций фрактала. В зависимости от которого так же меняется скорость углубления в сцене. GLUT не оптимизирован под современные ресурсы компьютеров, от чего максимальное количество итераций находится в пределах 5 000 000, учитывая уже проведённую работу по оптимизации программы. Но и этого количества итераций более чем достаточно (скриншот присутствует в самом конце отчёта).

**Приложение.**

****

#include <windows.h> /\*подключать библиотеку следует только под Windows,

под Linux используем start.sh с кодом

g++ rgzKashaev.cpp -lGL -lGLU -lglut && ./a.out

\*/

#include <GL/gl.h>

#include <GL/glu.h>

#include "../glut.h"

#include <iostream>

#include <vector>

#define ITERATIONS\_COUNT 1000

#define WINDOW\_WIDTH 800

#define WINDOW\_HEIGHT 800

double zoom = ITERATIONS\_COUNT/3.5;

double viewportSize = WINDOW\_WIDTH;

char title[] = "Kashaev Asker, ABT-113";

void drawString(float x, float y, float z, void\* font, char\* string)

{

glColor3f(0.0, 0.0, 0.0);

char\* c;

glRasterPos3f(x, y, z);

for (c = string; \*c != '\0'; c++) {

glutBitmapCharacter(font, \*c);

}

}

// Отрисовка рекурсивного закрученного квадрата

void fract(void) {

double x, y,a ;

double px[4], py[4];

const double x0 = 0, y0 = 0, pi = 3.1415, b = 10, c = 2;

const int count = ITERATIONS\_COUNT;

for (int i = 1; i <= count; i++)

{

x = x0 + (i / c \* cos(i / b));

y = y0 + (i / c \* sin(i / b));

a = i / b;

for (int j = 0; j < 4; j++)

{

px[j] = x + (a \* cos(i / b + pi \* j / 2));

py[j] = y + (a \* sin(i / b + pi \* j / 2));

}

if (i % 2 == 1) glColor3d(0, 0, 0);

else glColor3d(1, 1, 1);

glBegin(GL\_QUADS);

glVertex2d(px[0], py[0]);

glVertex2d(px[1], py[1]);

glVertex2d(px[2], py[2]);

glVertex2d(px[3], py[3]);

glEnd();

glEnable(GL\_LINE\_SMOOTH | GL\_LINE\_STIPPLE);

glBegin(GL\_LINE\_STRIP);

glColor3d(0, 0, 0);

glVertex2d(px[0], py[0]);

glVertex2d(px[1], py[1]);

glVertex2d(px[2], py[2]);

glVertex2d(px[3], py[3]);

glVertex2d(px[0], py[0]);

glEnd();

}

}

void display(void) {

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);

fract();

drawString(0, 0, 0, GLUT\_BITMAP\_HELVETICA\_18, title);

glutSwapBuffers();

}

void init(void) {

glEnable(GL\_COLOR\_MATERIAL);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glClearColor(1, 1, 1, 1);

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-zoom, zoom, -zoom, zoom, -zoom, zoom);

gluLookAt(0, 0, 5, 0, 1, 0, 0, 1, 0);

glMatrixMode(GL\_MODELVIEW);

}

void spinDisplay(void)

{

zoom -= ITERATIONS\_COUNT/1000;

if (abs(zoom) > ITERATIONS\_COUNT / 3.5) zoom = (ITERATIONS\_COUNT / 3.5);

init();

glutPostRedisplay();

}

static void reshape(int w, int h) {

viewportSize = min(w, h);

int x = (w - viewportSize) / 2;

int y = (h - viewportSize) / 2;

glViewport(x, y, viewportSize, viewportSize);

init();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(WINDOW\_WIDTH, WINDOW\_HEIGHT);

glutInitWindowPosition(50, 50);

glutCreateWindow("RGZ : Kashaev ABT-113");

init();

glutDisplayFunc(display);

glutReshapeFunc(reshape);

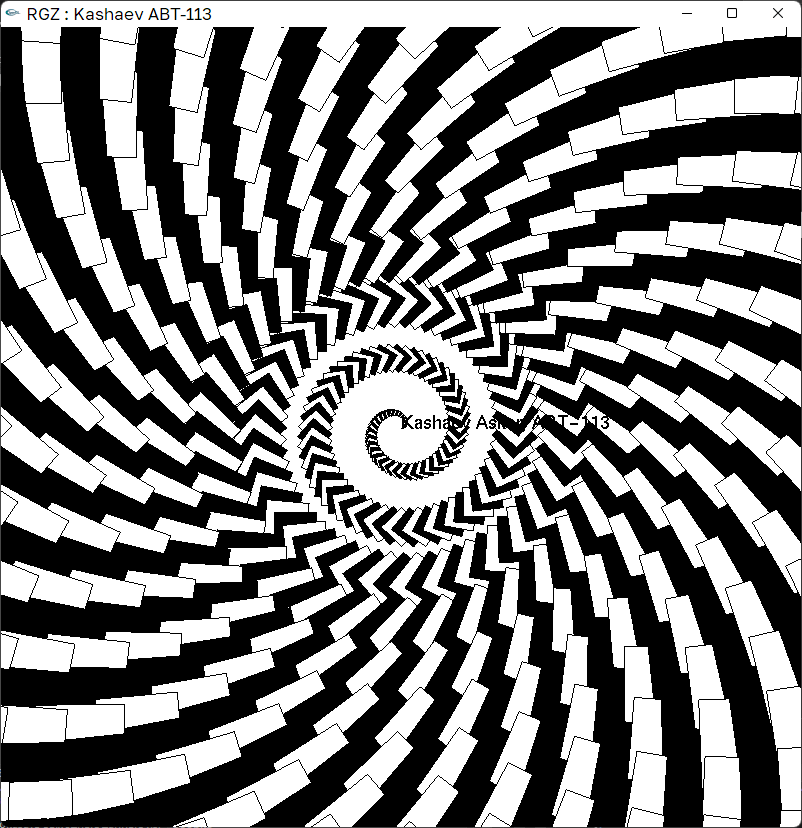
glutIdleFunc(spinDisplay);

glutMainLoop();

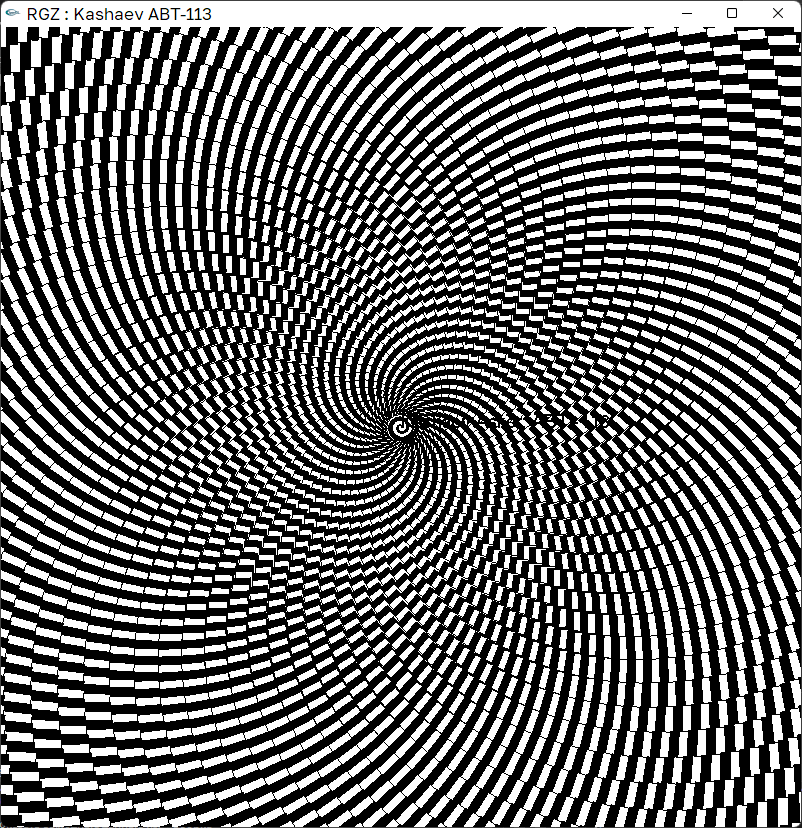
return 0;

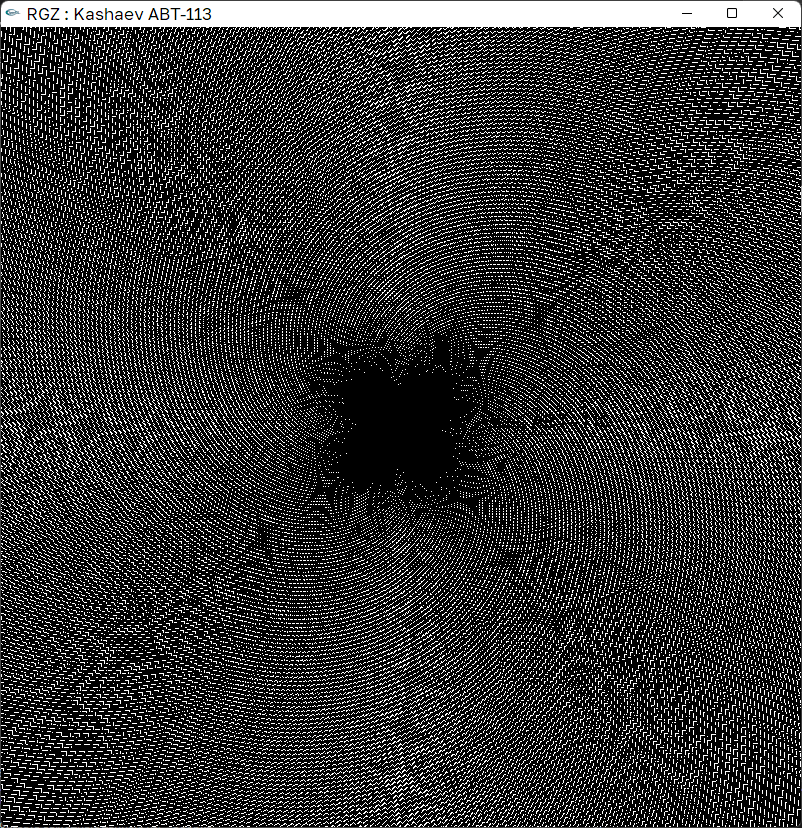
}

**Скриншот**







****

**Скриншот программы при экспериментальных значениях количества итераций = 100 000**