МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**КУРСОВОЙ ПРОЕКТ**

по дисциплине: “Разработка приложений в визуальных средах”

на тему: **“Формирование и отображение квадратов”**

Выполнил:студент группы 10701122 Ханяк Е.Д.

Руководитель**:** канд. т.н., доцент Гурский Н.Н.

Минск 2024

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту**

**по дисциплине «Разработка приложений в визуальных средах»**

Тема: «***Формирование и отображение квадратов***»

Выполнил: ст. гр. 10701122 (подпись)

Ханяк Е.Д.

Руководитель: канд. т.н., доцент  
Гурский Н.Н. (подпись)

Минск 2024

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 4](#_Toc167837084)

[1 Математическая формулировка задачи 5](#_Toc167837085)

[2 Описание программы 6](#_Toc167837086)

[2.1 Структурная схема приложения 6](#_Toc167837087)

[2.2 Описание структурных элементов приложения 6](#_Toc167837088)

[2.3 Основные возможности приложения 9](#_Toc167837089)

[3 Руководство пользователя 10](#_Toc167837090)

[4 Методика испытаний 17](#_Toc167837091)

[ЗАКЛЮЧЕНИЕ 23](#_Toc167837092)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 24](#_Toc167837093)

[**ПРИЛОЖЕНИЕ А** 25](#_Toc167837094)

# ВВЕДЕНИЕ

Фрактал — это множество, обладающее свойством самоподобия, т.е. часть фрактала является похожей на весь фрактал в целом. Для создания фракталов в программном обеспечении в основном используется рекурсия.

Одним из видов фракталов является квадратный фрактал. Самым распространенный фрактал такого типа известен под названием «Ковер Серпинского», представленный на рисунке 1.

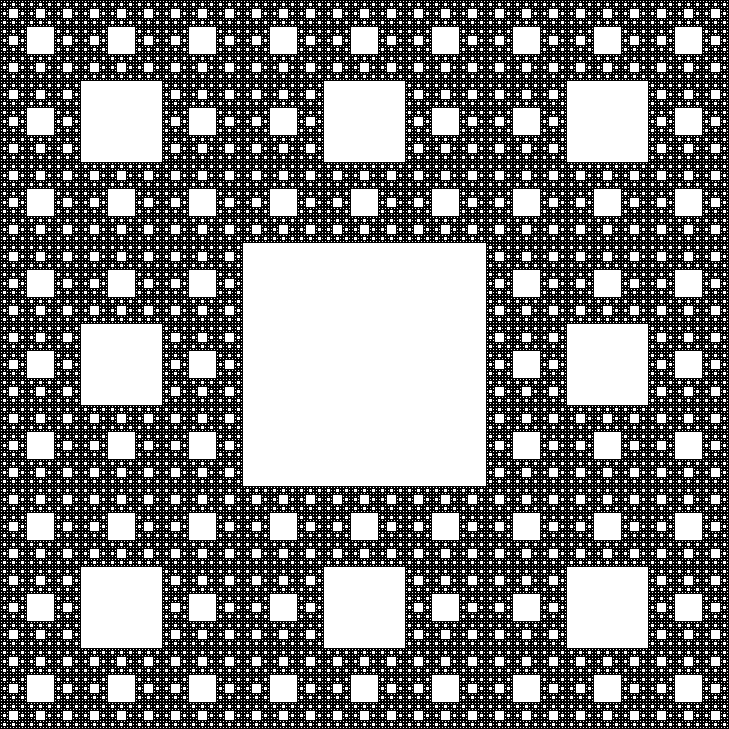


Рисунок 1 – Ковер Серпинского

Однако данный курсовой проект будет посвящен кубическим фракталам — продвинутой версии квадратных. Они очень схожи, однако в отличии от квадратного, где более мелкие квадраты появляется от 4 граней, будут появляться кубы от 6 граней.

Для реализации данного курсового проекта была выбрана платформа .NET (Windows Forms). Причиной этому является ряд плюсов:

1. Простота разработки: Windows forms предоставляет визуальный дизайнер в Visual Studio, который значительно упрощает разработку графически богатых приложений.
2. Богатый набор готовых элементов управления: Windows Forms предоставляет большое количество готовых утилит, таких как текстовые поля, таблицы, кнопки и другие.
3. Легкость развертывания и обновления: приложения Windows Forms собираются несколькими нажатиями мыши.

В заключение, в этой работе будет представлен процесс разработки приложения для генерации кубических фракталов с заданными параметрами, а также результаты его тестирования. Эта работа будет полезна для 3D-дизайнеров, для генерации ими моделей и текстур, а также для разработчиков программного обеспечение, заинтересованных в развитии области трехмерных генеративных приложений.

# Математическая формулировка задачи

Фрактал начинается с одного куба (основного куба) и на каждой итерации (или уровне глубины) вокруг каждого куба добавляются шесть новых кубов, смещенных по осям координат на определенное расстояние.

Пусть обозначает множество кубов, где:

* *—* координаты центра куба.
* *—* длина ребра куба.
* *n —* текущая глубина рекурсии.
* *—* коэффициент расстояния.

На каждом уровне глубины каждый куб с координатами центра и размером порождает 6 новых кубов с центрами в точках:

Процесс можно описать следующим образом: на нулевой итерации содержит один куб с центром и размером .

Для каждого уровня рекурсии от до :

Пусть — множество кубов на уровне .

Для каждого куба из :

Добавляем новые кубы в с центами .

Таким образом построение кубического фрактала с параметрами глубина , длина ребра , и коэффициент расстояния заключается в рекурсивном процессе деления и добавления новых кубов вокруг каждого существующего куба, где каждый новый куб смещен от центра исходного куба на расстояние, пропорциональное размеру исходного куба и коэффициенту . Этот процесс продолжается до достижения заданной глубины рекурсии .

# Описание программы

# 2.1 Структурная схема приложения

Программа была разработана таким образом, чтобы полностью покрыть всевозможные запросы пользователей. Структурная схема приложения представлена на рисунке ниже.

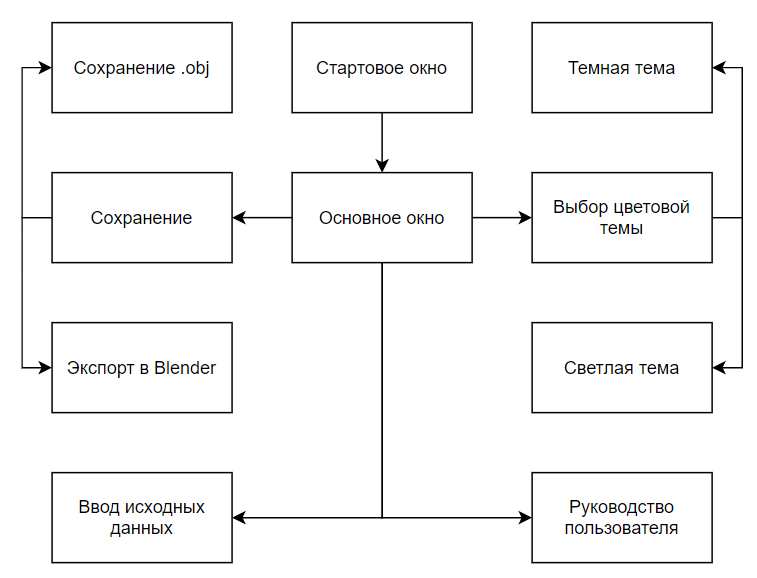


Рисунок 2.1 – Структурная схема приложения

# 2.2 Описание структурных элементов приложения

Для разработки приложения были разработаны 4 класса: FractalReder, ModelSaver, ControlEventHandler и Colors. UML-диаграмма классов представавлена на рисунке 2.2.

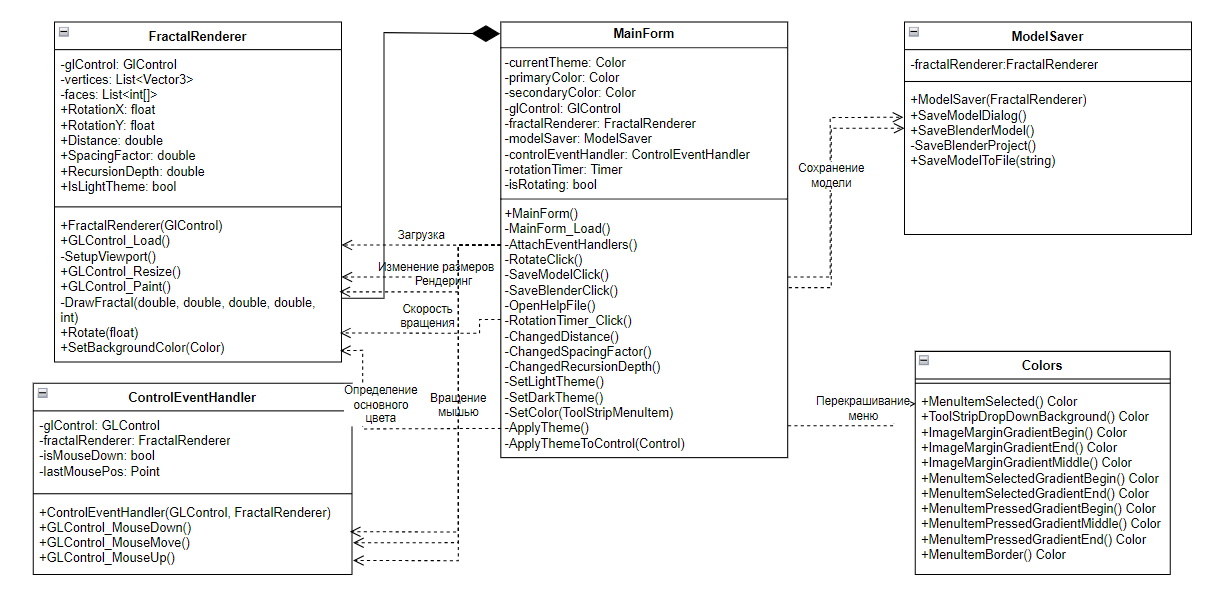


Рисунок 2.2 – Диаграмма классов приложения

Ниже представлено краткое описание всех структурных элементов, представленных на схеме. Руководство по взаимодействию с ними и изображения их пользовательского интерфейса присутствует в следующем разделе записки. Исходный код всех компонентов приведён в Приложении А.

**Класс MainForm:** Главное окно приложения для отображения и управления 3D-фракталом.

Поля:

- currentTheme – Поле, определяющее текущую тему оформления (светлая или тёмная).

- primaryColor – Основной цвет текущей темы оформления.

- secondaryColor – Второстепенный цвет текущей темы оформления.

- glControl – Поле для управления OpenGL-рендерингом.

- fractalRenderer – Поле для рендеринга 3D-фракталов.

- modelSaver – Поле для сохранения моделей фракталов.

- controlEventHandler – Поле для обработки событий управления.

- rotationTimer – Таймер для вращения фрактала.

- isRotating – Поле, определяющее, вращается ли фрактал в данный момент.

Методы:

- MainForm() – Конструктор класса.

- MainForm\_Load(object sender, EventArgs e) – Метод обработки загрузки главной формы.

- AttachEventHandlers() – Метод для присоединения обработчиков событий.

- RotateClick(object sender, EventArgs e) – Метод обработки нажатия кнопки вращения фрактала.

- SaveModelClick(object sender, EventArgs e) – Метод обработки нажатия кнопки сохранения модели.

- SaveBlenderClick(object sender, EventArgs e) – Метод обработки нажатия кнопки сохранения модели для Blender.

- OpenHelpFile(object sender, EventArgs e) – Метод обработки нажатия кнопки "Справка".

- RotationTimer\_Tick(object sender, EventArgs e) – Метод обработки тика таймера вращения.

- ChangedDistance(object sender, EventArgs e) – Метод обработки изменения дистанции.

- ChangedSpacingFactor(object sender, EventArgs e) – Метод обработки изменения коэффициента расстояния.

- ChangedRecursionDepth(object sender, EventArgs e) – Метод обработки изменения глубины рекурсии.

- SetLightTheme(object sender, EventArgs e) – Метод установки светлой темы.

- SetDarkTheme(object sender, EventArgs e) – Метод установки темной темы.

- ApplyTheme() – Метод применения текущей темы к интерфейсу.

- ApplyThemeToControl(Control control) – Метод применения текущей темы к указанному контролу.

- SetColor(ToolStripMenuItem item) – Метод установки цвета для элементов меню.

**Класс FractalRenderer**: Класс для рендеринга 3D-фракталов с использованием OpenGL.

Поля:

- glControl – Поле для управления OpenGL-рендерингом.

- vertices – Список вершин для построения фрактала.

- faces – Список граней для построения фрактала.

Свойства:

- Vertices – Список вершин, доступный только для чтения.

- Faces – Список граней, доступный только для чтения.

- RotationX – Угол поворота вокруг оси X.

- RotationY – Угол поворота вокруг оси Y.

- Distance – Расстояние до объекта фрактала.

- SpacingFactor – Коэффициент расстояния между фракталами.

- RecursionDepth – Глубина рекурсии для построения фрактала.

- IsLightTheme – Флаг, указывающий, используется ли светлая тема.

Методы:

- FractalRenderer(GLControl glControl) – Конструктор класса.

- GLControl\_Load(object sender, EventArgs e) – Метод инициализации OpenGL.

- SetupViewport() – Метод настройки вьюпорта.

- GLControl\_Resize(object sender, EventArgs e) – Метод обработки изменения размера окна.

- GLControl\_Paint(object sender, PaintEventArgs e) – Метод рендеринга сцены.

- DrawFractal(double x, double y, double z, double size, int depth) – Метод построения фрактала.

- Rotate(float angle) – Метод вращения фрактала.

- SetBackgroundColor(Color color) – Метод установки цвета фона.

**Класс ModelSaver**: Класс для сохранения моделей 3D-фракталов, сгенерированных FractalRenderer.

Поля:

- fractalRenderer – Поле для рендерера фракталов.

Методы:

- ModelSaver(FractalRenderer renderer) – Конструктор класса.

- SaveModelDialog() – Метод для сохранения модели в формате OBJ с использованием диалогового окна.

- SaveBlenderModel() – Метод для сохранения модели и запуска Blender с последующим выполнением скрипта.

- SaveBlenderProject() – Метод для сохранения проекта Blender после его обработки.

- SaveModelToFile(string filename) – Метод для сохранения модели в файл.

**Класс ControlEventHandler**: Обработчик событий для GLControl

Поля:

- glControl – Поле, хранящее ссылку на объект GLControl.

- fractalRenderer – Поле, хранящее ссылку на объект FractalRenderer.

- isMouseDown – Булевое поле, указывающее, нажата ли кнопка мыши.

- lastMousePos – Поле, хранящее последние координаты мыши.

Методы:

- ControlEventHandler(GLControl glControl, FractalRenderer fractalRenderer) – Конструктор класса.

- GLControl\_MouseDown(object sender, MouseEventArgs e) – Метод обработки события нажатия кнопки мыши.

- GLControl\_MouseMove(object sender, MouseEventArgs e) – Метод обработки события перемещения мыши.

- GLControl\_MouseUp(object sender, MouseEventArgs e) – Метод обработки события отпускания кнопки мыши.

# 2.3 Основные возможности приложения

Приложение имеет широкий функционал, полностью покрывающий потребности пользователей. Основным функционалом можно назвать:

1. Создание фракталов по заданым параметрам.
2. Возможность осмотра фрактала со всех сторон.
3. Сохранение фрактала с расширением obj.
4. Экспорт фрактала в Blender.

Дополнительный функционал:

1. Возможность выбора цветовой темы приложения.
2. Возможность автоматического вращения фрактала.

# Руководство пользователя

При запуске программы пользователь видит стартовое меню, на котором представлена информация о работе, исполнителе и руководителе. При нажатии на кнопку “Далее”, запустится окно работы с программой. Также имеется возможность выхода из данного окна путём нажатия на кнопку “Закрыть”. Начальное окно представлено на рисунке 3.1.



Рисунок 3.1 – Начальное окно программы

После нажатия на кнопку “Далее” в начальном окне, перед пользователем будет представлено основное окно для работы. Начальное состояние данного окна изображено на рисунке 3.2.

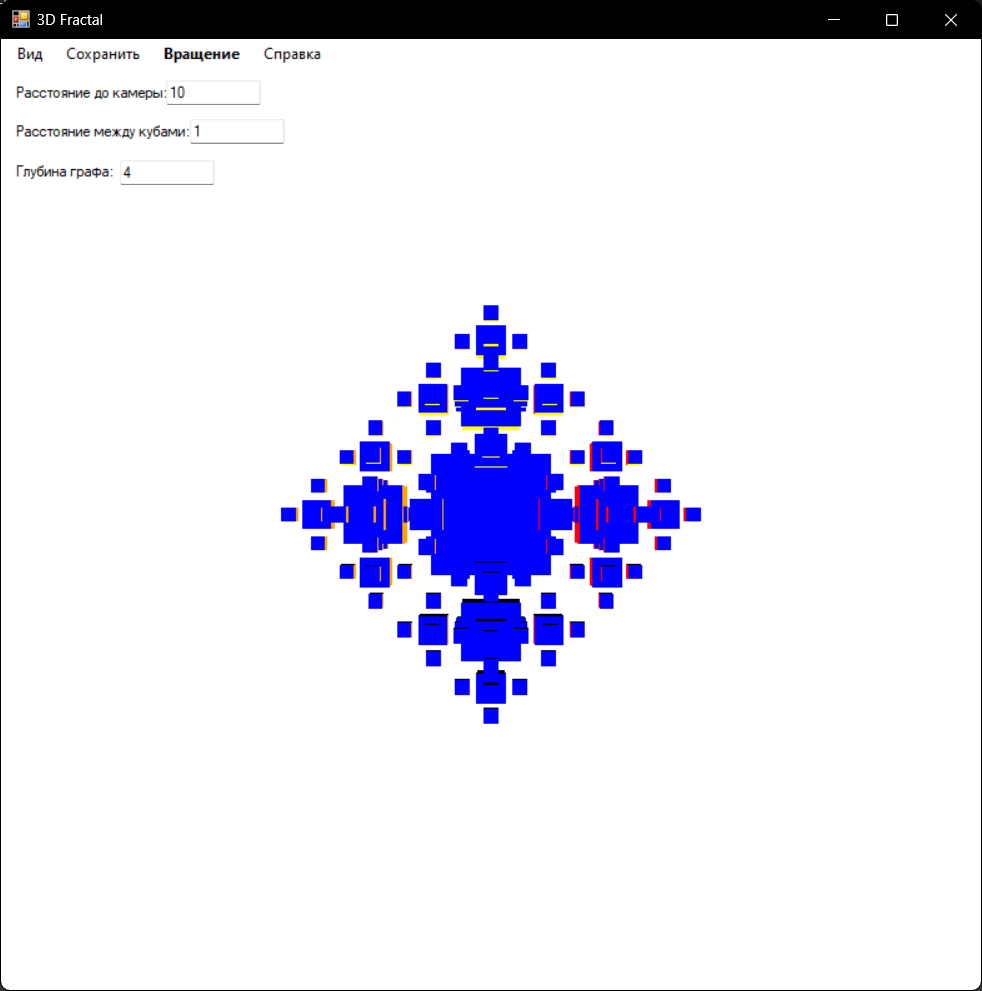


Рисунок 3.2 – Главное окно программы

На главном окне можно заметить следующие элементы:

Меню с вариантами: «Вид», «Сохранить», «Вращение», «Справка».

Вкладка «Вид» содержит два состояния:

1. Светлая тема
2. Темная тема (по умолчанию)

Вкладка «Сохранить» также хранит в себе два варианта:

1. Сохранить модель obj
2. Сохранить модель в Blender

Кнопка «Вращение», предназначенная для запуска автоматического вращения фрактала.

Кнопка «Справка», при нажатии на которую, откроется руководство пользователя.

Текстовые поля для ввода исходных данных.

Разница между светлой и темной темой заключается в выборе основного цвета. Темная тема – черный, светлая тема – белый. Ниже будут примеры обеих тем.

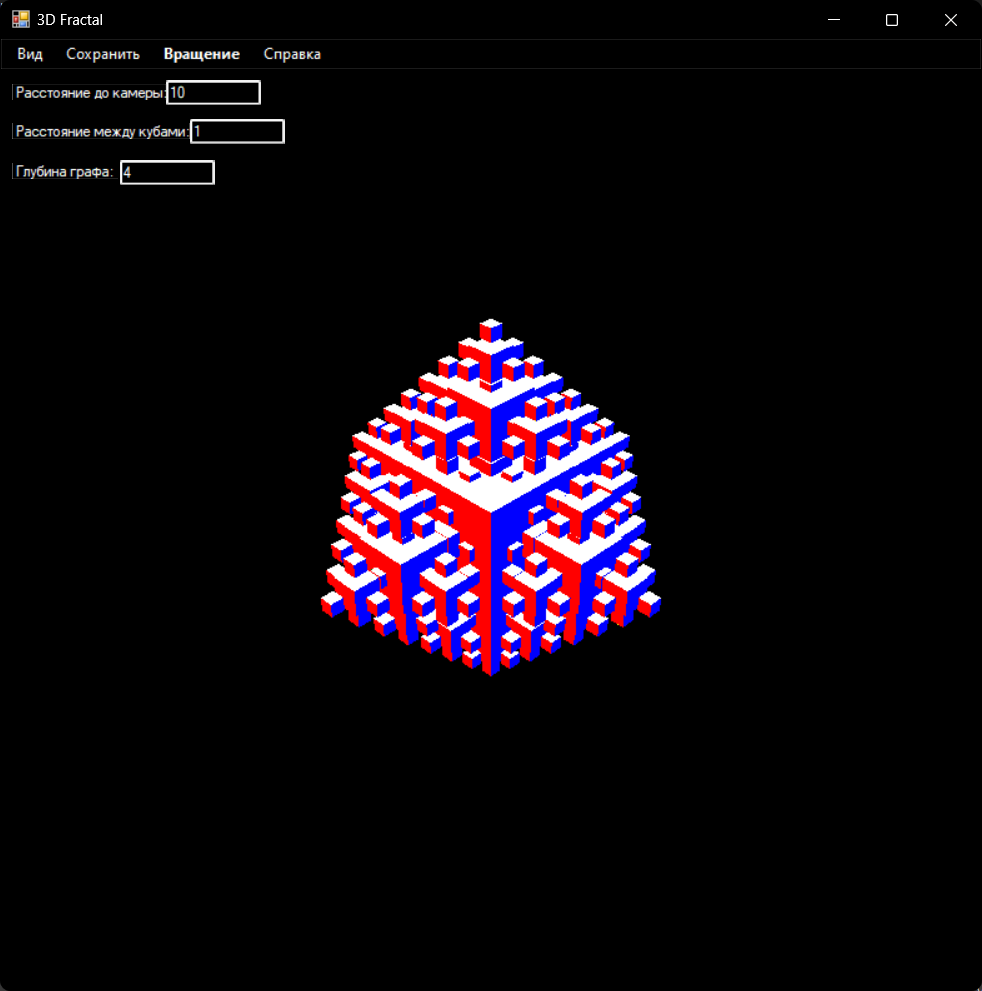


Рисунок 3.3 – Пример темной темы

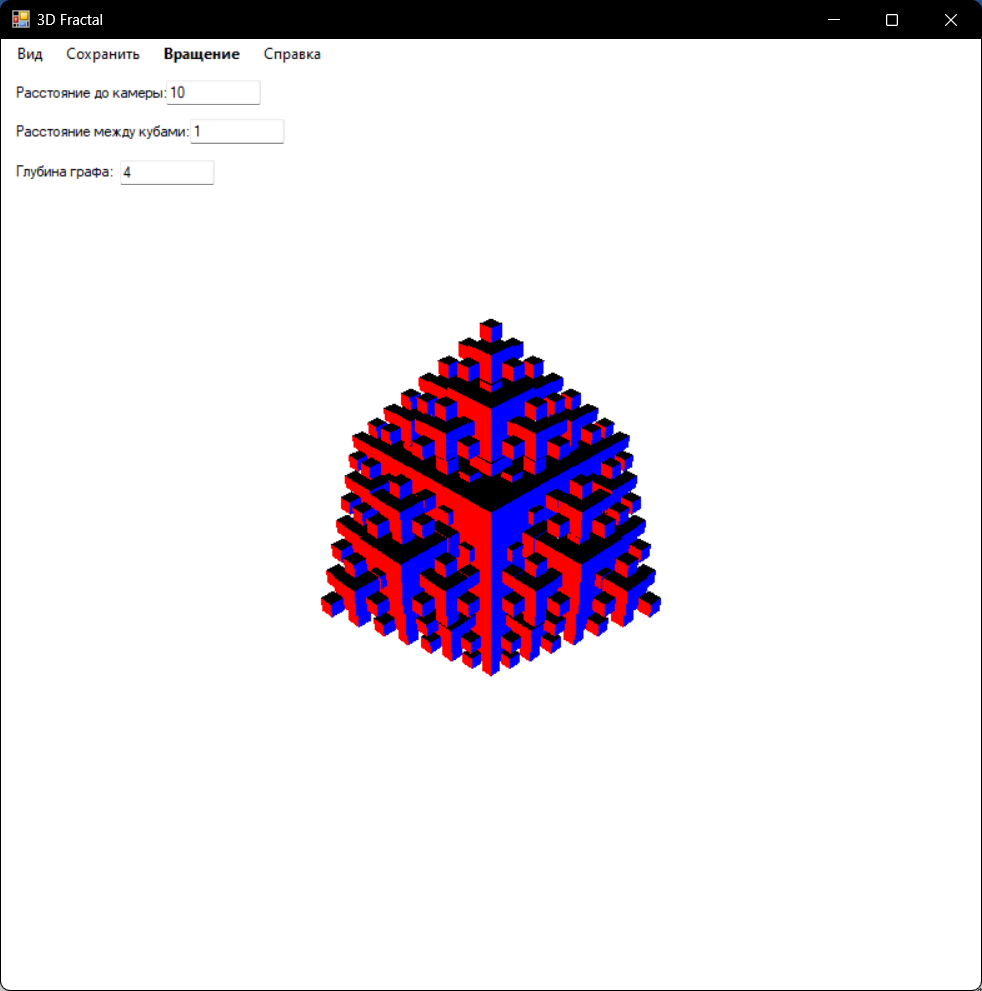


Рисунок 3.4 – Пример светлой темы

Для изменения фрактала достаточно изменить значения в текстовых полях. Отображаемый кубический фрактал моментально отреагирует на эти изменения.

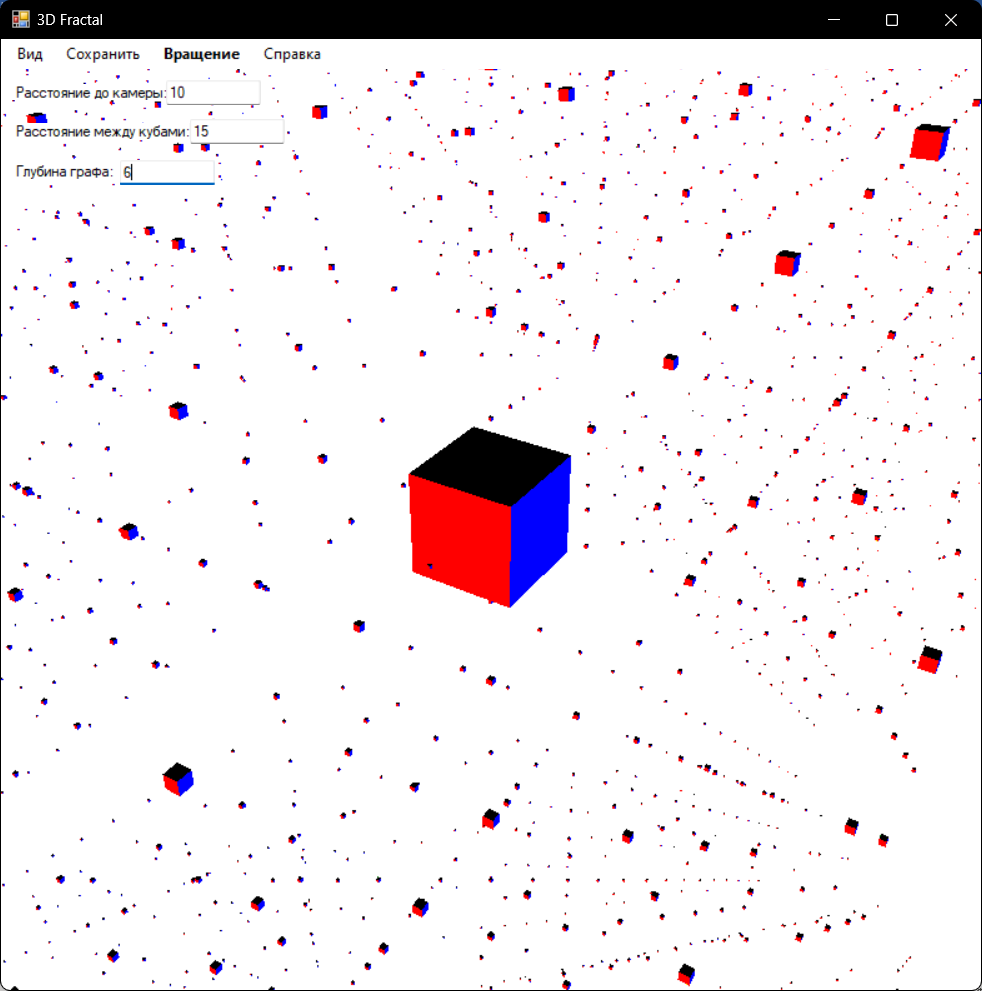


Рисунок 3.5 – Измененный фрактал

В ПО предусмотрено 2 варианта сохранение модели, рассмотрим оба из них.

Первый вариант – сохранение файлом с расширением obj. OBJ — универсальный формат для трехмерных объектов, который поддерживается большинством редакторов.

На рисунке ниже продемонстрирована возможность использования модели в таком формате на примере сайта компании по 3D печати https://studia3d.com/viewer/.

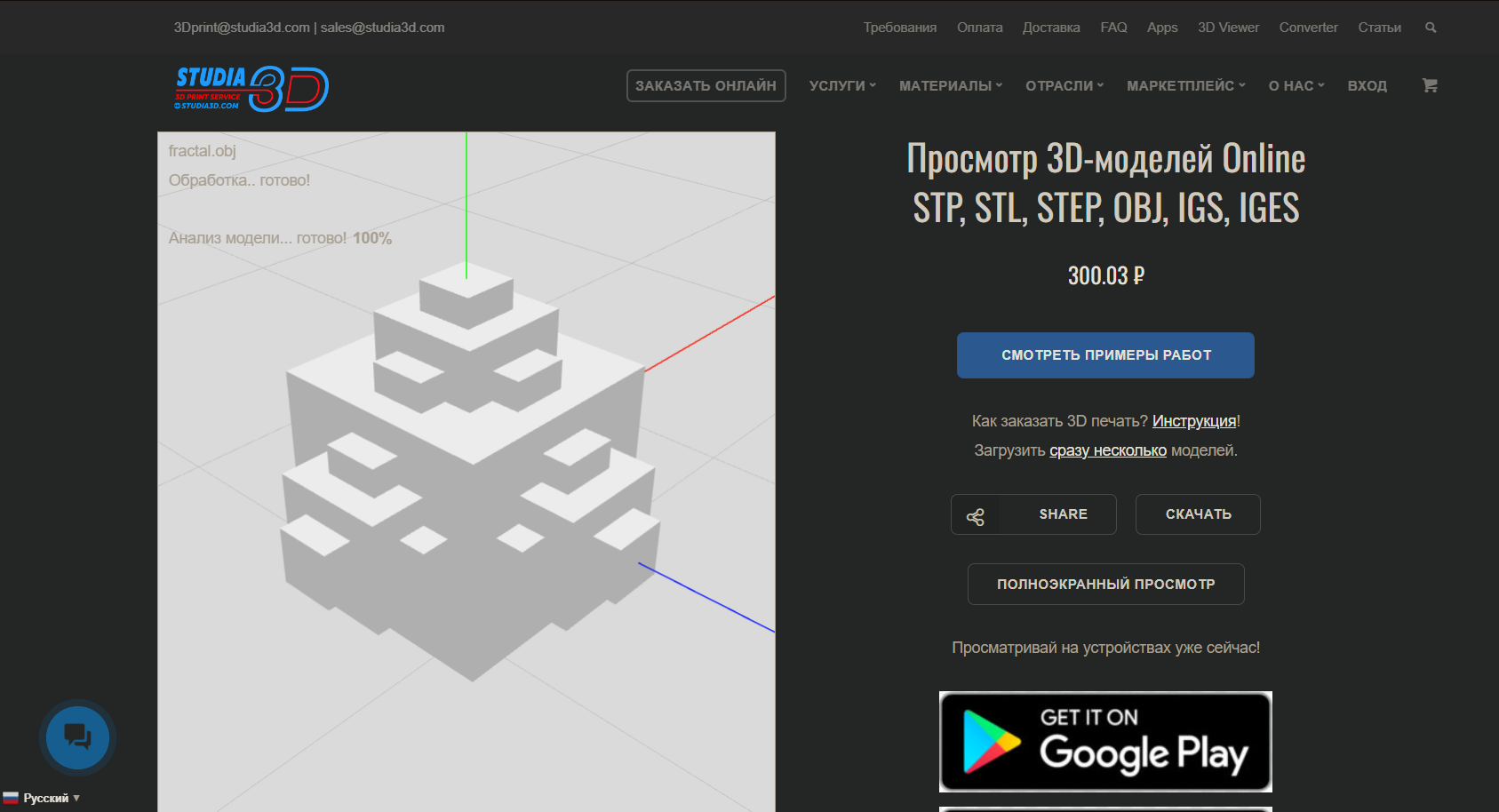


Рисунок 3.6 – Пример применения модели в формате obj

Второй вариант – сохранение проектом в Blender. Blender – профессиональное свободное и открытое программное обеспечение для создания трехмерной компьютерной графики, т.е. такой вариант будет удобен для разработчиков 3D моделей.

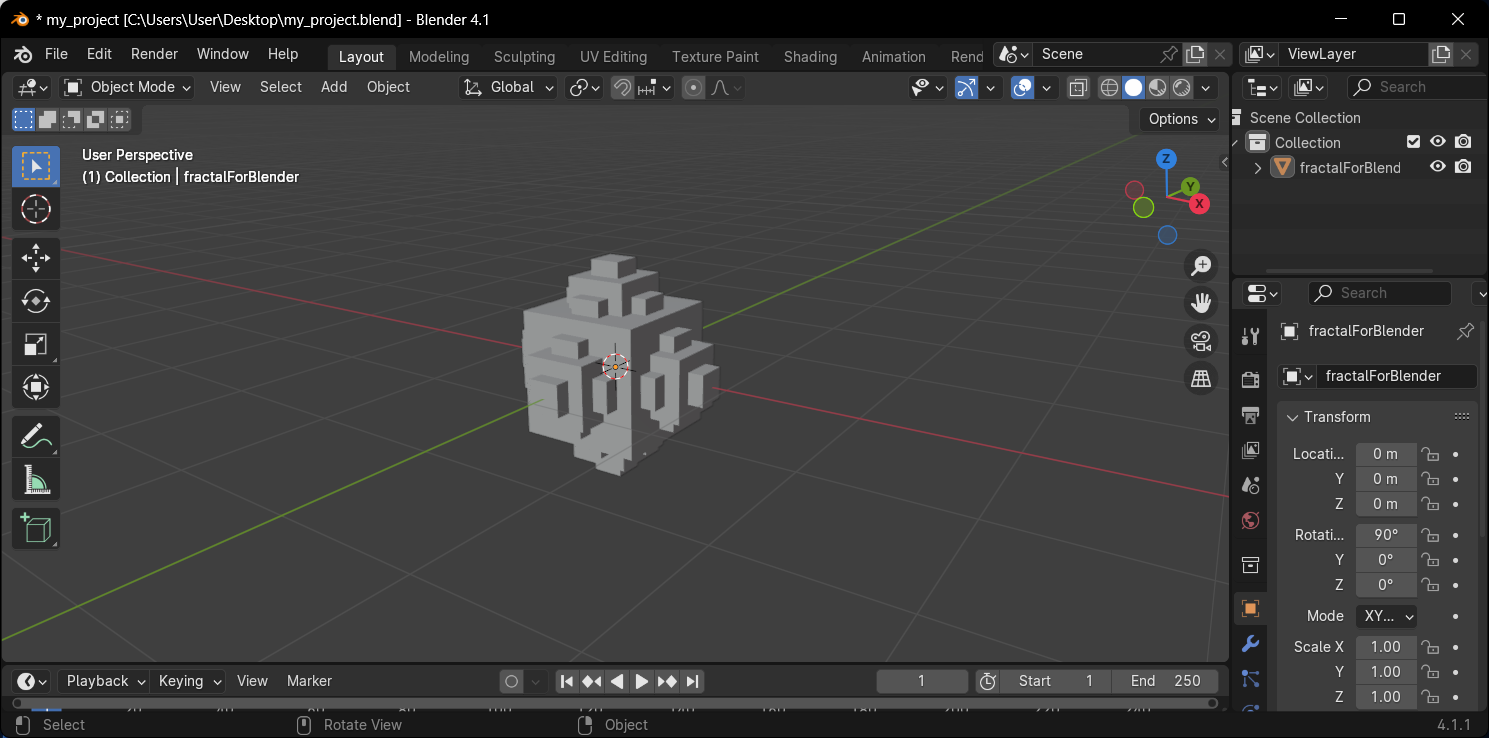


Рисунок 3.7 – Пример экспорта модели в Blender

По нажатию на пункт «Справка» пользователю будет представлено руководство (рисунок 3.8), подробно описывающий все возможности ПО.

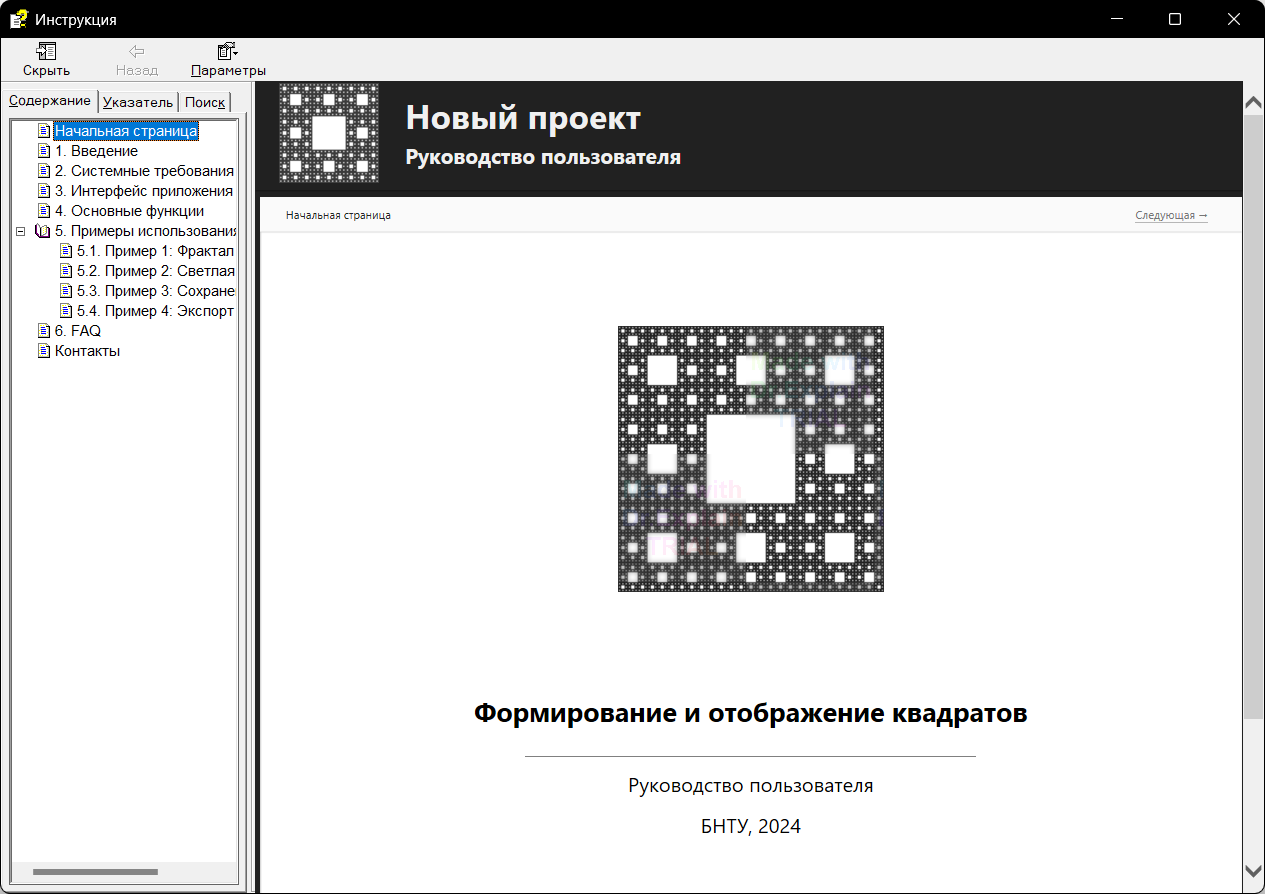


Рисунок 3.8 – Руководство пользователя

# Методика испытаний

Для проверки эффективности разработанного приложения, основанного на методе триангуляции для нахождения определителя матрицы, была проведена методика испытаний. Целью испытаний было убедиться в корректности и работоспособности приложения.

Для проверки корректности работы приложения по построению кубических фракталов было проведено несколько тестов:

1. Тест на корректность расстояния между кубами.

Описание: расстояние между кубами = 0.

Ожидаемый результат: отображается один единственный первоначальный куб.

Результат выполнения представлен на рисунке 4.1.

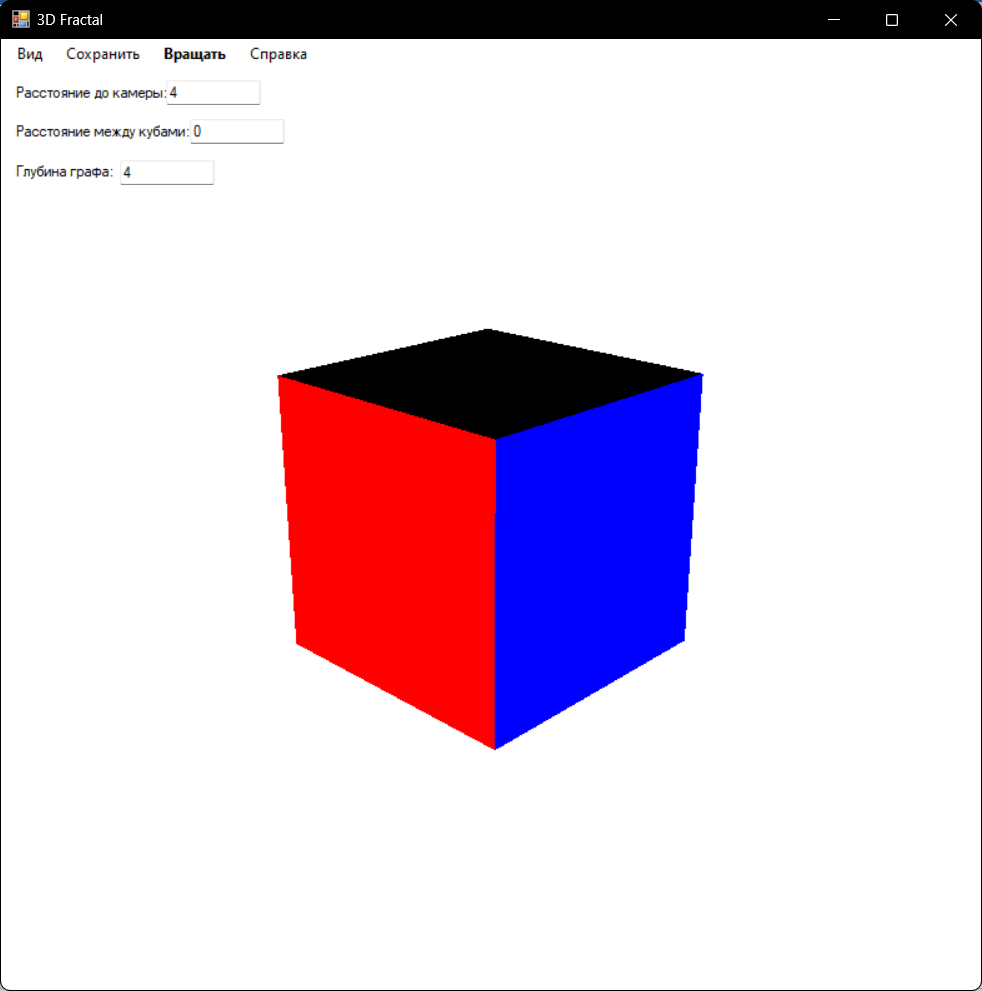


Рисунок 4.1 – Результат первого теста

1. Тест на нарушение числовых диапазонов.

Описание: программа поддерживает глубину рекурсии от 1 до 7, попытка ввода неподходящего значения.

Ожидаемый результат: ПО не позволит это сделать.

Результат выполнения представлен на рисунке 4.2.

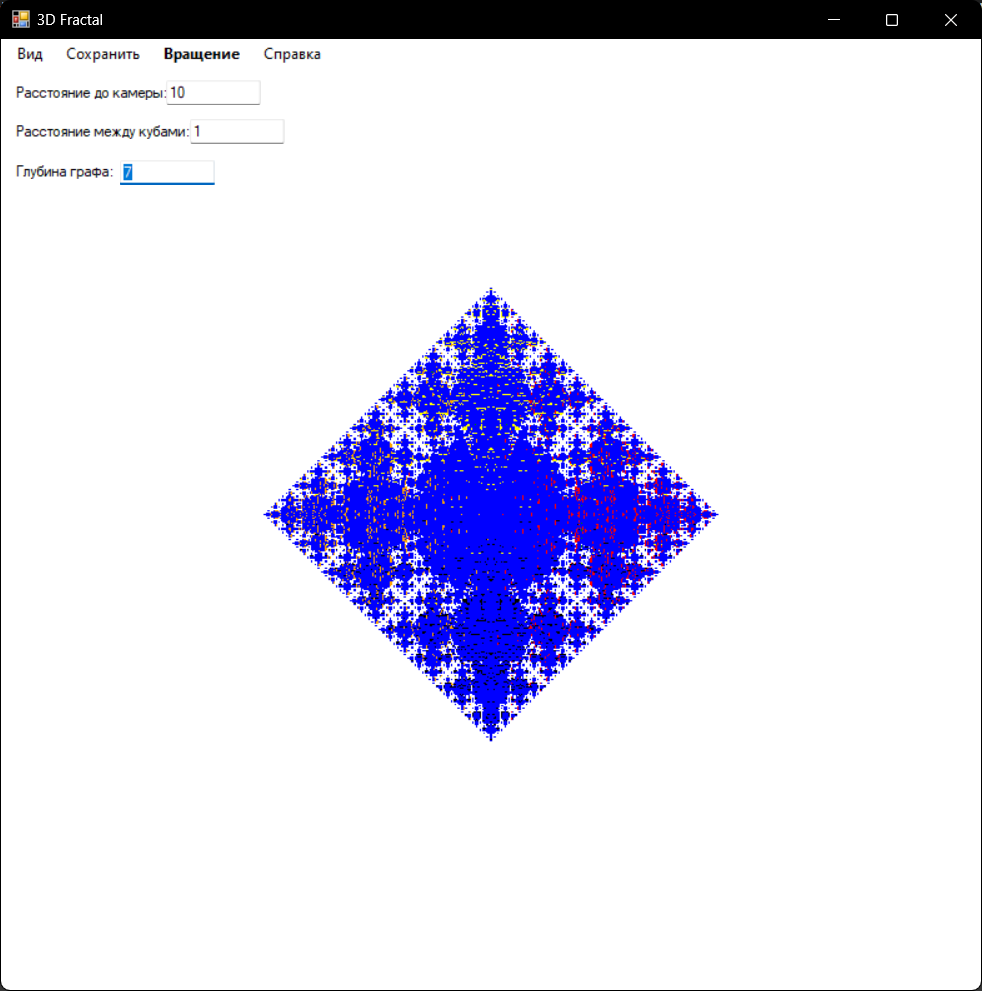


Рисунок 4.2 – Результат второго теста

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данного курсового проекта был рассмотрен процесс создания кубических фракталов, которые представляют собой трехмерные аналоги квадратных фракталов, таких как Ковер Серпинского. Использование платформы .NET (Windows Forms) для реализации проекта позволило воспользоваться преимуществами простоты разработки, наличия богатого набора готовых элементов управления и легкости развертывания приложений.

Разработанное приложение для генерации кубических фракталов продемонстрировало эффективность выбранного подхода и подтвердило жизнеспособность использования рекурсии для создания сложных самоподобных структур. Программа была протестирована и показала хорошие результаты, что делает её полезным инструментом для 3D-дизайнеров и разработчиков программного обеспечения.

В заключение можно отметить, что данный проект расширяет возможности генерации фракталов, добавляя поддержку трехмерных структур. Это открывает новые горизонты для применения фракталов в различных областях, таких как дизайн, моделирование и генеративное искусство. Настоящая работа может послужить основой для дальнейших исследований и разработок в области трехмерных генеративных приложений.

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Мандельброт В. Фракталы: форма, шанс и размер. - М.: Мир, 1982. - 240 с.

2. Пейдж Г. Введение в теорию фракталов. - СПб.: Питер, 2001. - 320 с.

3. Барнсли М. Фракталы повсюду. - М.: Мир, 1989. - 256 с.

4. Декей Т. Фрактальная геометрия природы. - М.: Техносфера, 2006. - 384 с.

5. Жаров В.А. Программирование на платформе .NET. - М.: Бином, 2012. - 472 с.

6. Документация Microsoft по платформе .NET и Windows Forms [Электронный ресурс] // Официальный сайт Microsoft. URL: https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/ (дата обращения: 29.05.2024).

7. Сайт "Фракталы и хаос" [Электронный ресурс] // Fractal-geometry.ru. URL: http://fractal-geometry.ru (дата обращения: 29.05.2024).

## **ПРИЛОЖЕНИЕ А**

**Листинг исходных кодов программы**

**Form1.Designer.cs**

namespace Course\_project

{

partial class Form1

{

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

private void InitializeComponent()

{

System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(Form1));

this.label1 = new System.Windows.Forms.Label();

this.label2 = new System.Windows.Forms.Label();

this.label3 = new System.Windows.Forms.Label();

this.label4 = new System.Windows.Forms.Label();

this.label5 = new System.Windows.Forms.Label();

this.continueButton = new System.Windows.Forms.Button();

this.closeButton = new System.Windows.Forms.Button();

this.label6 = new System.Windows.Forms.Label();

this.label7 = new System.Windows.Forms.Label();

this.label8 = new System.Windows.Forms.Label();

this.pictureBox1 = new System.Windows.Forms.PictureBox();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();

this.SuspendLayout();

//

// label1

//

this.label1.Anchor = ((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Left)

| System.Windows.Forms.AnchorStyles.Right)));

this.label1.AutoSize = true;

this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.label1.Location = new System.Drawing.Point(94, 21);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(622, 25);

this.label1.TabIndex = 0;

this.label1.Text = "БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ";

//

// label2

//

this.label2.Anchor = System.Windows.Forms.AnchorStyles.Top;

this.label2.AutoSize = true;

this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.label2.Location = new System.Drawing.Point(120, 60);

this.label2.Margin = new System.Windows.Forms.Padding(1000, 0, 1000, 0);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(569, 25);

this.label2.TabIndex = 1;

this.label2.Text = "Факультет информационных технологий и робототехники";

//

// label3

//

this.label3.AutoSize = true;

this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.label3.Location = new System.Drawing.Point(218, 168);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(365, 20);

this.label3.TabIndex = 2;

this.label3.Text = "Формирование и отображение квадратов";

//

// label4

//

this.label4.AutoSize = true;

this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.label4.Location = new System.Drawing.Point(413, 280);

this.label4.Name = "label4";

this.label4.Size = new System.Drawing.Size(312, 20);

this.label4.TabIndex = 3;

this.label4.Text = "Выполнил: студент группы 10701122";

//

// label5

//

this.label5.AutoSize = true;

this.label5.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.label5.Location = new System.Drawing.Point(356, 515);

this.label5.Name = "label5";

this.label5.Size = new System.Drawing.Size(106, 20);

this.label5.TabIndex = 4;

this.label5.Text = "Минск, 2024";

//

// continueButton

//

this.continueButton.Location = new System.Drawing.Point(163, 554);

this.continueButton.Name = "continueButton";

this.continueButton.Size = new System.Drawing.Size(143, 53);

this.continueButton.TabIndex = 5;

this.continueButton.Text = "Далее";

this.continueButton.UseVisualStyleBackColor = true;

this.continueButton.Click += new System.EventHandler(this.ContinueUse);

//

// closeButton

//

this.closeButton.Location = new System.Drawing.Point(523, 554);

this.closeButton.Name = "closeButton";

this.closeButton.Size = new System.Drawing.Size(143, 53);

this.closeButton.TabIndex = 6;

this.closeButton.Text = "Закрыть";

this.closeButton.UseVisualStyleBackColor = true;

this.closeButton.Click += new System.EventHandler(this.CloseWindow);

//

// label6

//

this.label6.AutoSize = true;

this.label6.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.label6.Location = new System.Drawing.Point(413, 309);

this.label6.Name = "label6";

this.label6.Size = new System.Drawing.Size(245, 20);

this.label6.TabIndex = 7;

this.label6.Text = "Ханяк Евгений Дмитриевич";

//

// label7

//

this.label7.AutoSize = true;

this.label7.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.label7.Location = new System.Drawing.Point(413, 394);

this.label7.Name = "label7";

this.label7.Size = new System.Drawing.Size(259, 20);

this.label7.TabIndex = 9;

this.label7.Text = "Гурский Николай Николаевич";

//

// label8

//

this.label8.AutoSize = true;

this.label8.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.label8.Location = new System.Drawing.Point(413, 365);

this.label8.Name = "label8";

this.label8.Size = new System.Drawing.Size(303, 20);

this.label8.TabIndex = 8;

this.label8.Text = "Преподаватель: канд. т.н., доцент";

//

// pictureBox1

//

this.pictureBox1.Image = ((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));

this.pictureBox1.Location = new System.Drawing.Point(125, 245);

this.pictureBox1.Name = "pictureBox1";

this.pictureBox1.Size = new System.Drawing.Size(217, 218);

this.pictureBox1.SizeMode = System.Windows.Forms.PictureBoxSizeMode.Zoom;

this.pictureBox1.TabIndex = 10;

this.pictureBox1.TabStop = false;

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.Color.White;

this.ClientSize = new System.Drawing.Size(813, 638);

this.Controls.Add(this.pictureBox1);

this.Controls.Add(this.label7);

this.Controls.Add(this.label8);

this.Controls.Add(this.label6);

this.Controls.Add(this.closeButton);

this.Controls.Add(this.continueButton);

this.Controls.Add(this.label5);

this.Controls.Add(this.label4);

this.Controls.Add(this.label3);

this.Controls.Add(this.label2);

this.Controls.Add(this.label1);

this.Name = "Form1";

this.Text = "Курсовой проект";

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Label label1;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.Label label4;

private System.Windows.Forms.Label label5;

private System.Windows.Forms.Button continueButton;

private System.Windows.Forms.Button closeButton;

private System.Windows.Forms.Label label6;

private System.Windows.Forms.Label label7;

private System.Windows.Forms.Label label8;

private System.Windows.Forms.PictureBox pictureBox1;

}

}

**Form1.cs**

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Course\_project

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void CloseWindow(object sender, EventArgs e)

{

this.Close();

}

private void ContinueUse(object sender, EventArgs e)

{

MainForm mainForm = new MainForm();

mainForm.ShowDialog();

this.Close();

}

}

}

MainForm.Designer.cs

namespace Course\_project

{

partial class MainForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.label1 = new System.Windows.Forms.Label();

this.label2 = new System.Windows.Forms.Label();

this.label3 = new System.Windows.Forms.Label();

this.distanceTextBox = new System.Windows.Forms.TextBox();

this.spacingFactorTextBox = new System.Windows.Forms.TextBox();

this.recursionDepthTextBox = new System.Windows.Forms.TextBox();

this.menuStrip1 = new System.Windows.Forms.MenuStrip();

this.видToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.lightTheme = new System.Windows.Forms.ToolStripMenuItem();

this.darkTheme = new System.Windows.Forms.ToolStripMenuItem();

this.сохранитьToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.сохранитьМодельВBlenderToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.сохранитьМодельВBlenderToolStripMenuItem1 = new System.Windows.Forms.ToolStripMenuItem();

this.rotateButton = new System.Windows.Forms.ToolStripMenuItem();

this.справкаToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.menuStrip1.SuspendLayout();

this.SuspendLayout();

//

// label1

//

this.label1.AutoSize = true;

this.label1.BackColor = System.Drawing.Color.Black;

this.label1.ForeColor = System.Drawing.Color.White;

this.label1.Location = new System.Drawing.Point(12, 44);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(158, 16);

this.label1.TabIndex = 0;

this.label1.Text = "Расстояние до камеры:";

//

// label2

//

this.label2.AutoSize = true;

this.label2.BackColor = System.Drawing.Color.Black;

this.label2.ForeColor = System.Drawing.Color.White;

this.label2.Location = new System.Drawing.Point(12, 82);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(183, 16);

this.label2.TabIndex = 1;

this.label2.Text = "Расстояние между кубами:";

//

// label3

//

this.label3.AutoSize = true;

this.label3.BackColor = System.Drawing.Color.Black;

this.label3.ForeColor = System.Drawing.Color.White;

this.label3.Location = new System.Drawing.Point(12, 122);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(130, 16);

this.label3.TabIndex = 2;

this.label3.Text = "Глубина рекурсии:";

//

// distanceTextBox

//

this.distanceTextBox.BackColor = System.Drawing.Color.Black;

this.distanceTextBox.ForeColor = System.Drawing.Color.White;

this.distanceTextBox.Location = new System.Drawing.Point(176, 41);

this.distanceTextBox.Name = "distanceTextBox";

this.distanceTextBox.Size = new System.Drawing.Size(100, 22);

this.distanceTextBox.TabIndex = 8;

this.distanceTextBox.Text = "10";

this.distanceTextBox.TextChanged += new System.EventHandler(this.ChangedDistance);

//

// spacingFactorTextBox

//

this.spacingFactorTextBox.BackColor = System.Drawing.Color.Black;

this.spacingFactorTextBox.ForeColor = System.Drawing.Color.White;

this.spacingFactorTextBox.Location = new System.Drawing.Point(201, 79);

this.spacingFactorTextBox.Name = "spacingFactorTextBox";

this.spacingFactorTextBox.Size = new System.Drawing.Size(100, 22);

this.spacingFactorTextBox.TabIndex = 9;

this.spacingFactorTextBox.Text = "1";

this.spacingFactorTextBox.TextChanged += new System.EventHandler(this.ChangedSpacingFactor);

//

// recursionDepthTextBox

//

this.recursionDepthTextBox.BackColor = System.Drawing.Color.Black;

this.recursionDepthTextBox.ForeColor = System.Drawing.Color.White;

this.recursionDepthTextBox.Location = new System.Drawing.Point(148, 119);

this.recursionDepthTextBox.Name = "recursionDepthTextBox";

this.recursionDepthTextBox.Size = new System.Drawing.Size(100, 22);

this.recursionDepthTextBox.TabIndex = 10;

this.recursionDepthTextBox.Text = "4";

this.recursionDepthTextBox.TextChanged += new System.EventHandler(this.ChangedRecursionDepth);

//

// menuStrip1

//

this.menuStrip1.ImageScalingSize = new System.Drawing.Size(20, 20);

this.menuStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.видToolStripMenuItem,

this.сохранитьToolStripMenuItem,

this.rotateButton,

this.справкаToolStripMenuItem});

this.menuStrip1.Location = new System.Drawing.Point(0, 0);

this.menuStrip1.Name = "menuStrip1";

this.menuStrip1.Size = new System.Drawing.Size(800, 28);

this.menuStrip1.TabIndex = 11;

this.menuStrip1.Text = "menuStrip1";

//

// видToolStripMenuItem

//

this.видToolStripMenuItem.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.lightTheme,

this.darkTheme});

this.видToolStripMenuItem.Name = "видToolStripMenuItem";

this.видToolStripMenuItem.Size = new System.Drawing.Size(49, 26);

this.видToolStripMenuItem.Text = "Вид";

//

// lightTheme

//

this.lightTheme.Name = "lightTheme";

this.lightTheme.Size = new System.Drawing.Size(184, 26);

this.lightTheme.Text = "Светлая тема";

this.lightTheme.Click += new System.EventHandler(this.SetLightTheme);

//

// darkTheme

//

this.darkTheme.Checked = true;

this.darkTheme.CheckState = System.Windows.Forms.CheckState.Checked;

this.darkTheme.Name = "darkTheme";

this.darkTheme.Size = new System.Drawing.Size(184, 26);

this.darkTheme.Text = "Темная тема";

this.darkTheme.Click += new System.EventHandler(this.SetDarkTheme);

//

// сохранитьToolStripMenuItem

//

this.сохранитьToolStripMenuItem.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.сохранитьМодельВBlenderToolStripMenuItem,

this.сохранитьМодельВBlenderToolStripMenuItem1});

this.сохранитьToolStripMenuItem.Name = "сохранитьToolStripMenuItem";

this.сохранитьToolStripMenuItem.Size = new System.Drawing.Size(97, 26);

this.сохранитьToolStripMenuItem.Text = "Сохранить";

//

// сохранитьМодельВBlenderToolStripMenuItem

//

this.сохранитьМодельВBlenderToolStripMenuItem.Name = "сохранитьМодельВBlenderToolStripMenuItem";

this.сохранитьМодельВBlenderToolStripMenuItem.Size = new System.Drawing.Size(289, 26);

this.сохранитьМодельВBlenderToolStripMenuItem.Text = "Сохранить модель obj";

this.сохранитьМодельВBlenderToolStripMenuItem.Click += new System.EventHandler(this.SaveModelClick);

//

// сохранитьМодельВBlenderToolStripMenuItem1

//

this.сохранитьМодельВBlenderToolStripMenuItem1.Name = "сохранитьМодельВBlenderToolStripMenuItem1";

this.сохранитьМодельВBlenderToolStripMenuItem1.Size = new System.Drawing.Size(289, 26);

this.сохранитьМодельВBlenderToolStripMenuItem1.Text = "Сохранить модель в Blender";

this.сохранитьМодельВBlenderToolStripMenuItem1.Click += new System.EventHandler(this.SaveBlenderClick);

//

// rotateButton

//

this.rotateButton.Font = new System.Drawing.Font("Segoe UI", 9F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.rotateButton.Name = "rotateButton";

this.rotateButton.Size = new System.Drawing.Size(98, 26);

this.rotateButton.Text = "Вращение";

this.rotateButton.Click += new System.EventHandler(this.RotateClick);

//

// справкаToolStripMenuItem

//

this.справкаToolStripMenuItem.Name = "справкаToolStripMenuItem";

this.справкаToolStripMenuItem.Size = new System.Drawing.Size(81, 26);

this.справкаToolStripMenuItem.Text = "Справка";

this.справкаToolStripMenuItem.Click += new System.EventHandler(this.OpenHelpFile);

//

// MainForm

//

this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(800, 800);

this.Controls.Add(this.recursionDepthTextBox);

this.Controls.Add(this.spacingFactorTextBox);

this.Controls.Add(this.distanceTextBox);

this.Controls.Add(this.label3);

this.Controls.Add(this.label2);

this.Controls.Add(this.label1);

this.Controls.Add(this.menuStrip1);

this.MainMenuStrip = this.menuStrip1;

this.Name = "MainForm";

this.Text = "MainForm";

this.menuStrip1.ResumeLayout(false);

this.menuStrip1.PerformLayout();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Label label1;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.TextBox distanceTextBox;

private System.Windows.Forms.TextBox spacingFactorTextBox;

private System.Windows.Forms.TextBox recursionDepthTextBox;

private System.Windows.Forms.MenuStrip menuStrip1;

private System.Windows.Forms.ToolStripMenuItem видToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem lightTheme;

private System.Windows.Forms.ToolStripMenuItem darkTheme;

private System.Windows.Forms.ToolStripMenuItem сохранитьToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem сохранитьМодельВBlenderToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem сохранитьМодельВBlenderToolStripMenuItem1;

private System.Windows.Forms.ToolStripMenuItem rotateButton;

private System.Windows.Forms.ToolStripMenuItem справкаToolStripMenuItem;

}

}

MainForm.cs

using System;

using System.Diagnostics;

using System.Drawing;

using System.IO;

using System.Windows.Forms;

using OpenTK;

using OpenTK.Graphics;

using OpenTK.Graphics.OpenGL;

using Word = Microsoft.Office.Interop.Word;

/// <summary>

/// Главная форма приложения для рендеринга 3D-фракталов, предоставляющая возможности настройки, сохранения моделей и управления темами.

/// </summary>

namespace Course\_project

{

public partial class MainForm : Form

{

/// <summary>

/// Перечисление тем оформления.

/// </summary>

private enum Theme

{

Light,

Dark

}

/// <summary>

/// Текущая тема оформления.

/// </summary>

private Theme currentTheme = Theme.Dark;

private Color primaryColor = Color.Black;

private Color secondaryColor = Color.White;

private GLControl glControl;

private FractalRenderer fractalRenderer;

private ModelSaver modelSaver;

private ControlEventHandler controlEventHandler;

private Timer rotationTimer;

private bool isRotating = false;

/// <summary>

/// Конструктор главной формы.

/// </summary>

public MainForm()

{

InitializeComponent();

this.Text = "3D Fractal";

this.Size = new Size(800, 800);

glControl = new GLControl(new GraphicsMode(32, 24, 0, 4));

glControl.Dock = DockStyle.Fill;

this.Controls.Add(glControl);

fractalRenderer = new FractalRenderer(glControl);

modelSaver = new ModelSaver(fractalRenderer);

controlEventHandler = new ControlEventHandler(glControl, fractalRenderer);

rotationTimer = new Timer();

rotationTimer.Interval = 3;

rotationTimer.Tick += new EventHandler(RotationTimer\_Tick);

AttachEventHandlers();

}

/// <summary>

/// Обработчик загрузки главной формы.

/// </summary>

private void MainForm\_Load(object sender, EventArgs e)

{

ApplyTheme(); // Применяем тему при загрузке формы

}

/// <summary>

/// Присоединяет обработчики событий.

/// </summary>

private void AttachEventHandlers()

{

this.Load += MainForm\_Load;

glControl.Paint += fractalRenderer.GLControl\_Paint;

glControl.Load += fractalRenderer.GLControl\_Load;

glControl.Resize += fractalRenderer.GLControl\_Resize;

glControl.MouseDown += controlEventHandler.GLControl\_MouseDown;

glControl.MouseMove += controlEventHandler.GLControl\_MouseMove;

glControl.MouseUp += controlEventHandler.GLControl\_MouseUp;

}

/// <summary>

/// Обработчик нажатия кнопки вращения.

/// </summary>

private void RotateClick(object sender, EventArgs e)

{

isRotating = !isRotating;

if (isRotating)

{

rotationTimer.Start();

this.Invoke((MethodInvoker)delegate {

rotateButton.Text = "Стоп";

});

}

else

{

rotationTimer.Stop();

this.Invoke((MethodInvoker)delegate {

rotateButton.Text = "Вращать";

});

}

}

/// <summary>

/// Обработчик нажатия кнопки сохранения модели.

/// </summary>

private void SaveModelClick(object sender, EventArgs e)

{

modelSaver.SaveModelDialog();

}

/// <summary>

/// Обработчик нажатия кнопки сохранения модели для Blender.

/// </summary>

private void SaveBlenderClick(object sender, EventArgs e)

{

modelSaver.SaveBlenderModel();

}

/// <summary>

/// Обработчик нажатия кнопки "Справка".

/// </summary>

private void OpenHelpFile(object sender, EventArgs e)

{

try

{

string helpFilePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Help.chm");

if (File.Exists(helpFilePath))

{

Process.Start(helpFilePath);

}

else

{

MessageBox.Show("Файл помощи не найден: " + helpFilePath);

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при открытии файла помощи: " + ex.Message);

}

}

/// <summary>

/// Обработчик тика таймера вращения.

/// </summary>

private void RotationTimer\_Tick(object sender, EventArgs e)

{

try

{

fractalRenderer.Rotate(1f);

}

catch (Exception ex)

{

MessageBox.Show("Ошибка во время вращения: " + ex.Message);

}

}

/// <summary>

/// Обработчик изменения дистанции.

/// </summary>

private void ChangedDistance(object sender, EventArgs e)

{

if (double.TryParse(((TextBox)sender).Text, out double result))

{

fractalRenderer.Distance = result;

glControl.Invalidate();

}

}

/// <summary>

/// Обработчик изменения коэффициента расстояния.

/// </summary>

private void ChangedSpacingFactor(object sender, EventArgs e)

{

if (double.TryParse(((TextBox)sender).Text, out double result))

{

fractalRenderer.SpacingFactor = result;

glControl.Invalidate();

}

}

/// <summary>

/// Обработчик изменения глубины рекурсии.

/// </summary>

private void ChangedRecursionDepth(object sender, EventArgs e)

{

if (int.TryParse(((TextBox)sender).Text, out int result))

{

if (result >= 8)

{

result = 7;

recursionDepthTextBox.Text = "7";

}

fractalRenderer.RecursionDepth = result;

glControl.Invalidate();

}

}

/// <summary>

/// Обработчик установки светлой темы.

/// </summary>

private void SetLightTheme(object sender, EventArgs e)

{

lightTheme.Checked = true;

darkTheme.Checked = false;

currentTheme = Theme.Light;

ApplyTheme();

}

/// <summary>

/// Обработчик установки темной темы.

/// </summary>

private void SetDarkTheme(object sender, EventArgs e)

{

lightTheme.Checked = false;

darkTheme.Checked = true;

currentTheme = Theme.Dark;

ApplyTheme();

}

/// <summary>

/// Применяет текущую тему к интерфейсу.

/// </summary>

private void ApplyTheme()

{

if (currentTheme == Theme.Dark)

{

primaryColor = Color.Black;

secondaryColor = Color.White;

fractalRenderer.IsLightTheme = false;

menuStrip1.Renderer = new ToolStripProfessionalRenderer(new Colors());

}

else

{

primaryColor = Color.White;

secondaryColor = Color.Black;

fractalRenderer.IsLightTheme = true;

menuStrip1.Renderer = new ToolStripProfessionalRenderer();

}

this.BackColor = primaryColor;

this.ForeColor = secondaryColor;

foreach (ToolStripMenuItem item in menuStrip1.Items)

{

SetColor(item);

}

foreach (Control control in this.Controls)

{

ApplyThemeToControl(control);

}

fractalRenderer.SetBackgroundColor(primaryColor);

}

/// <summary>

/// Применяет текущую тему к указанному контролу.

/// </summary>

private void ApplyThemeToControl(Control control)

{

control.BackColor = primaryColor;

control.ForeColor = secondaryColor;

foreach (Control child in control.Controls)

{

ApplyThemeToControl(child);

}

}

/// <summary>

/// Устанавливает цвет для элементов меню.

/// </summary>

private void SetColor(ToolStripMenuItem item)

{

item.ForeColor = secondaryColor;

foreach (ToolStripMenuItem it in item.DropDownItems)

{

SetColor(it);

}

}

}

}

FractalRenderer.cs

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

using OpenTK;

using OpenTK.Graphics.OpenGL;

namespace Course\_project

{

public class FractalRenderer

{

private GLControl glControl;

private List<Vector3> vertices = new List<Vector3>();

private List<int[]> faces = new List<int[]>();

public IReadOnlyList<Vector3> Vertices => vertices;

public IReadOnlyList<int[]> Faces => faces;

public float RotationX { get; set; } = 0f;

public float RotationY { get; set; } = 0f;

public double Distance { get; set; } = 10.0;

public double SpacingFactor { get; set; } = 1;

public int RecursionDepth { get; set; } = 4;

public bool IsLightTheme { get; set; } = false;

public FractalRenderer(GLControl glControl)

{

this.glControl = glControl;

}

public void GLControl\_Load(object sender, EventArgs e)

{

try

{

glControl.MakeCurrent();

GL.ClearColor(Color.Black);

SetupViewport();

GL.Enable(EnableCap.DepthTest);

GL.Enable(EnableCap.Multisample);

}

catch (Exception ex)

{

MessageBox.Show("Ошибка инициализации OpenGL: " + ex.Message);

}

}

private void SetupViewport()

{

try

{

int w = glControl.Width;

int h = glControl.Height;

GL.Viewport(0, 0, w, h);

GL.MatrixMode(MatrixMode.Projection);

GL.LoadIdentity();

float aspectRatio = (float)w / h;

float near = 0.1f;

float far = 500f;

float fov = 45f;

float top = (float)Math.Tan(fov \* 0.5 \* Math.PI / 180.0) \* near;

float bottom = -top;

float left = aspectRatio \* bottom;

float right = aspectRatio \* top;

GL.Frustum(left, right, bottom, top, near, far);

GL.MatrixMode(MatrixMode.Modelview);

}

catch (Exception ex)

{

MessageBox.Show("Ошибка настройки вьюпорта: " + ex.Message);

}

}

public void GLControl\_Resize(object sender, EventArgs e)

{

SetupViewport();

}

public void GLControl\_Paint(object sender, PaintEventArgs e)

{

try

{

glControl.MakeCurrent();

GL.Clear(ClearBufferMask.ColorBufferBit | ClearBufferMask.DepthBufferBit);

GL.LoadIdentity();

GL.Translate(0.0, 0.0, -Distance);

GL.Rotate(RotationX, 1, 0, 0);

GL.Rotate(RotationY, 0, 1, 0);

vertices.Clear();

faces.Clear();

DrawFractal(0, 0, 0, 1, RecursionDepth);

glControl.SwapBuffers();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка рендеринга: " + ex.Message);

}

}

private void DrawFractal(double x, double y, double z, double size, int depth)

{

if (depth == 0) return;

Vector3[] cubeVertices = new Vector3[]

{

new Vector3((float)(x - size / 2), (float)(y - size / 2), (float)(z + size / 2)),

new Vector3((float)(x + size / 2), (float)(y - size / 2), (float)(z + size / 2)),

new Vector3((float)(x + size / 2), (float)(y + size / 2), (float)(z + size / 2)),

new Vector3((float)(x - size / 2), (float)(y + size / 2), (float)(z + size / 2)),

new Vector3((float)(x - size / 2), (float)(y - size / 2), (float)(z - size / 2)),

new Vector3((float)(x + size / 2), (float)(y - size / 2), (float)(z - size / 2)),

new Vector3((float)(x + size / 2), (float)(y + size / 2), (float)(z - size / 2)),

new Vector3((float)(x - size / 2), (float)(y + size / 2), (float)(z - size / 2))

};

int baseIndex = vertices.Count;

vertices.AddRange(cubeVertices);

int[][] cubeFaces = new int[][]

{

new int[] { baseIndex, baseIndex + 1, baseIndex + 2, baseIndex + 3 },

new int[] { baseIndex + 4, baseIndex + 5, baseIndex + 6, baseIndex + 7 },

new int[] { baseIndex, baseIndex + 1, baseIndex + 5, baseIndex + 4 },

new int[] { baseIndex + 2, baseIndex + 3, baseIndex + 7, baseIndex + 6 },

new int[] { baseIndex + 1, baseIndex + 2, baseIndex + 6, baseIndex + 5 },

new int[] { baseIndex, baseIndex + 3, baseIndex + 7, baseIndex + 4 }

};

faces.AddRange(cubeFaces);

GL.Begin(PrimitiveType.Quads);

Color[] faceColors = new Color[] { Color.Blue, Color.Green, Color.Yellow, Color.White, Color.Orange, Color.Red };

if (IsLightTheme)

{

faceColors[3] = Color.Black;

}

for (int i = 0; i < cubeFaces.Length; i++)

{

GL.Color3(faceColors[i]);

foreach (int vertexIndex in cubeFaces[i])

{

Vector3 vertex = vertices[vertexIndex];

GL.Vertex3(vertex);

}

}

GL.End();

double newSize = size / 2;

double offset = size \* SpacingFactor;

DrawFractal(x - offset, y, z, newSize, depth - 1);

DrawFractal(x + offset, y, z, newSize, depth - 1);

DrawFractal(x, y - offset, z, newSize, depth - 1);

DrawFractal(x, y + offset, z, newSize, depth - 1);

DrawFractal(x, y, z - offset, newSize, depth - 1);

DrawFractal(x, y, z + offset, newSize, depth - 1);

}

public void Rotate(float angle)

{

RotationY += angle;

if (RotationY > 360.0f) RotationY -= 360.0f;

RotationX = RotationY - 5;

glControl.Invalidate();

}

public void SetBackgroundColor(Color color)

{

try

{

glControl.MakeCurrent();

GL.ClearColor(color);

glControl.Invalidate();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка установки цвета фона: " + ex.Message);

}

}

}

}

ModelSaver.cs

using System.Diagnostics;

using System;

using System.Globalization;

using System.IO;

using System.Windows.Forms;

namespace Course\_project

{

public class ModelSaver

{

private FractalRenderer fractalRenderer;

public ModelSaver(FractalRenderer renderer)

{

fractalRenderer = renderer;

}

public void SaveModelDialog()

{

using (SaveFileDialog saveFileDialog = new SaveFileDialog())

{

saveFileDialog.Filter = "OBJ files (\*.obj)|\*.obj|All files (\*.\*)|\*.\*";

saveFileDialog.DefaultExt = "obj";

saveFileDialog.AddExtension = true;

saveFileDialog.FileName = "fractal.obj"; // Предложенное имя файла по умолчанию

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

string selectedFilePath = saveFileDialog.FileName;

SaveModelToFile(selectedFilePath);

MessageBox.Show($"Модель сохранена в файл {selectedFilePath}");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка сохранения модели: " + ex.Message);

}

}

}

}

public void SaveBlenderModel()

{

try

{

string objFilePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "fractalForBlender.obj");

SaveModelToFile(objFilePath);

string pythonScriptPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "script.py");

string blenderPath = @"C:\Program Files\Blender Foundation\Blender 4.1\blender.exe"; // Укажите правильный путь к Blender

string arguments = $"-b --python \"{pythonScriptPath}\"";

ProcessStartInfo startInfo = new ProcessStartInfo

{

FileName = blenderPath,

Arguments = arguments,

UseShellExecute = false,

RedirectStandardOutput = true,

RedirectStandardError = true,

CreateNoWindow = true

};

using (Process process = Process.Start(startInfo))

{

string output = process.StandardOutput.ReadToEnd();

string error = process.StandardError.ReadToEnd();

process.WaitForExit();

if (process.ExitCode == 0)

{

MessageBox.Show("Blender завершил выполнение скрипта успешно.");

SaveBlenderProject();

}

else

{

MessageBox.Show("Ошибка при выполнении скрипта в Blender: " + error);

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при запуске Blender: " + ex.Message);

}

}

private void SaveBlenderProject()

{

string sourceBlendFilePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "my\_project.blend");

using (SaveFileDialog saveFileDialog = new SaveFileDialog())

{

saveFileDialog.Filter = "Blender files (\*.blend)|\*.blend|All files (\*.\*)|\*.\*";

saveFileDialog.DefaultExt = "blend";

saveFileDialog.AddExtension = true;

saveFileDialog.FileName = "my\_project.blend"; // Предложенное имя файла по умолчанию

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

string destinationBlendFilePath = saveFileDialog.FileName;

File.Copy(sourceBlendFilePath, destinationBlendFilePath, true);

File.Delete(sourceBlendFilePath);

MessageBox.Show($"Проект Blender сохранен в файл {destinationBlendFilePath}");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка сохранения проекта Blender: " + ex.Message);

}

}

}

}

/// <summary>

/// Сохраняет модель в файл.

/// </summary>

/// <param name="filename">Имя файла для сохранения модели.</param>

public void SaveModelToFile(string filename)

{

using (StreamWriter writer = new StreamWriter(filename))

{

// Запись вершин

foreach (var vertex in fractalRenderer.Vertices)

{

writer.WriteLine($"v {vertex.X.ToString(CultureInfo.InvariantCulture)} {vertex.Y.ToString(CultureInfo.InvariantCulture)} {vertex.Z.ToString(CultureInfo.InvariantCulture)}");

}

// Запись граней

foreach (var face in fractalRenderer.Faces)

{

writer.WriteLine($"f {face[0] + 1}//{face[0] + 1} {face[1] + 1}//{face[1] + 1} {face[2] + 1}//{face[2] + 1} {face[3] + 1}//{face[3] + 1}");

}

}

}

}

}

ControlEventsHandler  
using System.Drawing;

using System.Windows.Forms;

using OpenTK;

namespace Course\_project

{

/// <summary>

/// Обрабатывает события управления для GLControl.

/// </summary>

public class ControlEventHandler

{

private GLControl glControl;

private FractalRenderer fractalRenderer;

private bool isMouseDown = false;

private Point lastMousePos = new Point();

/// <summary>

/// Инициализирует новый экземпляр класса ControlEventHandler.

/// </summary>

/// <param name="glControl">GLControl для обработки событий.</param>

/// <param name="fractalRenderer">Рендерер фрактала.</param>

public ControlEventHandler(GLControl glControl, FractalRenderer fractalRenderer)

{

this.glControl = glControl;

this.fractalRenderer = fractalRenderer;

}

/// <summary>

/// Обрабатывает событие нажатия кнопки мыши для GLControl.

/// </summary>

public void GLControl\_MouseDown(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

isMouseDown = true;

lastMousePos = e.Location;

}

}

/// <summary>

/// Обрабатывает событие перемещения мыши для GLControl.

/// </summary>

public void GLControl\_MouseMove(object sender, MouseEventArgs e)

{

if (isMouseDown)

{

float deltaX = e.X - lastMousePos.X;

float deltaY = e.Y - lastMousePos.Y;

fractalRenderer.RotationX += deltaY \* 0.5f;

fractalRenderer.RotationY += deltaX \* 0.5f;

lastMousePos = e.Location;

glControl.Invalidate();

}

}

/// <summary>

/// Обрабатывает событие отпускания кнопки мыши для GLControl.

/// </summary>

public void GLControl\_MouseUp(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

isMouseDown = false;

}

}

}

}

MenuStrip.cs

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Course\_project

{

public class Colors : ProfessionalColorTable

{

public override Color MenuItemSelected

{

get { return Color.FromArgb(51, 153, 255); }

}

public override Color ToolStripDropDownBackground

{

get { return Color.Black; }

}

public override Color ImageMarginGradientBegin

{

get { return Color.Black; }

}

public override Color ImageMarginGradientEnd

{

get { return Color.Black; }

}

public override Color ImageMarginGradientMiddle

{

get { return Color.Black; }

}

public override Color MenuItemSelectedGradientBegin

{

get { return Color.FromArgb(51, 153, 255); }

}

public override Color MenuItemSelectedGradientEnd

{

get { return Color.FromArgb(51, 153, 255); }

}

public override Color MenuItemPressedGradientBegin

{

get { return Color.FromArgb(51, 153, 255); }

}

public override Color MenuItemPressedGradientMiddle

{

get { return Color.FromArgb(51, 153, 255); }

}

public override Color MenuItemPressedGradientEnd

{

get { return Color.FromArgb(51, 153, 255); }

}

public override Color MenuItemBorder

{

get { return Color.FromArgb(51, 153, 255); }

}

}

}

Script.py

import bpy

# Удаляем все объекты в сцене

bpy.ops.object.select\_all(action='SELECT')

bpy.ops.object.delete()

# Импортируем объект из файла

bpy.ops.wm.obj\_import(filepath="fractalForBlender.obj")

# Сохраняем проект

bpy.ops.wm.save\_as\_mainfile(filepath="my\_project.blend")