Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №1**

**«Исследование фракталов»**

**ПО ДИСЦИПЛИНЕ «МДК.05.02»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Вершинин Григорий Сергеевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

**Цель работы:**

Получение навыков реализации алгоритмов с рекурсивными вычислениями, знакомство с фракталами.

**Формулировка задания:**

* Написать программу для визуализации фрактала «Кривая Гильберта».
* Предусмотреть возможность масштабирования, изменения глубины прорисовки и перемещения полученной фигуры.
* Построение множества ломанных, образующих фрактал, должно осуществляться в отдельном модуле.

**Описание алгоритма и ответы на вопросы:**

Алгоритм, реализованный в модуле HilbertCurve, строит **кривую Гильберта** — фрактальную линию, которая заполняет пространство (квадрат) с помощью рекурсивного разбиения. Вот краткое описание алгоритма: на каждом шаге квадрат разбивается на 4 меньших квадрата. В каждом из этих квадратов строится часть кривой Гильберта. Для правильного соединения частей кривой используются повороты и отражения. Это достигается за счёт изменения параметров Xi, Xj, Yi, Yj, которые определяют направление и масштаб. В процессе рекурсии координаты точек записываются в массив Points. В основной программе реализована работа кнопок: Q, W, E, R, T для изменения глубины, стрелки для перемещения фигуры, «+» и «-» для изменения масштаба.

**Схемы алгоритмов:**

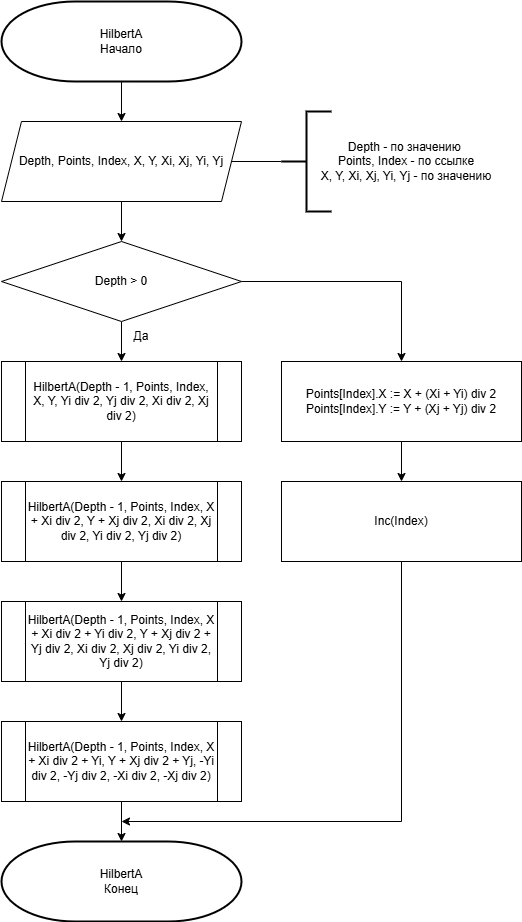


Рисунок 1 – схема алгоритма процедуры HilbertA в модуле HilbertCurve

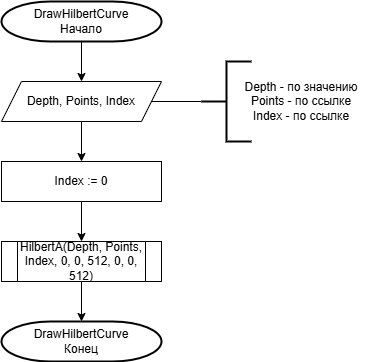


Рисунок 2 – схема алгоритма процедуры DrawHilbertCurve в модуле HilbertCurve

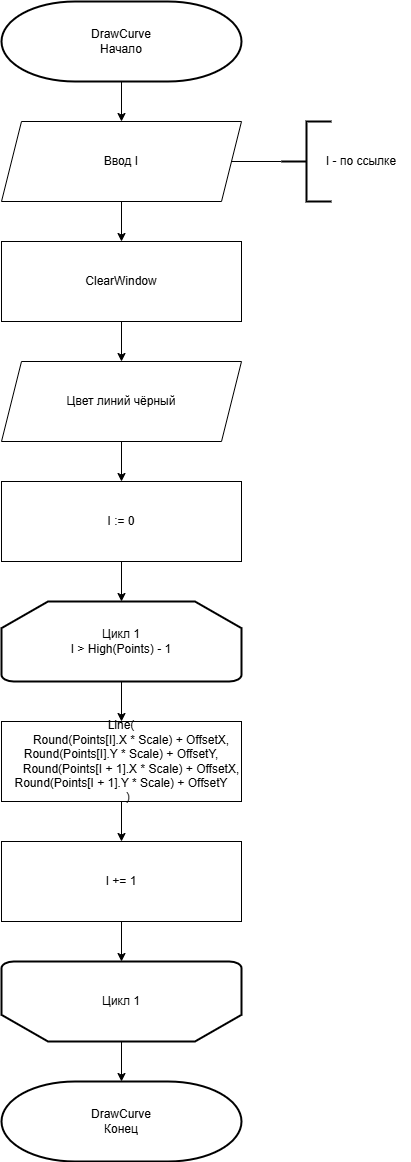


Рисунок 3 – схема алгоритма процедуры DrawCurve основной программы

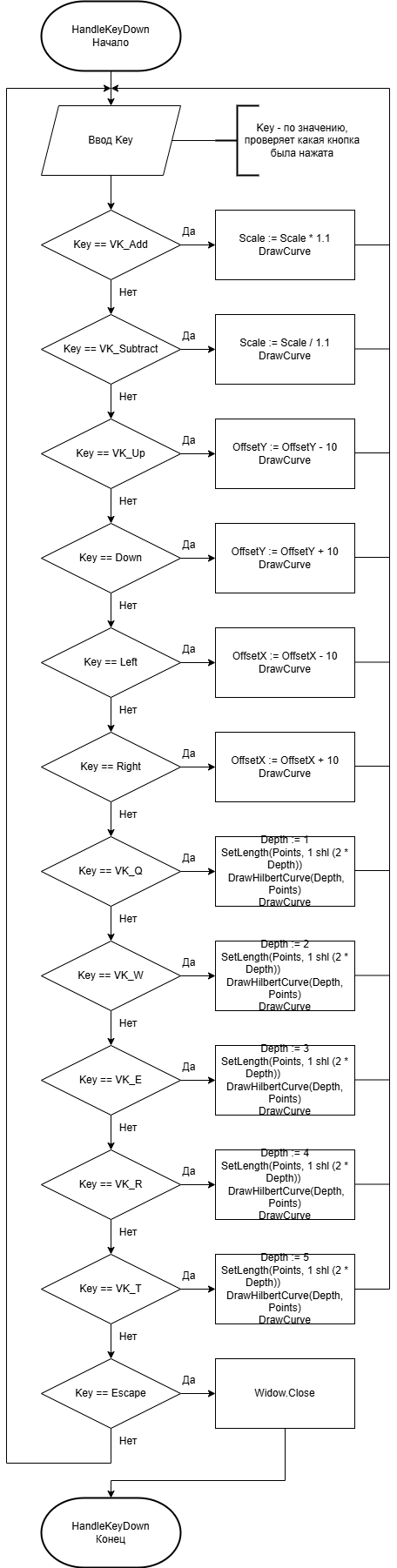


Рисунок 4 – схема алгоритма процедуры HandleKeyDown

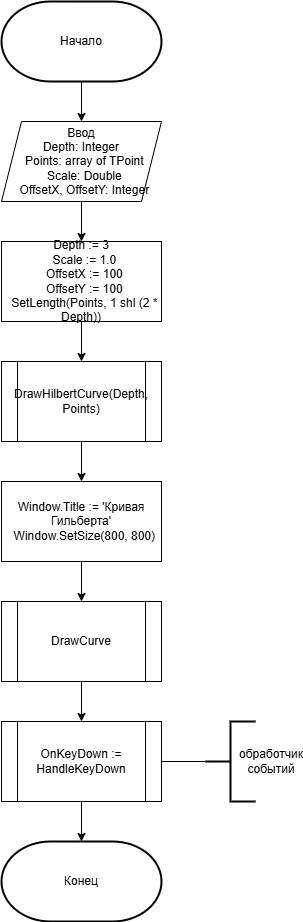


Рисунок 5 – схема алгоритма основной программы

**Код программы:**

**uses** GraphABC, HilbertCurve;

**var**

Depth: Integer;

Points: **array of** TPoint;

Scale: Double;

OffsetX, OffsetY: Integer;

**procedure** DrawCurve;

**var**

I: Integer;

**begin**

ClearWindow;

SetPenColor(clBlack);

**for** I := 0 **to** High(Points) - 1 **do**

**begin**

Line(

Round(Points[I].X \* Scale) + OffsetX, Round(Points[I].Y \* Scale) + OffsetY,

Round(Points[I + 1].X \* Scale) + OffsetX, Round(Points[I + 1].Y \* Scale) + OffsetY

);

**end**;

**end**;

**procedure** HandleKeyDown(Key: Integer);

**begin**

**case** Key **of**

VK\_Add: **begin**

Scale := Scale \* 1.1;

DrawCurve;

**end**;

VK\_Subtract: **begin**

Scale := Scale / 1.1;

DrawCurve;

**end**;

VK\_Up: **begin**

OffsetY := OffsetY - 10;

DrawCurve;

**end**;

VK\_Down: **begin**

OffsetY := OffsetY + 10;

DrawCurve;

**end**;

VK\_Left: **begin**

OffsetX := OffsetX - 10;

DrawCurve;

**end**;

VK\_Right: **begin**

OffsetX := OffsetX + 10;

DrawCurve;

**end**;

VK\_Q: **begin**

Depth := 1;

SetLength(Points, 1 **shl** (2 \* Depth));

DrawHilbertCurve(Depth, Points);

DrawCurve;

**end**;

VK\_W: **begin**

Depth := 2;

SetLength(Points, 1 **shl** (2 \* Depth));

DrawHilbertCurve(Depth, Points);

DrawCurve;

**end**;

VK\_E: **begin**

Depth := 3;

SetLength(Points, 1 **shl** (2 \* Depth));

DrawHilbertCurve(Depth, Points);

DrawCurve;

**end**;

VK\_R: **begin**

Depth := 4;

SetLength(Points, 1 **shl** (2 \* Depth));

DrawHilbertCurve(Depth, Points);

DrawCurve;

**end**;

VK\_T: **begin**

Depth := 5;

SetLength(Points, 1 **shl** (2 \* Depth));

DrawHilbertCurve(Depth, Points);

DrawCurve;

**end**;

VK\_Escape: Window.Close; // Выход по нажатию Esc

**end**;

**end**;

**begin**

Depth := 3;

Scale := 1.0;

OffsetX := 100;

OffsetY := 100;

SetLength(Points, 1 **shl** (2 \* Depth));

DrawHilbertCurve(Depth, Points);

Window.Title := 'Кривая Гильберта';

Window.SetSize(800, 800);

DrawCurve;

// Назначение обработчика события OnKeyDown

OnKeyDown := HandleKeyDown;

**end**.

**unit** HilbertCurve;

**type**

TPoint = **record**

X, Y: Integer;

**end**;

**procedure** HilbertA(Depth: Integer; **var** Points: **array of** TPoint; **var** Index: Integer; X, Y, Xi, Xj, Yi, Yj: Integer);

**begin**

**if** Depth > 0 **then**

**begin**

HilbertA(Depth - 1, Points, Index, X, Y, Yi **div** 2, Yj **div** 2, Xi **div** 2, Xj **div** 2);

HilbertA(Depth - 1, Points, Index, X + Xi **div** 2, Y + Xj **div** 2, Xi **div** 2, Xj **div** 2, Yi **div** 2, Yj **div** 2);

HilbertA(Depth - 1, Points, Index, X + Xi **div** 2 + Yi **div** 2, Y + Xj **div** 2 + Yj **div** 2, Xi **div** 2, Xj **div** 2, Yi **div** 2, Yj **div** 2);

HilbertA(Depth - 1, Points, Index, X + Xi **div** 2 + Yi, Y + Xj **div** 2 + Yj, -Yi **div** 2, -Yj **div** 2, -Xi **div** 2, -Xj **div** 2);

**end**

**else**

**begin**

Points[Index].X := X + (Xi + Yi) **div** 2;

Points[Index].Y := Y + (Xj + Yj) **div** 2;

Inc(Index);

**end**;

**end**;

**procedure** DrawHilbertCurve(Depth: Integer; **var** Points: **array of** TPoint);

**var**

Index: Integer;

**begin**

Index := 0;

HilbertA(Depth, Points, Index, 0, 0, 512, 0, 0, 512);

**end**;

**end**.

**Результат выполнения программы:**

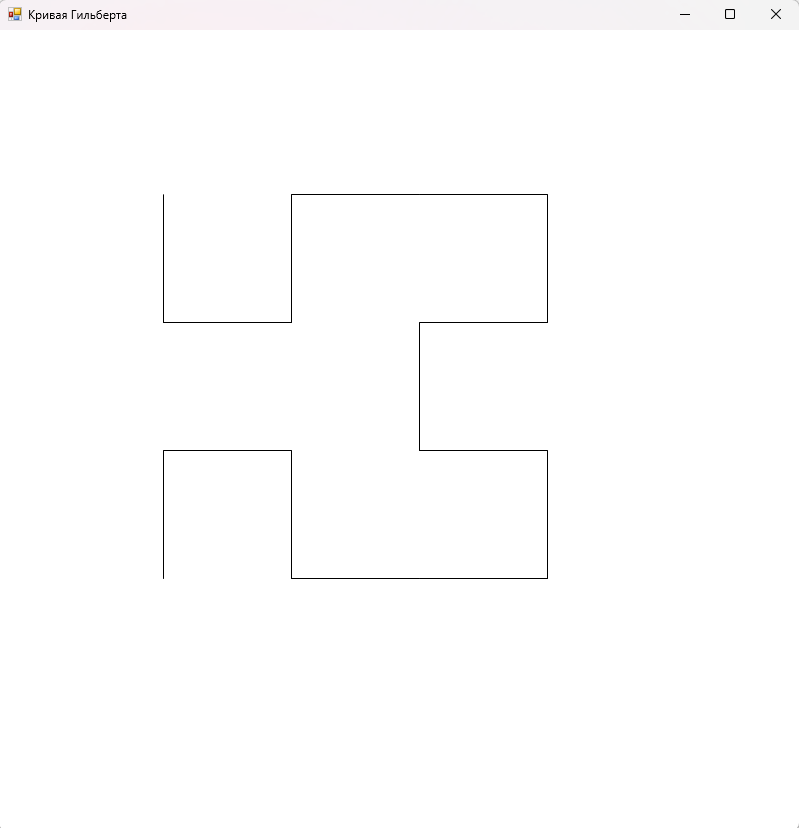
****

Рисунок 6 – результат выполнения работы одной из кнопок

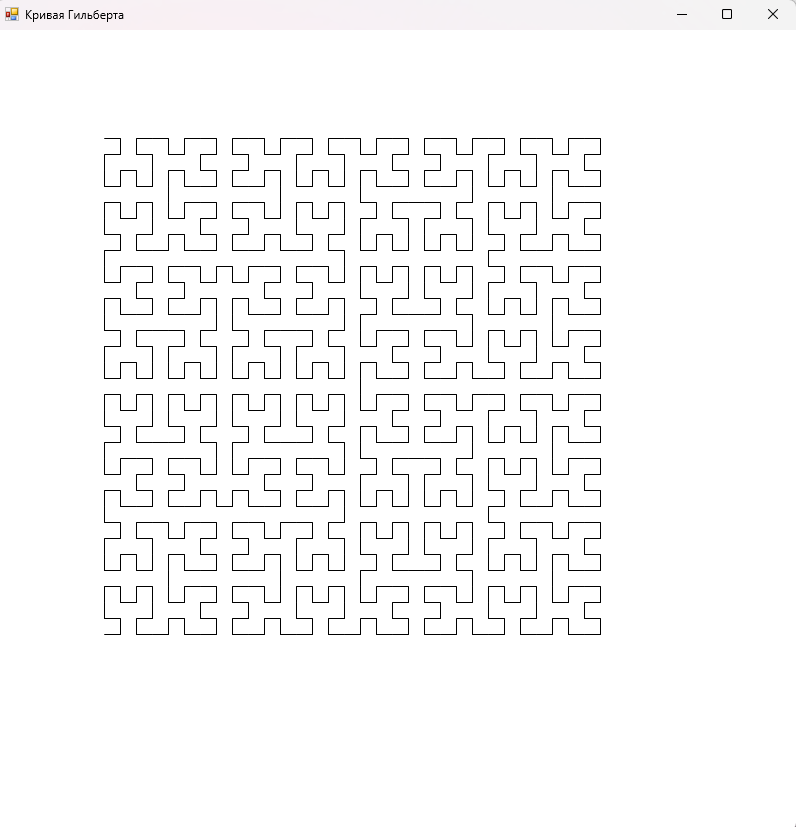


Рисунок 7 – результат выполнения одной из кнопок

**Вывод:**

Изучив правила построения кривой Гильберта, а также теоретический материал про рекурсивные алгоритмы, можно сделать следующий вывод: работа показалась довольно интересной и не очень сложной, также пришлось изучить информацию про модули и подключение работы клавиш. Фракталы довольно интересная тема для изучения и дальнейшего исследования.