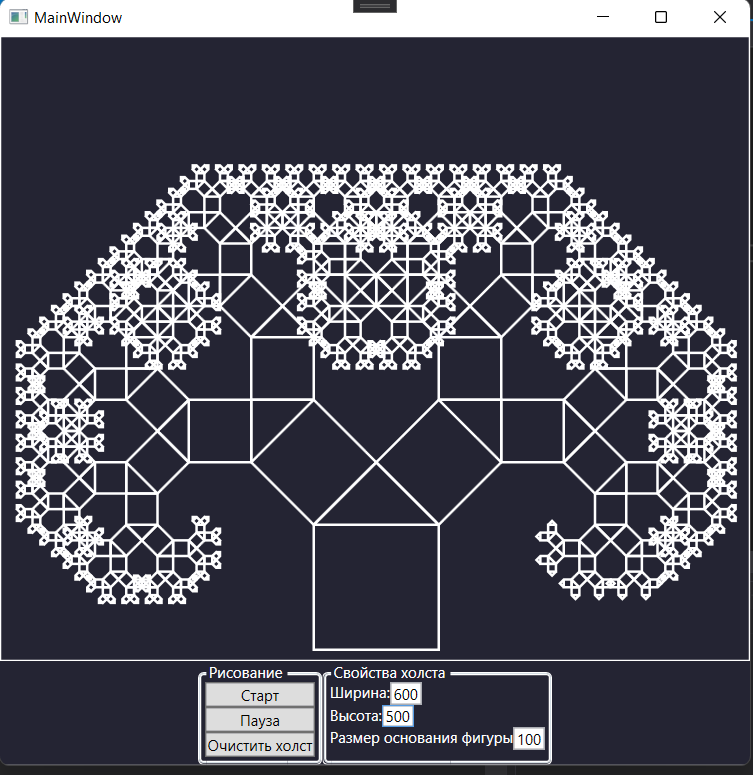
***Вступление***

Фракталы интересная тема, которая не всегда затрагивается в учебном курсе высших учебных заведений поэтому крайне познавательно реализовывать проекты, связанные с фракталами. В качестве одного из таких проектов можно привести разработанный нашей команды проект под названием дерево Пифагора. Проект позволяет посредством несложного и понятного интерфейса рисовать фрактал наблюдать сам процесс рисования останавливать его, а также при необходимости очищать холст. Также пользователь может задать некоторые свойства холста и начальной фигуры, с которой и начинается процесс рисования фрактала.

Сам проект состоит из интерфейсной части (WPF) и Модели. В процессе создания проекта был использован паттерн программирование MVVM.

***Интерфейс***

******

Нажав на кнопку “Старт”, пользователь начнёт процесс рисования фрактала. Фрактал будет изображён в реальном времени постепенно. Код самой кнопки изложен ниже:

При нажатии кнопки “стоп” рисование фрактала останавливается. Происходит это следующим образом:

При нажатии кнопки “очистить холст” полученное ранее изображение удаляется.

***Модель.***

Модель как это не сложно предположить содержит методы позволяющие изобразить фрактал "дерево Пифагора".

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Media;

namespace WpfApp

{

public static class Coordinate

{

/// <summary>

/// Очередь фигур

/// </summary>

static Queue<MyFigure> figures = new();

/// <summary>

/// Добавить в очередь первый элемент

/// </summary>

/// <param name="x">Координата X</param>

/// <param name="y">Координата Y</param>

/// <param name="widht">Размер ребра основания фигуры</param>

/// <returns></returns>

public static void AddToQueue(double x, double y, double widht)

{

figures.Enqueue(new MyFigure(x, y, widht, 0));

}

/// <summary>

/// Очистка очереди

/// </summary>

public static void Clear()

{

figures.Clear();

}

/// <summary>

/// Возвращает фигуру координаты

/// </summary>

public static PointCollection Get()

{

if (figures.Count == 0)

return null;

MyFigure f = figures.Dequeue();

return GetPoints(f);

}

/// <summary>

/// Рассчёт координат для фигуры

/// </summary>

/// <param name="f">Фигура</param>

/// <returns>Координаты вершин</returns>

private static PointCollection GetPoints(MyFigure f)

{

Point point = new(f.X, f.Y);

PointCollection position = new()

{

PointRotate(f.Alfa, new Point(f.X, f.Y - f.Width), point),

PointRotate(f.Alfa, new Point(f.X + f.Width / 2, f.Y - 1.5 \* f.Width), point),

PointRotate(f.Alfa, new Point(f.X + f.Width, f.Y - f.Width), point),

PointRotate(f.Alfa, new Point(f.X + f.Width, f.Y), point),

PointRotate(f.Alfa, new Point(f.X, f.Y), point),

PointRotate(f.Alfa, new Point(f.X, f.Y - f.Width), point),

PointRotate(f.Alfa, new Point(f.X + f.Width, f.Y - f.Width), point)

};

double width = f.Width / Math.Sqrt(2);

figures.Enqueue(new MyFigure(position[0].X, position[0].Y, width, f.Alfa - 45));

figures.Enqueue(new MyFigure(position[1].X, position[1].Y, width, f.Alfa + 45));

position.Freeze();

return position;

}

private static Point PointRotate(double a, Point p, Point p0)

{

double alfa = a \* Math.PI / 180;

double x = p.X - p0.X;

double y = p.Y - p0.Y;

double X = x \* Math.Cos(alfa) - y \* Math.Sin(alfa) + p0.X;

double Y = x \* Math.Sin(alfa) + y \* Math.Cos(alfa) + p0.Y;

return new Point(X, Y);

}

}

}

namespace WpfApp

{

/// <summary>

/// Фигура

/// </summary>

public class MyFigure

{

/// <summary>

/// Начальная координата по X

/// </summary>

public double X { get; set; }

/// <summary>

/// Начальная координата по Y

/// </summary>

public double Y { get; set; }

/// <summary>

/// Длина ребра

/// </summary>

public double Width { get; set; }

/// <summary>

/// Угол поворота

/// </summary>

public double Alfa { get; set; }

/// <summary>

/// Конструктор создания новой фигуры

/// </summary>

/// <param name="x">Координата X</param>

/// <param name="y">Координата Y</param>

/// <param name="width">Начальный размер</param>

/// <param name="alfa">Угол наклона</param>

public MyFigure(double x, double y, double width, double alfa)

{

X = x;

Y = y;

Width = width;

Alfa = alfa;

}

}

}

***Вывод.***

 Разработка проекта "дерево Пифагора" крайне важна по той простой причине что в данном проекте сочетается большое количество различных тем объектно-ориентированного программирования, и, переплетаясь вместе, формируя единое целое, я они дают возможность изучить один из очень распространённых фракталов, понять, как он изображается, я и изучить этот принцип.