|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет экономики, менеджмента и бизнес-информатики* |
|  |
| Целиков Владимир Эдуардович  **РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ВИЗУАЛИЗАЦИИ СЛОЖНОГО ГРАФА**  *Курсовая работа*  студента образовательной программы «Программная инженерия»  по направлению подготовки *09.03.04 Программная инженерия*   |  |  | | --- | --- | |  | Руководитель  Доцент кафедры информационных технологий в бизнесе  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Е. А. Еремин | |

Пермь, 2021 год

# Аннотация

Курсовая работа «Разработка приложения для визуализации сложного графа» подготовлен студентом НИУ-ВШЭ Пермь Целиковым Владимиром Эдуардовичем. В работе описаны этапы разработки информационной системы.

Работа содержит 31 страницу формата А4 основного текста, включая 3 главы: анализ предметной области, проектирование информационной системы и разработка информационной системы.

В основной части содержится 17 картинок, 7 таблиц, описание алгоритма и работы программы. В работе содержится 7 приложений: техническое задание, руководство пользователя, руководство администратора, листинг программы, диаграммы, проектирование интерфейса, программа и методика испытаний. Библиографический список состоит из 6 источников.

# Оглавление

[Аннотация 2](#_Toc67646471)

[Оглавление 3](#_Toc67646472)

[Введение 4](#_Toc67646473)

[Глава 1. Анализ предметной области 6](#_Toc67646474)

[1.1 Анализ назначения системы и требований 6](#_Toc67646475)

[1.2 Анализ существующих аналогов 6](#_Toc67646476)

[1.3 Формирование требований 8](#_Toc67646477)

[Глава 2. Проектирование информационной системы 12](#_Toc67646478)

[2.1 Проектирование интерфейса. 12](#_Toc67646479)

[Глава 3. Разработка информационной системы 14](#_Toc67646480)

[3.1 Описание созданной системы 14](#_Toc67646481)

[3.2 Разработка запросов к данным 15](#_Toc67646482)

[3.3 Тестирование 17](#_Toc67646483)

[Заключение 18](#_Toc67646484)

[Список условных обозначений и сокращений 19](#_Toc67646485)

[Библиографический список 20](#_Toc67646486)

[Приложение А. Техническое задание 21](#_Toc67646487)

[Приложение Б. Диаграммы активностей 28](#_Toc67646488)

[Приложение В. Интерфейсы 29](#_Toc67646489)

[Приложение Г. Листинг 33](#_Toc67646490)

[Приложение Д. Руководство пользователя 37](#_Toc67646491)

[1. Введение 38](#_Toc67646492)

[1.1. Область применения 38](#_Toc67646493)

[1.2. Краткое описание возможностей 38](#_Toc67646494)

[1.3. Уровень подготовки пользователя 38](#_Toc67646495)

[1.4. Перечень эксплуатационной документации, с которой необходимо ознакомиться пользователю 38](#_Toc67646496)

[2. Назначение и условия применения 39](#_Toc67646497)

[2.1. Функциональное назначение программы 39](#_Toc67646498)

[2.2. Условия применения 39](#_Toc67646499)

[3. Подготовка к работе 39](#_Toc67646500)

[3.1. Состав и содержание дистрибутивного носителя данных 39](#_Toc67646501)

[3.2. Порядок загрузки данных и программ 39](#_Toc67646502)

[3.3. Порядок проверки работоспособности 39](#_Toc67646503)

[4. Описание операций 40](#_Toc67646504)

[4.1. Выполняемые функции и задачи 40](#_Toc67646505)

[5. Аварийные ситуации 40](#_Toc67646506)

[6. Рекомендации по освоению 41](#_Toc67646507)

[Приложение Е. Руководство администратора 42](#_Toc67646508)

[Приложение Ж. Программа и методика испытаний 46](#_Toc67646509)

# Введение

Данная работа предназначена для комфортного построения сложных графов разной сложности, что поможет студентам и просто людям, которым нужно визуализировать графы, как для учебы так и для работы.

Люди уже давно начали переходить от рабочих тетрадей к планшетам, телефонам и личным компьютерам, так как это позволяет быть любой информации более доступной, легче исправить свои ошибки и современные инструменты помогают работать с информацией не прибегая к лишним действиям.

Работа является востребованной так как большинство процессов уже давно автоматизированы и визуализация сложных графов не является исключением, но все приложения, которые созданы для этого на данный момент тяжелы в своем использовании и для работы с ними необходима подготовка, так же для из работы нужно довольно производительное устройство, так как их работа предназначена не только визуализации сложных графов, но и для других работ.

Объект исследования – способы визуализации сложных графов.

Цель работы – создание приложения, при помощи которого пользователь сможет визуализировать сложный граф не испытывая при этом сложности в использовании.

Для успешного выполнения работы необходимо выполнить следующие задачи:

1. Анализ теоретических данных
2. Разработка информационной системы.
3. Тестирование системы.
4. Подготовка документации.

Данная сфера не является сильно проработанной, так как приложение имеет специфичное направление и не каждый человек, вообще знает что такое сложный граф. Данное приложение сможет помочь тем людям, которым необходимо визуализировать сложный граф, как для работы так и для учебы.

Будут использованы такие методы как сбор и обработка данных, моделирование, теоретический анализ, визуализация, аналогия и анализ.

План глав:

1. Первая глава будет содержать анализ предметной области. Описание назначения системы и требований, подробный анализ аналогов и на основе этого формирование требований с выбором инструментов для разработки системы.
2. Проектирование информационной системы.
   1. Проектирование интерфейса.
3. Разработка информационной системы.
   1. Описание созданной системы.
   2. Разработка основных функций
   3. Тестирование.

# Глава 1. Анализ предметной области

В этой главе описана предметная область, изучено назначение программного продукта. Так же представлен анализ существующих аналогов, описаны их плюсы и минусы. На основе этого анализа выявлены основные требования к разрабатываемой информационной системе.

## Анализ назначения системы и требований

Построение сложных графов уже на протяжении нескольких десятков лет преподается в университетах по всему миру. В большинстве своем студенты рисуют свои графы от руки, что не всегда может позволить точность и гибкость при дальнейшей работе. Хоть и визуализация сложных графов при помощи программ это уже давно не новшество, но не у всех пользователей есть возможность использовать программы для этого так некоторые из них требуют платных подписок, другие производительные ПК.

В большинстве университетах на факультетах с математическим наклоном студенты проходят сложные графы. И для более простой работы остановимся на создании приложения, которое позволяет построить два основных вида графов: ориентированные и неориентированные.

## Анализ существующих аналогов

Как было сказано в главах раннее люди уже давно используют графы для учебы и работы, так как это довольно простой и доступный способ для представления типов отношений, например дорог соединяющих города, людей и других видов сетей. В основном это отрисованные от руки сложные графы, иногда при помощи приложений для рисования, которые содержат в себе объекты, которые упрощают визуализацию. На данный момент существует множество библиотек для программирования, которые позволяют с легкостью визуализировать графики, но из существующих решений можно выделить только 2: приложение для компьютеров «TM4L» и сайт «Graph online». Разберем оба решения более подробно.

1) Приложение «TM4L» данное приложение довольно тяжело в использовании и предназначено для решения сложных задач, для не продвинутого пользователя уйдет не меньше часа на то чтобы понять, как и что и работает в данном приложении, так как у приложения нет, простого и понятного интерфейса. И само приложение довольно требовательно к производительности устройства, на котором оно запускается. Редактирование графа производится только при помощи переписи его связей.

2) Сайт «Graph online» данный сайт, вобрал в себя много хороших функций и простату в использовании. Возможно несколько путей экспорта нашего графа себе на компьютер, содержит в себе уже несколько алгоритмов, которые упрощают работу пользователя. Но так как данное решение представлено как сайт масштабирование полотна на котором изображены графы не удобно, что портит все, так как работа с большими графами не комфортная. А цель приложения — это удобство работы

| Критерий анализа | Graph online | TM4L |
| --- | --- | --- |
| Авторизация для использования сервиса | - | - |
| Масштабирование полотна | + | + |
| Интуитивный дизайн | + | - |
| Редактирование полотна | + | + |
| Возможность работы с ориентированными и неориентированными графами | + | - |
| Экспорт графов | + | + |

Таблица 1.1. Сравнение аналогов

Из таблицы представленной выше мы можем заметить, что каждое из решений имеет, как свои плюсы так и минусы. И при помощи данных критерий мы сможем сформировать требования, для разрабатываемого приложения.

## Формирование требований

В данном разделе будут описаны требования, которым должно соответствовать приложение и задачи, которые оно должно выполнять.

Данное приложение представляет собой графический визуализатор сложных графов и должно выполнять такие задачи, как: построение графов, экспорт графов, редактирование графов, масштабирование полотна с графами.

Для более подробного рассмотрения выделим функциональные и нефункциональные требования:

Нефункциональные требования:

Интуитивно понятный интерфейс-пользователь не должен задаваться вопросом как ему создать граф, так как функций в данном приложении не много то их можно вынести на главную страницу;

Быстрая работа-под данным требованием подразумевается что отклик в приложении не должен превышать 1 секунду

Функциональные требования:

Создание вершин

Соединение вершин: ориентированное и неориентированное

Редактирование вершин: удаление и перемещение

Экспорт графов в формате .txt

Импорт графов в программу в формате .txt

Вывод координат вершин

На основании выделенных требований была составлена диаграмма прецендентов (Рисунок 1.1). На основе данного анализа было составлено техническое задание (Приложение А). Для работы с данным приложение выявлен только один актор: пользователь. В диаграмме отражены основные функции, которые выполняет приложение

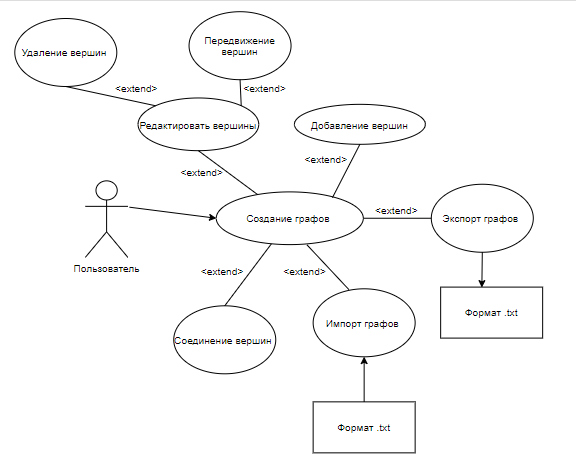


Рисунок 1.1 Диаграмма прецедентов

На основании данной диаграммы прецедентов составим диаграммы активностей для некоторых основных функций. Ниже представлена одна из основных функций — это добавление новых вершин (Рисунок 1.2).

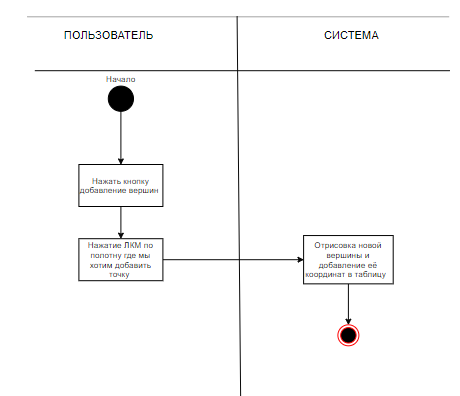


Рисунок 1.2

Следующая функция — это редактирование нашей вершины тут нам доступно несколько видов редактирования: перемещение и удаление (Рисунок 1.3)

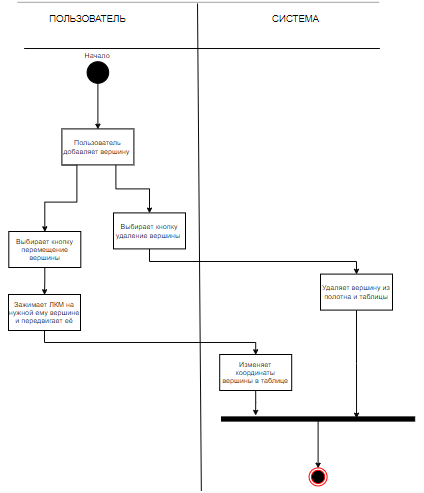


Рисунок 1.3

Следующая ключевая функция — это соединение вершин (Рисунок 1.4)

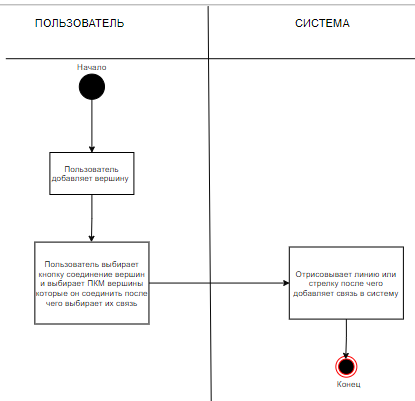


Рисунок 1.4

Последняя функции — это экспорт и импорт графа (Рисунок 1.5)

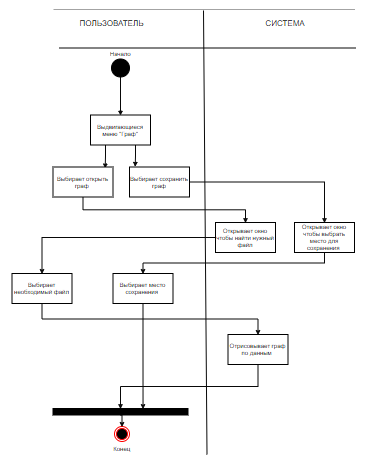


Рисунок 1.5

# Глава 2. Проектирование информационной системы

Вторая глава содержит в себе описание проектирования информационной системы на основе всех требований, выявленных в первой главе анализа предметной области.

## 2.1 Проектирование интерфейса.

Проектирование интерфейса одна из самых важных вещей в данном приложении ведь на простату и интуитивность сделан большой упор. Он должен быть доступным и давать пользователю возможность выполнить все функции программы, которые в ней представлены не тратя лишнее время на разбирательство в кнопках.

Приложение подразумевает всего два окна: основное, котором будет происходить сама визуализация сложного графа и всплывающее, в котором у нас будет спрашивать какую связь между вершинами мы хотим.

В главном окне представлено полотно и все функции, которые может выполнять приложение, также слева располагается таблица с координатами вершин (Рисунок 2.1)

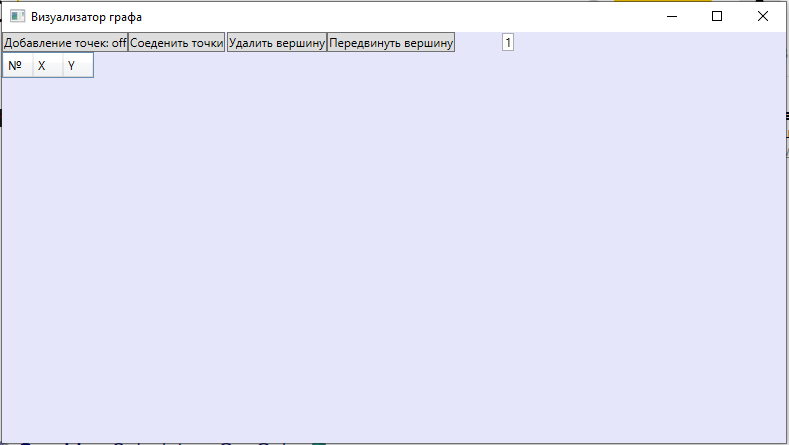


Рисунок 2.1Главное окно

Окно выбора связи между вершинами представляет собой всплывающее окно с двумя кнопками, при помощи которых пользователь должен сделать выбор связи между вершинами, которое он хочет (Рисунок 2.2)

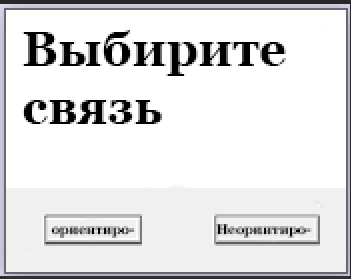


Рисунок 2.2 Окно выбора

# Глава 3. Разработка информационной системы

Заключительная третья глава содержит в себе подробное описание разработки информационной системы и отладку с тестированием.

## Описание созданной системы

Разрабатываемое приложение было написано на платформе WPF, так как при помощи нее можно создать клиентское приложение для настольных систем Windows с привлекательным пользовательским интерфейсом, не затрачивая на это много времени и сил.

Пользовательский интерфейс был написан на языке XAML, так это позволяет не связывать тесно внешний вид приложения и его код поведения.

Основной функциональный код был написан на C#

Используя разработанный интерфейс во второй главе было создано 2 формы:

1. MainWindow – главное меню где пользователь визуализирует свой граф.

2. MessageBox –Форма, в которой пользователь выбирает тип связи

Листинг данных форм представлены в приложении А

## Тестирование

Тестирование разработанной и реализованной информационной системы проходило в рамках реализации системы. В ходе которого было обнаружено множество ошибок в работе с формами. Данное тестирование не задокументировано.

После нахождения каких-либо ошибок в процессе тестирования ПО, происходила отладка и тестирование возобновлялось.

**Тест 1**

Проверка клика по полотну, когда ни одна из функций не активна

**Ожидаемое поведение:**

Ничего не выполниться

**Фактическое поведение:**

Ничего не выполнилось

**Тест 2**

Проверка добавления вершин в одну и туже точку

**Ожидаемое поведение:**

Вторая вершина создастся в ближайшем диапазоне

**Фактическое поведение:**

Вторая вершина создалась в ближайшем диапазоне (Рисунок 3.1)

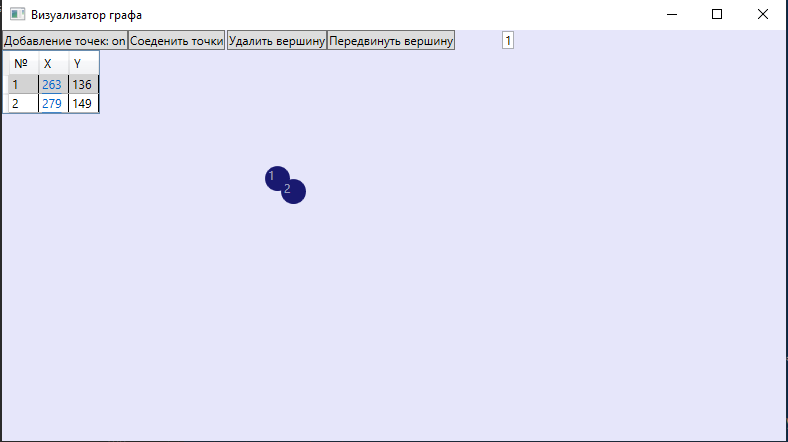


Рисунок 3.1 Добавление второй вершины в существующую

**Тест 3**

Проверка на работу нескольких переключатель одновременно

**Ожидаемое поведение:**

После включения второго первый отключается

**Фактическое поведение:**

После включения второго первый отключается

**Тест 4**

Соединение точек не ориентировано

**Ожидаемое поведение:**

Вершины соединяться прямой

**Фактическое поведение:**

Вершины соединяться прямой (см. Рисунок 3.2)

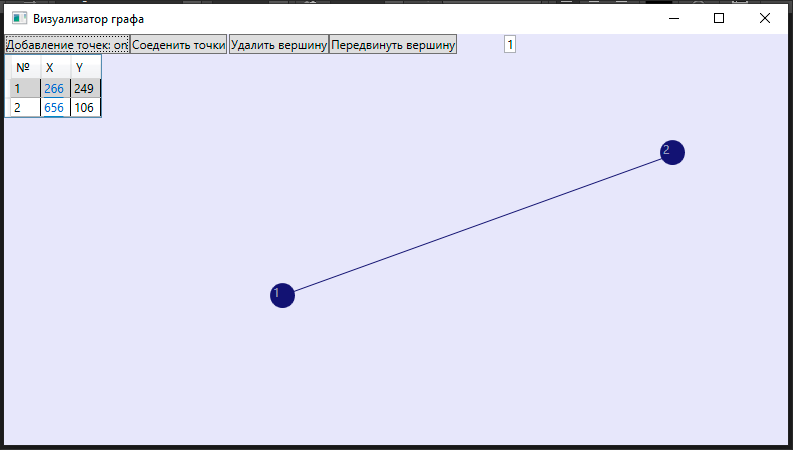


Рисунок 3.2 Соединение вершин не ориентированной прямой

**Тест 5**

Соединение точек ориентировано

**Ожидаемое поведение:**

Вершины соединяться стрелкой

**Фактическое поведение:**

Вершины соединяться стрелкой (см. Рисунок 3.3)

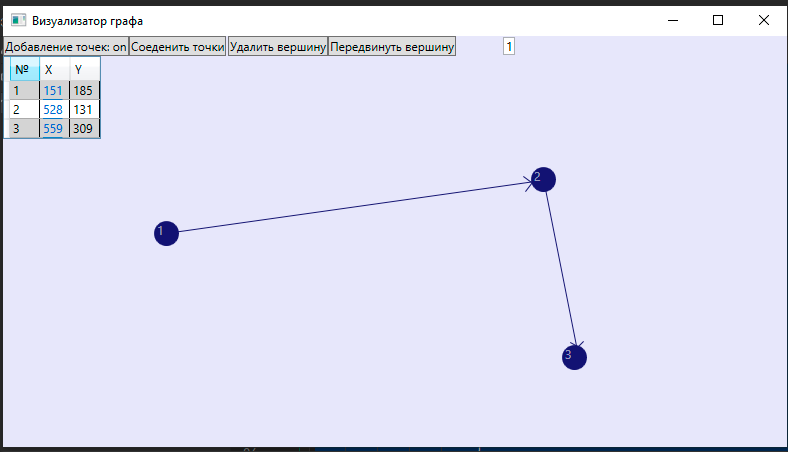


Рисунок 3.3 Соединение вершин ориентированно

**Тест 6**

Удаление точки

**Ожидаемое поведение:**

Вершина удалится и данные о ней в таблице тоже

**Фактическое поведение:**

Вершина удалилась и данные о ней в таблице тоже

**Тест 7**

Удаление точки со связью

**Ожидаемое поведение:**

Вершина удалиться и связи к ней тоже

**Фактическое поведение:**

Вершина удалилась и связи к ней тоже (Рисунок 3.4)

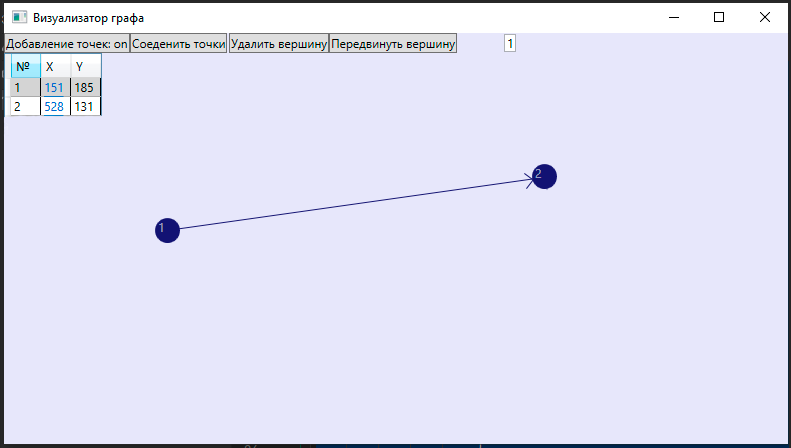


Рисунок 3.4 Удаление вершины со связью

# Заключение

# Приложение А. Листинг

Код XAML:

<Window x:Class="WpfApp1.MainWindow"  
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
 xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  
 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  
 xmlns:local="clr-namespace:WpfApp1"  
 mc:Ignorable="d"  
 Title="Визуализатор графа" Height="450" Width="800">  
 <Grid MouseDown="Grid\_MouseDown">  
   
 <Canvas Name="Can" Background="Lavender">  
 <Button Name="per" Click="buttonPrintPoint\_Click" Content="Добавление точек: off"></Button>  
 <Button Name="Selection" Click="Point\_Conection\_select" Content="Соеденить точки" Margin="126,0,0,0"></Button>  
 <Button Margin="225,0,0,0">Удалить вершину</Button>  
 <Button Margin="325,0,0,0">Передвинуть вершину</Button>  
 <TextBox Name="text1" Margin="500,1,1,1">1</TextBox>  
 <DataGrid Name="table" AutoGenerateColumns="False" HorizontalGridLinesBrush="DarkGray"  
 RowBackground="LightGray" AlternatingRowBackground="White" Padding="0" Margin="0,20,0,0">  
 <DataGrid.Columns>  
 <DataGridTextColumn Header="№" Binding="{Binding First}" Width="30" />  
 <DataGridHyperlinkColumn Header="X" Binding="{Binding Second}" Width="30" />  
 <DataGridTextColumn Header="Y" Binding="{Binding Third}" Width="30" />  
 </DataGrid.Columns>  
 </DataGrid>  
 </Canvas>  
   
 </Grid>  
   
</Window>

Код C#:

using System;  
using System.Collections.Generic;  
using System.Collections.ObjectModel;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Windows;  
using System.Windows.Controls;  
using System.Windows.Data;  
using System.Windows.Documents;  
using System.Windows.Input;  
using System.Windows.Media;  
using System.Windows.Media.Imaging;  
using System.Windows.Navigation;  
using System.Windows.Shapes;  
using System.Collections.Generic;  
  
namespace WpfApp1  
{  
   
 class MyClass  
 {  
 public string First { get; set; }  
  
 public string Second { get; set; }  
   
 public string Third { get; set; }  
   
 }  
 */// <summary>  
 /// Interaction logic for MainWindow.xaml  
 /// </summary>* public partial class MainWindow : Window  
 {  
 public MainWindow()  
 {  
 InitializeComponent();  
   
   
 }  
  
 public List<List<double>> list = new List<List<double>>();*// Двумерный список* public bool chek = false;*//переменная включен ли переключатель для добавления кнопок* public int Count\_Point = 0; *// счетчик количества точек-вершин* public bool Select = false; *// переменная которая хранит включенность кнопки* private void Grid\_MouseDown(object sender, MouseButtonEventArgs e) *// добавление точки* {  
   
 if (chek == false) *// проверка переулючателя добавления точки* {  
   
 }  
 else  
 {  
   
 Count\_Point++;*// прибавление +1 к счетчику вершин* TextBox TX = new TextBox(); *// создаём поле которое будет внутри нашей точки* TX.Text = Count\_Point.ToString();*// передаем полю значение нашего счетчика точек* Point p = e.GetPosition(this);*//получаем позицию нашей мышки в переменную "p"* Ellipse point = new Ellipse();*//создаём круг* TX.BorderBrush = null; *// убираем цвет бортика поля с переменной* TX.Background= null; *// убираем цвет поля с переменной* TX.IsEnabled = false; *// убираем возможность изменения надписи* TX.Foreground=Brushes.White; *// цвет шрифта для счетчка - белый* point.Fill = Brushes.MidnightBlue;*//цвет самой вершины-точки* point.Width = 25;*//размер точки в ширину* point.Height = 25;*// размер точки в высоту* point.Margin = new Thickness(p.X, p.Y, 0, 0);*//расположение точки - вершины на нащем канвасе* TX.Margin=new Thickness(p.X, p.Y, 0, 0);*//Распололжения счетчика на нашем канвасе* Can.Children.Add(point);*//добавления точки в канвас* Can.Children.Add(TX);*//добавление счетчика в канвас* ObservableCollection<MyClass> collection = null;  
 table.Items.Add(new MyClass()  
 {First = Count\_Point.ToString(), Second = p.X.ToString(), Third = p.Y.ToString()});  
 table.ScrollIntoView(true);  
 list.Add(new List<double>());  
 list[Count\_Point-1].Add(Count\_Point);  
 list[Count\_Point-1].Add(p.X);  
 list[Count\_Point-1].Add(p.Y);  
 }  
  
 if (Select == false)  
 {  
   
 }  
 else  
 {  
 Point p = e.GetPosition(this);  
 bool ok = false;  
 foreach (var item in list)  
 {  
   
 if (p.X <(item[1]+12.5)&&p.X>(item[1]-12.5) && p.Y < (item[2]+12.5)&&p.Y>(item[2]-12.5))  
 {  
 ok = true;  
 MessageBox.Show("Вы выбрали вершинцу" + item[0]);  
   
 }  
 }  
 if (ok == true)  
 {  
 text1.Text = "+";  
 }  
 else  
 {  
 text1.Text = "-";  
   
 }  
 }  
 }  
  
 private void Point\_Conection\_select(object sender, RoutedEventArgs e)*//Функция для выбора точки* {  
   
   
 if (Select == false)  
 {  
 Select = true;  
 Selection.Content = "+";  
   
 }  
 else  
 {  
 Select = false;  
 Selection.Content = "-";  
 }  
   
  
   
 }  
   
 private void Point\_Conect(object sender, RoutedEventArgs e) *// Функция соеденяющая две точки* {  
 Line line = new Line();  
 line.X1 = 10;  
 line.X2 = 50;  
 line.Y1 = 10;  
 line.Y2 = 50;  
 line.Fill = Brushes.Black;  
 Can.Children.Add(line);  
 }  
 private void buttonPrintPoint\_Click(object sender, RoutedEventArgs e) *// изменение значения переключателя и надписи в самой кнопке* {  
  
   
 if (chek == false)  
 {  
 per.Content = "Добавление точек: on";  
 chek = true;  
 }  
 else  
 {  
 per.Content = "Добавление точек: off";  
 chek = false;  
 }  
   
   
  
 }  
 }  
   
}