# Лабораторная работа №5

## Графические приложения на Windows Presentation Foundation (WPF-приложения)

**Краткое описание:** в данной лабораторной работе мы познакомимся с основами создания графических интерфейсов с использованием технологии WPF в C#. Мы рассмотрим создание оконных приложений, работу с элементами управления, обработку событий и рисование на холсте.

**Состав работы:** Данная лабораторная работа состоит из 4х заданий. Вот краткие условия заданий:

1) Создать WPF-приложение, которое при запуске сразу отображает окно с сообщением об ошибке.
2) Создать приложение с двумя текстовыми полями для ввода целых чисел (координат X и Y) и кнопкой. При нажатии на кнопку на указанных координатах рисуется красная точка.
3) Создать приложение, где точка изначально расположена в центре окна. При нажатии на точку выводится сообщение.
4) Создать приложение, где точка появляется в случайном месте при нажатии на нее, и внизу отображается прогресс-бар. Нужно нажать на точку 10 раз, чтобы прогресс-бар заполнился.

**Задание 1.**

**Создать простое WPF-приложение, которое при запуске сразу отображает окно с сообщением об ошибке.**

Для разработки приложений на WPF необходимо создать проект в VisualStudio с соответствующим шаблоном. Для этого после запуска нажмите «Создание проекта» и в открывшемся окне, строке поиска введите WPF и пролистайте до шаблона C# «Приложение WPF» как на рис. 1. Нажимайте на фиолетовую кнопку, пока не исчезнет диалоговое окно.

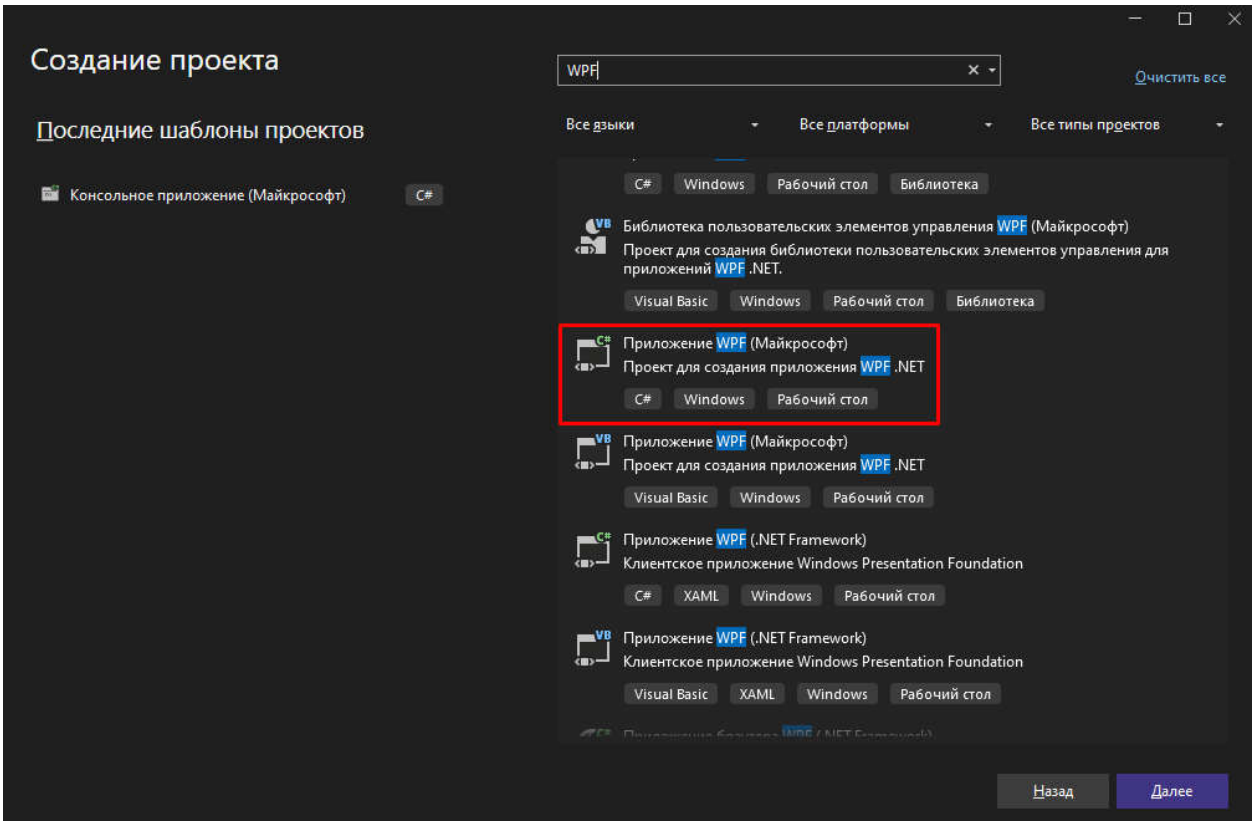


Рис. 1 – поиск необходимого шаблона.

Сразу после создания приложения по шаблону перед вами окажутся 2 вкладки (графический редактор и код). При запуске приложения должно появиться окно, которое можно перемещать, изменять его размер, растягивать на весь экран, сворачивать и закрывать.

Перейдите на вкладку с кодом, обратите внимание на отсутствие «Main». В консольных приложениях точкой входа (функция, которая запускается самой операционной системой) является метод «Main». В WPF-приложениях точка входа также существует, но она скрыта от пользователя и автоматически генерируется при компиляции. Тем не менее, функция «MainWindow» будет выполнена сразу при запуске приложения, а значит, именно здесь должен находиться вывод сообщения.

В «System.Windows» (`using System.Windows;`) существует класс «MessageBox», предоставляющий функцию создания окна с уведомлением. По аналогии с «Array.Reverse»: класс «Array» предоставляющий функцию сортировки.

Для выполнения первого задания необходимо вызвать функцию `MessageBox.Show("Успех!","Ошибка!");` в функции «MainWindow».

**Задание 2.**

**Создать приложение с двумя текстовыми полями для ввода целых чисел (координат X и Y) и кнопкой. При нажатии на кнопку на указанных координатах рисуется красная точка.**

Для выполнения задания 2 необходимо создать два текстовых поля и кнопку. Для этого откройте вкладку «MainWindow.xaml» и найдите справа «Панель элементов». Если она отсутствует, см. рис. 2.

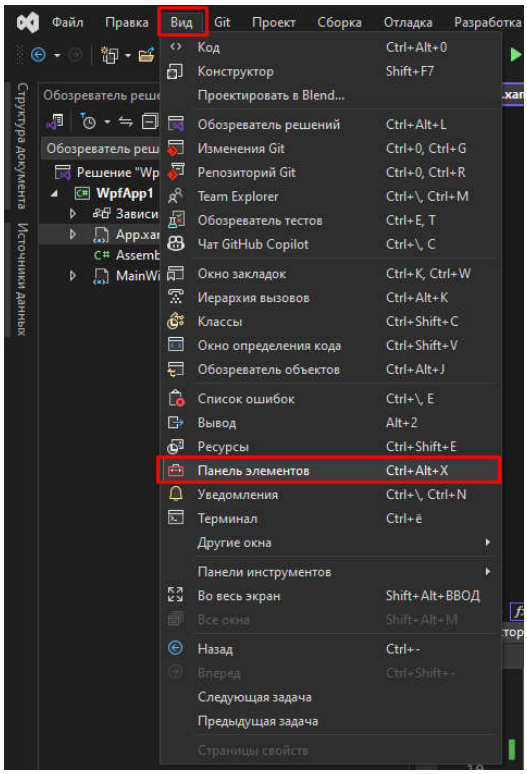

Рис. 2 – поиск «Панели элементов»

В панели элементов раскройте выпадающий список «Типовые элементы управления WPF». Перетащите Button куда-нибудь на окно графического интерфейса. Над появившейся кнопкой появится лампочка, нажмите на неё. Должно открыться окно как на рис. 3.

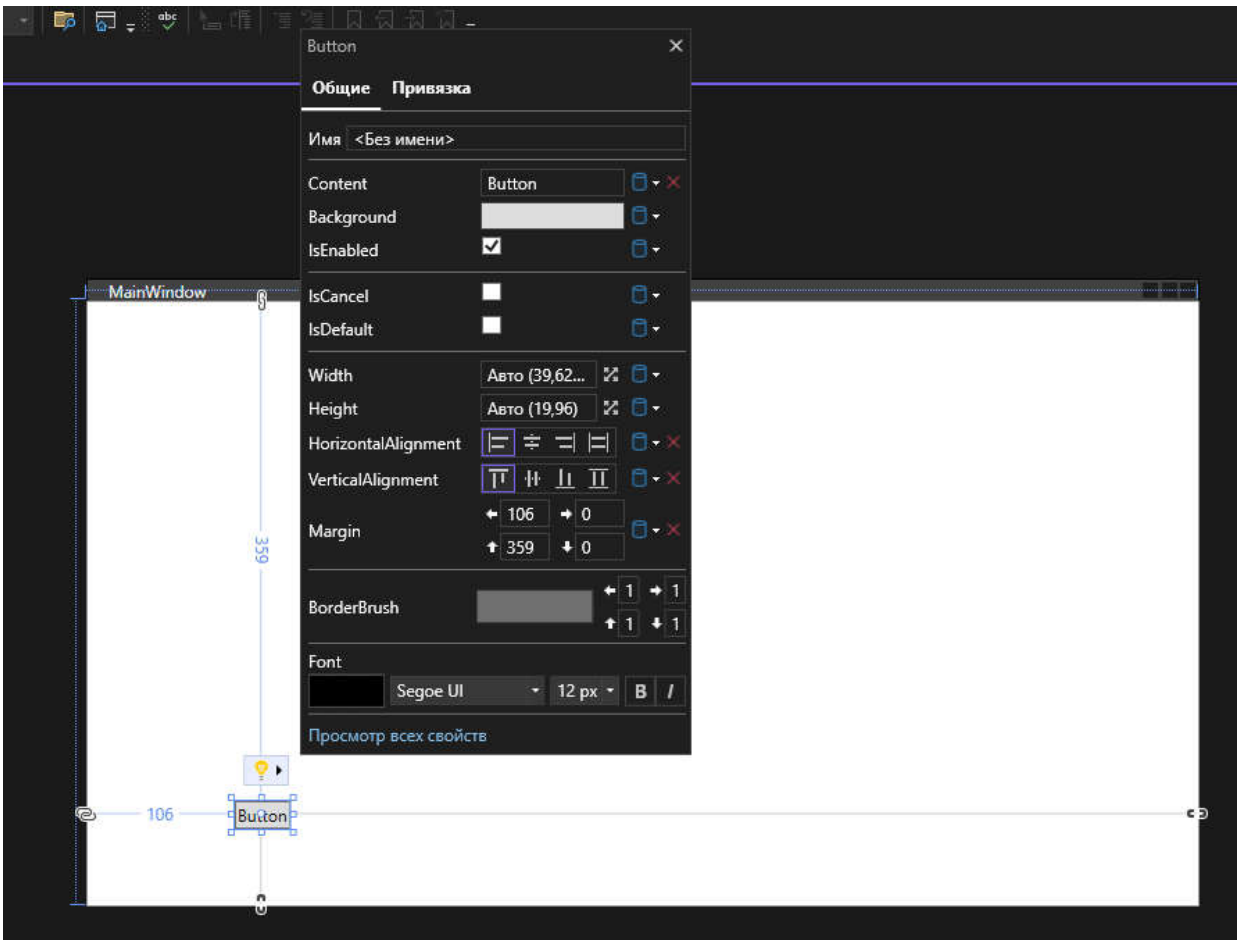


Рис. 3 – Редактирование свойств элемента

Для того, чтобы изменить надпись на кнопке, отредактируйте поле «Content». Обязательно нужно задать имя элемента, если собираетесь его использовать. Допускается использовать кириллицу в названии элементов, но крайне не рекомендуется этого делать.

Теперь, вы можете дважды кликнуть по созданной кнопке (в редакторе). Снова откроется окно с кодом, но автоматически добавится функция. Эта функция будет вызываться каждый раз на по нажатию кнопки.

Для того, чтобы добавить текстовое поле переместите в окно TextBox и присвойте имя элементу. Для того, чтобы получить содержимое этого текстового поля (пусть, с именем TB), используйте переменную «TB.Text». Вам не нужно инициализировать эту переменную. После того, как пользователь введет текст в это поле, значение переменной в программе изменится автоматически.

Также нужно иметь возможность добавить красную точку. Для этого в панели элементов, в разделе «Все элементы управления WPF» найдите элемент Canvas и перетащите его в окно (вам может показаться, что ничего не произошло). Добавьте имя элементу (например Canv).

Следующий код создает красную точку и рисует её в Canv таким образом, что левый верхний угол оказывается в координатах x, y:

```
Ellipse redDot = new Ellipse
{
    Fill = Brushes.Red,
    Width = 10,
    Height = 10
};

Canvas.SetLeft(redDot, x);
Canvas.SetTop(redDot, y);
Canv.Children.Add(redDot);
```

**Задание 3.**

**Создать приложение, где точка изначально расположена в центре окна. При нажатии на точку выводится сообщение.**

Теперь вам нужно сделать так, чтобы точка работала как кнопка. Для этого необходимо после создания точки добавить строку: `redDot.MouseLeftButtonDown += YourFunc;` и заменить «YourFunc» на ваше название функции для обработки нажатия.

**Задание 4.**

**Создать приложение, где точка появляется в случайном месте при нажатии на нее, и внизу отображается прогресс-бар. Нужно нажать на точку 10 раз, чтобы прогресс-бар заполнился.**

Для того, чтобы изменить положение уже существующего элемента, вам необходимо использовать конструкцию `Canvas.SetLeft(redDot, newX);` `Canvas.SetTop(redDot, newY);`. Здесь, вместо redDot должно быть название элемента, положение которого вы хотите изменить, а вместо newX, newY координаты левого верхнего угла элемента в новом положении.

Для того, чтобы добавить прогресс-бар (полоска загрузки) перетащите ProgressBar из раздела «Все элементы управления WPF» и присвойте ему имя (пусть PrBar). При изменении параметра PrBar.Value будет автоматически меняться полоска загрузки. Параметр Value имеет смысл процента заполнения и должен принадлежать диапазону 0-100.

**Теория.**

В данном разделе собрана теория разных уровней. Вы не обязаны понимать все термины т.к.:

1) В дальнейшем у вас будет предмет, посвященный объектно-ориентированному программированию, где дается большинство терминов.
2) Термины должны быть у вас «на слуху», чтобы потом было проще проассоциировать новый термин с уже знакомыми явлениями и скорректировать имеющееся представление.

**Разбор шаблона:**

Строка «`public partial class MainWindow : Window`» объявляет класс «MainWindow» при помощи трех ключевых слов. С 1 и 3 вы знакомы, а ключевое слово partial позволяет разделять определение класса на несколько файлов. В контексте WPF это используется для разделения кода логики (MainWindow.xaml.cs) и разметки интерфейса (MainWindow.xaml). Благодаря «partial» существует и метод «InitializeComponent», который также определен в классе «MainWindow», но в другом, скрытом файле, который генерируется и изменяется автоматически.

Двоеточие в объявлении класса в языке C# используется для обозначения наследования классов. В данном случае создаваемый вами класс «MainWindow» наследуется от уже существующего в C# класса «Window». Это означает, что «MainWindow» является подклассом «Window», то есть наследует все его свойства, методы и события, что позволяет «MainWindow» функционировать как окно в WPF (т.е., например, иметь кнопки свернуть, растянуть, закрыть).

Метод «MainWindow» (не путать с классом «MainWindow») должен существовать в классе «MainWindow» в обязательном порядке. Он отвечает за инициализацию класса (в данном случае инициализирует свои компоненты, которые определены в файле MainWindow.xaml при помощи «InitializeComponent»).

Метод «InitializeComponent» считывает файл MainWindow.xaml и добавляет в класс соответствующие объекты. Например, если создать при помощи графического редактора кнопку с названием MyBtn, то в коде можно будет использовать переменную MyBtn.

**Общий принцип работы WPF-приложения:**

В отличие от консольных приложений, которые выполняют код последовательно, WPF-приложения после инициализации переходят в режим ожидания событий. Событиями могут быть нажатия кнопок, перемещение мыши, ввод с клавиатуры и другие действия пользователя. По событию выполняется соответствующая реализованная вами функция.

**Общий алгоритм разработки простых WPF-приложений:**

1) Сформировать графический интерфейс приложения, перетаскивая в редакторе с панели инструментов необходимые элементы.
2) Присвоить созданным элементам имена.
3) Определить, какие элементы должны реагировать на события.
4) Реализовать в коде обработчики событий для таких элементов.