Практическая работа №18

Тема: Элементы управления. Работа с элементом InkCanvas

Цель работы: формирование навыков создания приложений по технологии WPF с использованием XAML в MS VS 2022

Задачи:

– изучение приемов создания приложений по технологии WPF с элементами управления и элементом InkCanvas

Материально-техническое обеспечение:

Место проведения: Компьютерный класс.

Время на выполнение работы: 2 часа.

Оборудование: ПК

Средства обучения: операционная система, текстовый процессор MS Word, программные средства определенного вида

Исходные данные:

1. Конспект занятия.
2. Задание для практической работы.

Перечень справочной литературы:

1) Программирование на языке высокого уровня. Программирование на языке С++: учеб. пособие / Т. И. Немцова, С. Ю. Голова, А. И. Терентьев ; под ред. Л. Г. Гагариной. – М. : ИД «ФОРУМ» : ИНФРА-М, 2018. – 512 с. – (Среднее профессиональное образование).

Краткие теоретические сведения:

Основные элементы управления WPF

Button (кнопка)

- Событие **Click** – нажатие на кнопку. В атрибуте Click указывается название функции-обработчика этого события.
- Свойство **IsCancel**. Возможные значения: True, False. Если записано True, то кнопка будет срабатывать при нажатии на кнопку Esc в данном окне, т.е. когда пользователь хочет закрыть окно без выполнения каких-либо действий.
- Свойство **IsDefault**. Возможные значения: True, False. Если записано True, то кнопка будет срабатывать при нажатии на кнопку Enter в данном окне, но только если не выделена какая-либо другая кнопка. В отличие от приложения Windows Forms, в WPF-приложении при открытии окна не происходит автоматического выделения какого-либо элемента. Чтобы выделить первый элемент в окне, необходимо нажать кнопку Tab. Кнопка со свойством IsDefault=”True” подсвечивается в окне, как будто она получила фокус. Но на самом деле кнопка не получает фокус, т.к. нажатие на клавишу «Пробел» не приводит к нажатию кнопки, а нажатие клавиши Tab приводит к выделению первого элемента на странице, а не элемента, следующего за кнопкой.

ToggleButton (переключаемая кнопка)

- Событие **Click** – нажатие или отжатие кнопки. В атрибуте Click указывается название функции-обработчика этого события.
- Событие **Checked** – нажатие кнопки. В атрибуте Checked указывается название функции-обработчика этого события.
- Событие **Unchecked** – отжатие кнопки. В атрибуте Unchecked указывается название функции-обработчика этого события.
- Свойство **IsChecked** – состояние кнопки. True – кнопка нажата, False – кнопка отжата.

CheckBox (независимый переключатель)

Класс CheckBox является наследником от класса ToggleButton и наследует его свойства и события.

Для обращения к элементу управления из кода программы необходимо в XAML-коде задать для него имя в атрибуте Name с префиксом x, как это показано в примере выше. Префикс 'x:' означает пространство имен XAML, а не пространство имен WPF.

RadioButton (зависимый переключатель)

Класс RadioButton является наследником от класса ToggleButton и наследует его свойства и события.

- Свойство **GroupName** – название группы зависимых переключателей. В одном окне может быть несколько групп зависимых переключателей с разными названиями групп.

ComboBox (выпадающий список)

элементы которого определены с помощью элементов **ComboBoxItem** В качестве содержимого элементов выпадающего списка можно задавать не только текст, но и другие элементы, например эллипс или прямоугольник.

ListBox (список)

элементы которого определены с помощью элементов **ListBoxItem**: В качестве содержимого элементов списка можно задавать не только текст, но и другие элементы.

После заполнения элемента управления ComboBox (или ListBox) есть три способа определен выбранного в них элемента. Во-первых, если необходимо найти числовой индекс выбранного элемента, необходимо использовать свойство **SelectedIndex** (отсчет начинается с 0; -1 означает отсутствие выбора). Во-вторых, если требуется получить объект, выбранный внутри списка, то используется свойство **SelectedItem**.В-третьих, **SelectedValue** позволяет получить значение выбранного объекта.

Slider

Элемент Slider представляет собою ползунок с минимальным значением **Minimum**, максимальным значением **Maximum** и текущим значением **Value**.

Меню

Меню в WPF представлено классом **Menu**, который может включать в себя набор объектов **MenuItem**. Каждый объект **MenuItem** в свою очередь может включать в себя другие объекты **MenuItem** и объекты **Separator** (разделитель), а также другие элементы управления, например зависимые (RadioButton) и независимые (CheckBox) переключатели:. Знак подчеркивания в названиях пунктов меню указывает «горячие» клавиши для доступа к этим пунктам меню

Панель инструментов

Панель инструментов в WPF представлена классом **ToolBar**, который в качестве содержимого может включать в себя коллекцию любых других элементов. Панели инструментов обычно используются как альтернативный способ активизации пунктов меню.

Строка состояния

Строка состояния в WPF представлена классом **StatusBar**, который в качестве содержимого может включать в себя коллекцию любых других элементов, в том числе **StatusBarItem**.

InkCanvas

Элемент управления InkCanvas позволяет рисовать и редактировать линии с помощью мыши или пера. Размеры элемента управления можно задать с помощью свойств **Width** и **Height**. Свойства пера (цвет, ширину и высоту) можно настроить с помощью свойства **DefaultDrawingAttributes**: Компьютерная графика в отличие от бумажной позволяет довольно просто перемещать объекты. Особенно, если работа ведется с векторными объектами или с отдельными слоями растрового изображения.

У контрола **InkCanvas** есть свойство **EiditingMode** (дословно - режим редактирования), позволяющий менять режим его работы.

Возможные значения свойства:

- ***GestureOnly*** – след на экране показывается только тогда, когда зажата левая кнопка мышки. Как только кнопка отпущена, нарисованное пропадает. Такой режим хорош для указания на какую-то часть изображения при проведении презентации;
- ***InkAndGesture***—интересный режим, позволяет как рисовать, так и удалять не рисовать
- ***EraseByStroke*** - в этом режиме в InkCanvas происходит удаление штриха при клике.
- ***EraseByPoint*** - удаление рисунка по точкам;
- ***Select*** – в этом режиме можно выделять объекты для их дальнейшего перемещения или удаления. Для выделения нужно обвести объект или кликнуть на него;
- ***None*** - запрет ввода рисунка.

<u>Ход работы:</u>

Требования к содержанию отчета:

- Номер и название практической работы.

- Цель работы.
- По каждой заданию (задаче/примеру) экранные формы (при наличии) и листинг программного кода, показывающие порядок выполнения практической работы, и результаты, полученные в ходе её выполнения.
- Ответы на контрольные вопросы в тетради.

Порядок выполнения работы:

Все задания практической работы размещать в своей папке проектов в новой папке

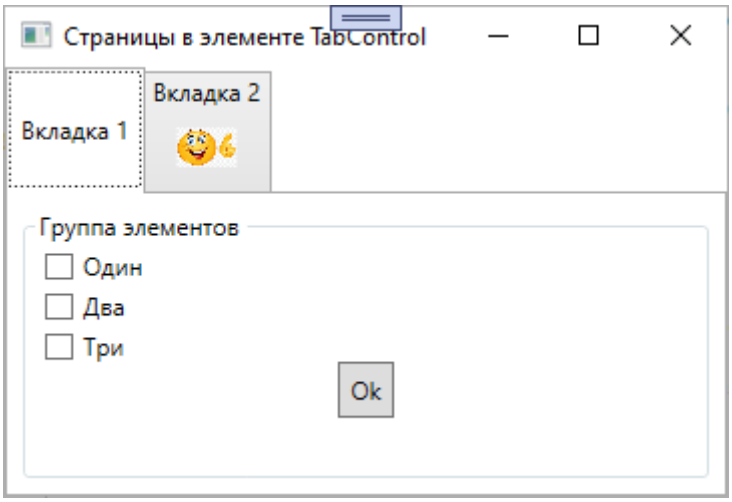
Пр18_Фамилия

Файлы проектов сохранять под именами pr18_N_Фамилия (N - номер задания)

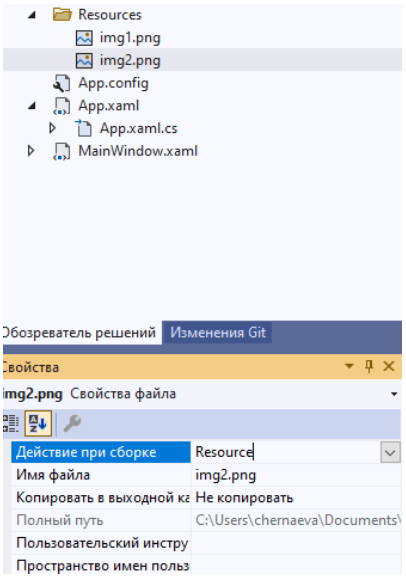
В начале каждого файла проекта установить комментарии: пр.р.№_____ (указать номер), свою Фамилию. Формулировку задания

Класс TabItem. Объекты TabItem представляют собой страницы в элементе TabControl. Единственным существенным членом, добавленным в класс Tabltem, является свойство *IsSelected*, которое указывает, видима ли данная вкладка в TabControl.

Задание 1. Создайте интерфейс приложения по технологии WPF (.NET Framework) (WPF Application) по образцу:



Примечание: *Как и свойство* ***Content****, свойство* ***Header*** *может содержать объект любого типа. Объекты, порожденные от класса* ***UIElement****, выводятся с помощью графической прорисовки, а текстовые и другие объекты — с помощью метода* ***ToStringO.*** *Это означает, что можно создать групповую панель или вкладку, заголовок которой содержит графическое содержимое или другие произвольные элементы.*



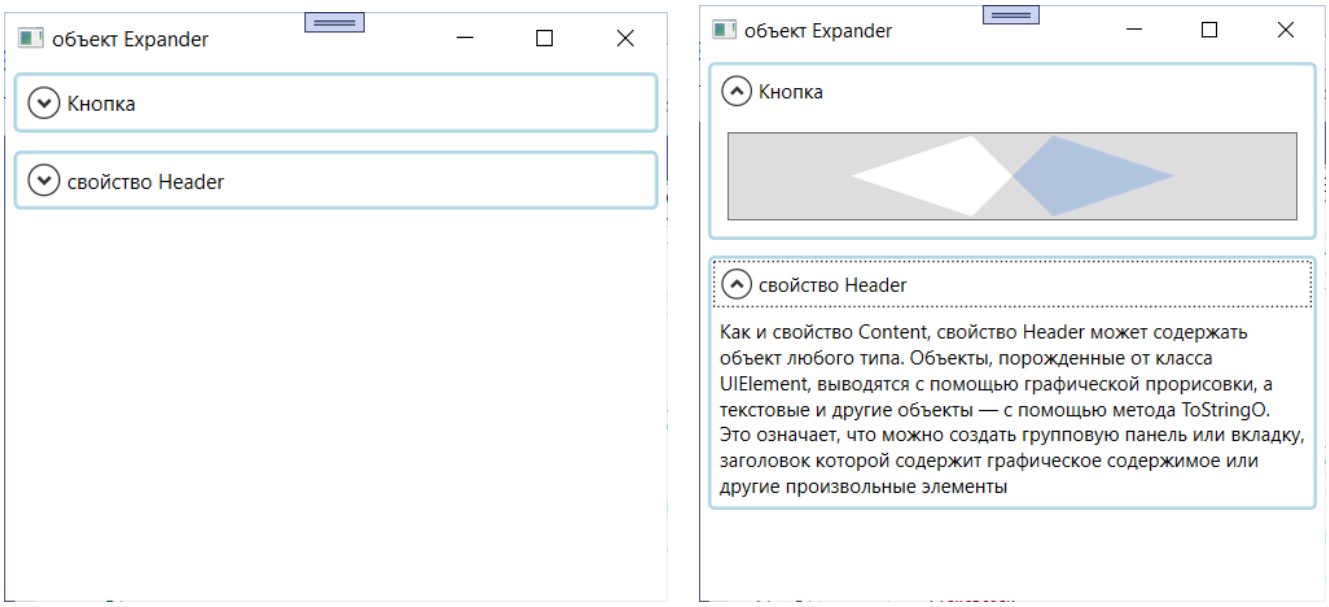
Не забудьте про изображения!!! Добавьте изображение к ресурсам проекта. Выделите имя файла с изображением в структуре проекта и проверьте следующие установки:

В поле **Действие при сборке** установить значение ***Resource*****;** в поле **Копировать в выходной каталог** установить значение ***Не копировать***

При необходимости выполните пересборку решения

Задание 2. Используя объект **Expander**, создайте интерфейс приложения по технологии WPF (.NET Framework) (WPF Application) следующего вида:

Для этого нужно лишь упаковать в него все сворачиваемое содержимое.

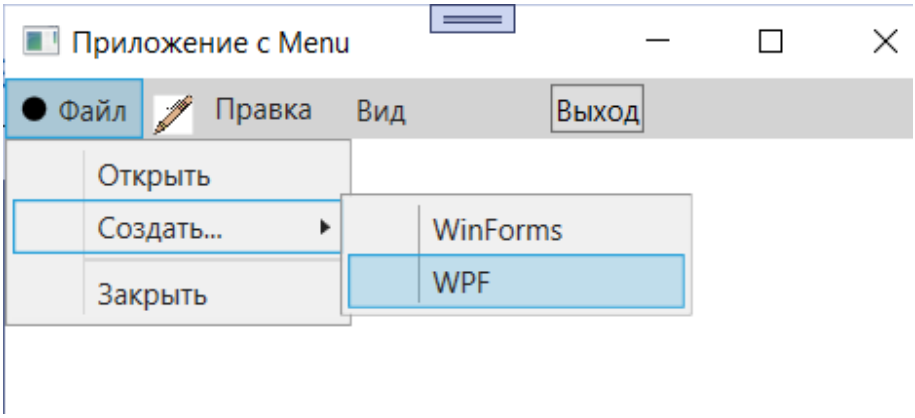


***Примечание**: Для формирования кнопки используйте графическую фигуру полигон:*

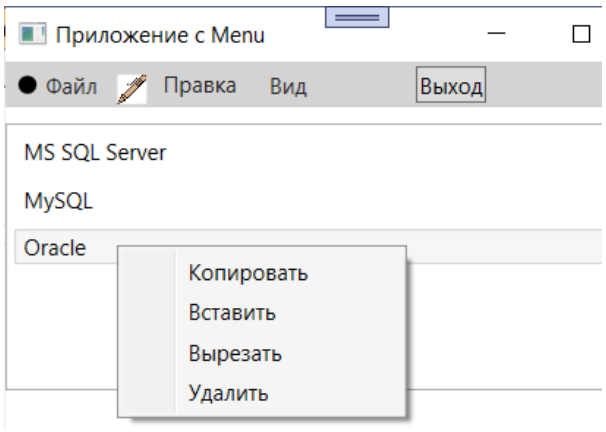
```
<Polygon Points="100,25 125,0 200,25 125,50" Fill="LightSteelBlue"></Polygon>
<Polygon Points="100,25 75,0 0,25 75,50" Fill="White"></Polygon>
```

Задание 3. Создайте интерфейс приложения по технологии WPF (.NET Framework) (WPF Application) по образцу:

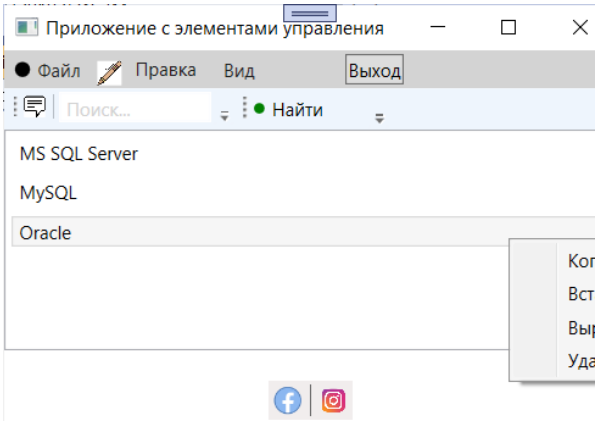
1) Используя различные элементы и свойства, создайте меню, содержащее иконку, изображение и кнопку:



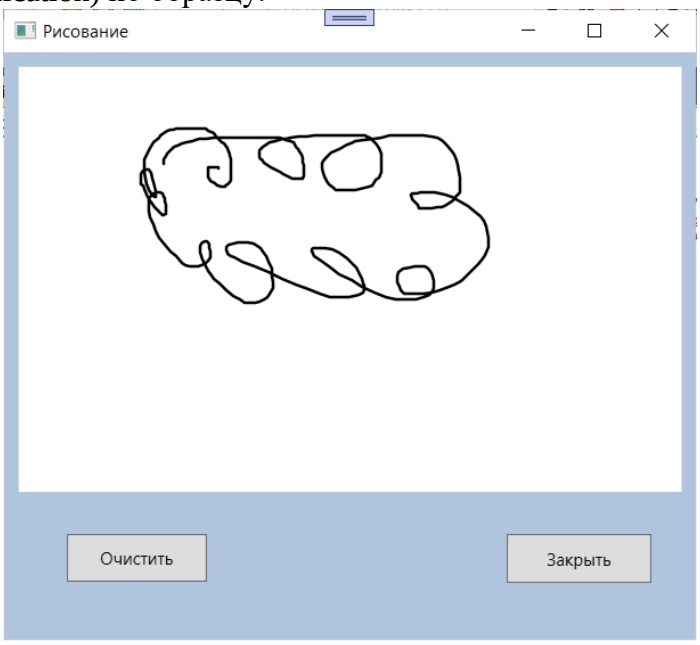
2) Добавьте список и контекстное меню



3) Используя элементы **ToolBarTray** и **StatusBar** добавить панель инструментов и значки социальных сетей



Задание 4. Используя элемент **InkCanvas,** создайте приложение по технологии WPF (.NET Framework) (WPF Application) по образцу:



1) Создать полотно и кнопки

```
    Title="Рисование" Height="450" Width="500">
<Grid Background="LightSteelBlue">
    <InkCanvas x:Name="inkCanvas1"  Margin="10,10,10,10" Height="299" Width="497"
               HorizontalAlignment="Left"   VerticalAlignment="Top"/>
    <Button Content="Очистить" Click="Button_Click"  Margin="44,338,0,0"
            HorizontalAlignment="Left"  VerticalAlignment="Top" Width="101" Height="35"/>
    <Button x:Name="btn_Close" Click="btn_Close_Click" Content="Закрыть" Margin="353,338,0,0"
            HorizontalAlignment="Left"  VerticalAlignment="Top" Width="101" Height="35"  />
</Grid>
```

2) Код обработчика для кнопки **Очистить:**

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    this.inkCanvas1.Strokes.Clear();
}
```

3) Самостоятельно код обработчика для кнопки **Закрыть**
4) Добавьте в окно кнопку Сохранить. Опишите код обработчика для кнопки **Сохранить:**

```
private void btn_Save_Click(object sender, RoutedEventArgs e)
{
    string imgPath = @"file.gif"; //Имя файл куда сохраняется рисунок
    //При необходимости создать поток памяти :)
    //MemoryStream ms = new MemoryStream();
    FileStream fs = new FileStream(imgPath, FileMode.Create); //  Поток файла :)

    //rtb – объект класса RenderTargetBitmap
    RenderTargetBitmap rtb = new RenderTargetBitmap((int)inkCanvas1.Width, (int)inkCanvas1.Height, 96, 96, PixelFormats.Default);
    rtb.Render(inkCanvas1);

    GifBitmapEncoder gifEnc = new GifBitmapEncoder(); //сохраняем в формате GIF
    gifEnc.Frames.Add(BitmapFrame.Create(rtb));
    gifEnc.Save(fs);
    fs.Close();
    MessageBox.Show("Файл сохранен, " + imgPath); //Для информации
}
```

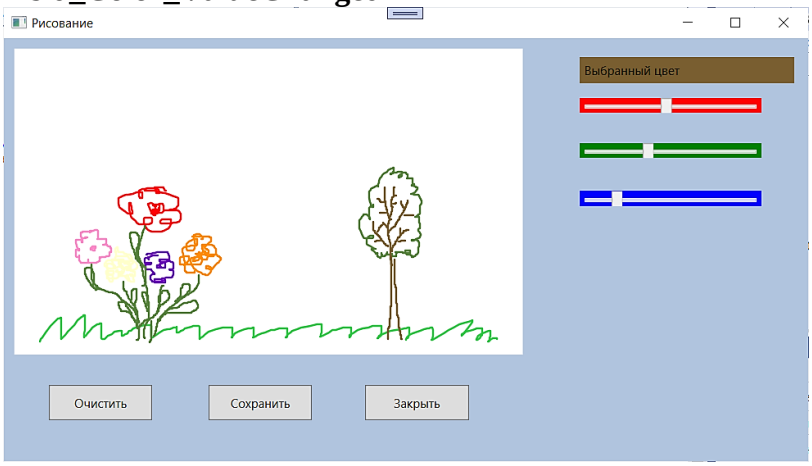
5) Создайте в окне элементы для выбора цвета.
   Для динамического подбора составляющих цвета воспользуйтесь элементом управления типа ***Slider****.*
   - Перетащите в окно приложения **3 слайдера** и **1** контрол типа **Label** для отображения цвета. Настройте одинаковые размеры слайдеров.

- Для **Label** задайте имя **lbl1**.

Ползунки необходимо раскрасить в соответствующие цвета, а также для всех их создать один обработчик на событие ValueChanged.
Задайте имена для ползунков так, чтобы было понятно, каким цветом они управляют - sld_RedColor, sld_GreenColor, sld_BlueColor, А также минимальное значение – 0, максимальное – 255, обработчик назовите sld_Color_ValueChanged.

- Для этого в файле XAML-кода укажите:

  Для Slider задайте имена соответственно **sld_RedColor, sld_GreenColor, sld_BlueColor**
  Minimum="0" Maximum="255"
  ValueChanged="sld_Color_ValueChanged"



- цвет в системе **RGB** передается через 3 цвета. Для этого создайте класс **ColorRGB** для удобства доступа к свойствам цвета:

```
public class ColorRGB
{
    Ссылок: 5
    public byte red { get; set; }
    Ссылок: 5
    public byte green { get; set; }
    Ссылок: 5
    public byte blue { get; set; }
}
```

- Далее создайте публичную переменную **mcolor** класса **ColorRGB** и переменную класса **Color** (Sytem.Windows.Media.Color) **clr**.

```
public ColorRGB mcolor { get; set; }
Ссылок: 2
public Color clr { get; set; }
```

- В методе **MainWindow()** установите значения цвета по умолчанию и установите фон контрола **Label**.

```
public MainWindow()
{
    InitializeComponent();

    mcolor = new ColorRGB();
    mcolor.red = 0;
    mcolor.green = 0;
    mcolor.blue = 0;
    this.lbl1.Background = new SolidColorBrush(Color.FromRgb(mcolor.red, mcolor.green, mcolor.blue));
}
```

- Код обработчика события **sld_Color_ValueChanged**

```
private void sld_Color_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
{
    var slider = sender as Slider;
    string crlName = slider.Name; //Определяем имя контрола, который покрутили
    double value = slider.Value; // Получаем значение контрола

    //В зависимости от выбранного контрола, меняем ту или иную компоненту и переводим ее в тип byte
    if (crlName.Equals("sld_RedColor"))
    {
        mcolor.red = Convert.ToByte(value);
    }
    if (crlName.Equals("sld_GreenColor"))
    {
        mcolor.green = Convert.ToByte(value);
    }
    if (crlName.Equals("sld_BlueColor"))
    {
        mcolor.blue = Convert.ToByte(value);
    }
    //Задаем значение переменной класса clr
    clr = Color.FromRgb(mcolor.red, mcolor.green, mcolor.blue);
    //Устанавливаем фон для контрола Label
    this.lbl1.Background = new SolidColorBrush(Color.FromRgb(mcolor.red, mcolor.green, mcolor.blue));
    // Задаем цвет кисти для контрола InkCanvas
    this.inkCanvas1.DefaultDrawingAttributes.Color = clr;
}
```

- Проверьте результат работы слайдеров.

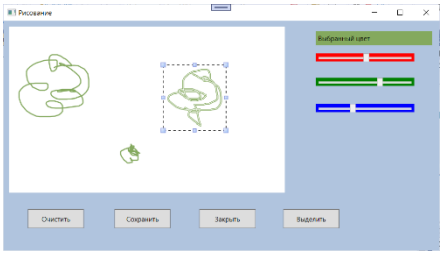
6) Попробуйте самостоятельномзадать цвет по умолчанию, отличный от черного
7) Выделение нарисованных объектов

   Для возможности выделения нарисованных объектов необходимо установить для свойства ***InkCanvas EditingMode*** *значение* ***Select****.*

- Разместите в окне приложения кнопку "**Выделить**" со свойствами **x:Name="btn_Select"** и назначьте обработчик для нажатия **btn_Select_Click**.
- В обработчике установите свойству **EditingMode** значение **Select**.

```
private void btn_Select_Click(object sender, RoutedEventArgs e)
{
    this.inkCanvas1.EditingMode = InkCanvasEditingMode.Select;
}
```

- Проверьте результат работы



При нажатии кнопки "**Выделить**" **InkCanvas** переключается в режим **Select**, из-за чего рисовать больше нельзя, но можно двигать или удалять выделенные объекты

- Попробуйте сохранить изображение, когда объект выделен. В данном случае – выделение также будет сохранено.
- Для того чтобы это изменить, добавьте в обработчик кнопки сохранения файла перевод **EditingMode** сначала в режим **None**, потом в **Ink**:

в начале описания кода события кнопки Сохранить режим запрета ввода рисунка:

```
this.inkCanvas1.EditingMode = InkCanvasEditingMode.None;
```

и перед выводом сообщения о сохранении файла режим рисования:

```
this.inkCanvas1.EditingMode = InkCanvasEditingMode.Ink;
```

- Проверьте результат работы

8) Доработайте обработчик событий **btn_Select_Click** так, чтобы можно было менять режим работы **InkCanvas** с режима рисования **Ink** на режим выделения **Select** и обратно (реализуйте

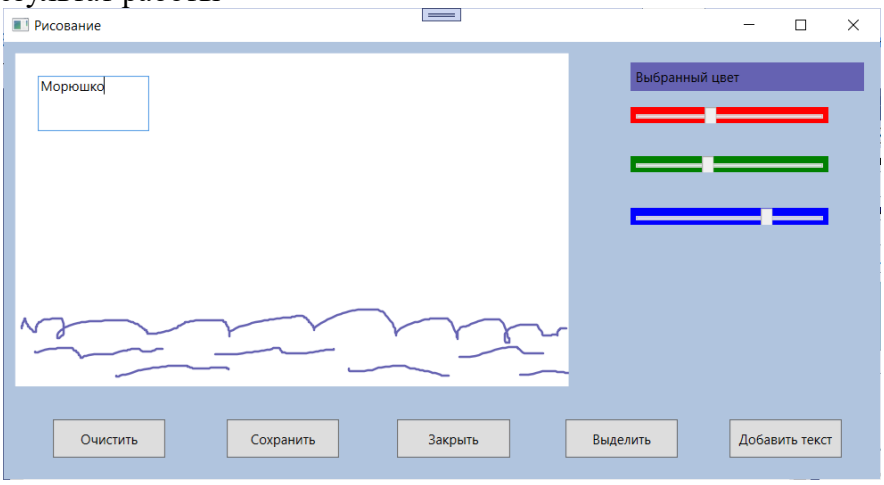
на свое усмотрение, например, добавив элемент выбора режима из списка или при выборе переключателя и т.п.).

9) Добавление текста

- Добавьте в рабочую область кнопку с заголовком "**Добавить текст**" и пропишите ей обработчик для нажатия "**btn_AddText_Click**":

```
private void btn_AddText_Click(object sender, RoutedEventArgs e)
{
    //Инициализация контрола tb типа TextBox
    TextBox tb = new TextBox
    {
        Width = 100,
        Height = 50,
        BorderThickness = new Thickness(1),
        BorderBrush = new SolidColorBrush(Color.FromRgb(5, 5, 5)),
        Margin = new Thickness(20, 20, 0, 0)
    };
    //Добавление контрола tb
    this.inkCanvas1.Children.Add(tb);
    //Переключение фокуса на элемент, чтоб можно было сразу ввести текст с клавиатуры
    tb.Focus();
}
```

- Проверьте результат работы



*Чтобы переместить введенный текст в другое место на экране, нужно переключиться в режим выделения (нажать кнопку "**Выделить**"), кликнуть на область с текстом, чтоб вокруг текста появилась рамка, и, наведя курсор на границы появившейся рамки и зажав левую кнопку мыши, переместить в нужное место.*

Таким же образом можно изменять размеры области для ввода текста, только нужно не перемещать рамку, а изменить ее размеры.

*При повторных нажатиях на кнопку "**Добавить текст**" будут добавляться новые элементы для ввода текста*

Контрольные вопросы:

1) Куда сохраняются данные, если не задан путь?
2) Приведите пример кода как добавить изображение во вкладку
3) Приведите пример кода формирования кнопки, используя графическую фигуру.
4) Как создать строку состояния. Приведите пример кода
5) Опишите методы работы с элементом **InkCanvas**: создание, очистка полотна, задание цвета, сохранение рисунка, выделение нарисованного объекта
6) Опишите возможные значения свойства **EiditingMode** элемента **InkCanvas**. Приведите пример как программно установить режим

Дополнительные задания:

1) Разместите в окне три кнопки и одно текстовое поле. Сделайте н кнопках следующие надписи: «*********», «**+++++**», «**00000**». Создайте обработчики события нажатия на данные кнопки, которые будут выводить текст, написанный на кнопках, в поле ввода.
2) Разместите в окне текстовое поле ввода, метку и кнопку. Создайте обработчик события нажатия на кнопку, который будет копировать текст из поля ввода в метку. Создайте обработчик события нажатия кнопки мыши в окне (Click), который будет устанавливать цвет окна и менять текст метки на строку **«Начало работы»** и очищать поле ввода.