Windows 10

Лабораторный практикум

Использование рукописного ввода

Октябрь 2015 года

Microsoft

Обзор

Рукописный ввод является мощным средством преобразования естественного движения в пиксели. Поскольку все больше устройств поддерживает сенсорные экраны, то рукописный ввод играет всё большую роль во взаимодействии с пользователями. Традиционно стилус является основным устройством ввода чернил. В UWP, использование стилуса, касаний и мыши может быть добавлено всего несколькими строчками кода. Элемент InkCanvas упростит работу с чернилами в вашем приложении.

В этом курсе вы создадите InkCanvas, принимающий ввод данных через мышку, прикосновения и стилус. Вы научитесь моментально настраивать цвет чернил и стирать, очищать, сохранять и загружать штрихи. В Упражнении 2 вы осуществите распознавание рукописных текстов на InkCanvas.

Цели

Пройдя эту работу, вы сможете:

- Создавать InkCanvas в приложении UWP
- Применять для ввода данных стилус, прикосновение и мышь
- Изменять цвет чернил по выбору пользователя
- Стирайте одиночные штрихи
- Очищать холст
- Сохранять и загружать штрихи
- Распознавать рукописный текст

Системные требования

Чтобы выполнить этот курс, необходимо обладать следующим набором программных инструментов:

- Microsoft Windows 10
- Microsoft Visual Studio 2015

Настройка

Вам следует выполнить следующие действия для подготовки компьютера:

1. Установить Microsoft Windows 10.
2. Установить Microsoft Visual Studio 2015.

Упражнения

Эта лабораторная работа включает следующие упражнения:

1. Рисование с помощью InkCanvas
2. Распознавание рукописных текстов

Расчетное время для завершения практикума: **45-60 минут**.

Упражнение 1: Рисование с помощью InkCanvas

InkCanvas является отличным способом быстро применения чернил в вашем UWP приложении. В этом упражнении вы создадите InkCanvas и модифицируете его графические атрибуты для создания штрихов разных цветов. Вы также сможете сохранить, загрузить и стереть штрихи и очистить холст.

Задача 1 – Создайте шаблон приложения Universal Windows

Начнём с создания проекта из шаблона пустого приложения.

1. В новой версии Visual Studio 2015 выберите **File (Файл) -> New (Новый) -> Project (Проект)**, чтобы открыть диалоговое окно New Project (Новый проект). Далее **Installed (Установленное) > Templates (Шаблоны) > Visual C# > Windows > Universal**, а затем выберите шаблон **Blank App** приложения (Universal Windows).
2. Назовите свой проект **Inking** и выберите местоположение файловой системы, в котором будет осуществлено хранение результатов прохождения лабораторного курса. Мы создали папку на диске **C** и назвали её **HOL**, именно папку с этим названием вы будете видеть на скриншотах в ходе всего практического курса.

   Не изменяйте настройки, установленные для **Create new solution (Создания нового решения)** и **Create directory for solution (Создания папки для решения)**. Вы можете

снять галочки как с **Add to source control (Добавить в систему контроля версий)**, так и с **Show telemetry in the Windows Dev Center (Отобразить телеметрию в Windows Dev Center)**, если не хотите использовать эти возможности. Нажмите **OK** для создания проекта.

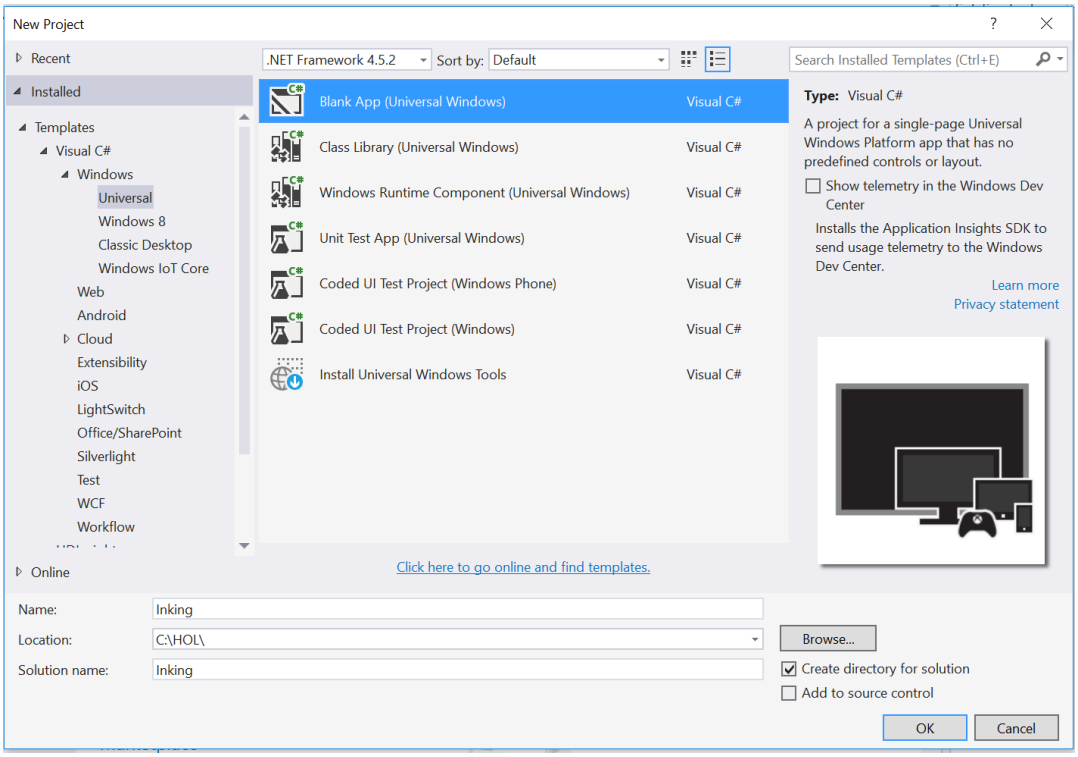

Рисунок 1

Создание нового проекта приложения в Visual Studio 2015.

3. Настройте Текущую конфигурацию решения на **отладку** и платформу решений на **x86**. Выберите **Локальный компьютер** из выпадающего меню **цели отладки**.

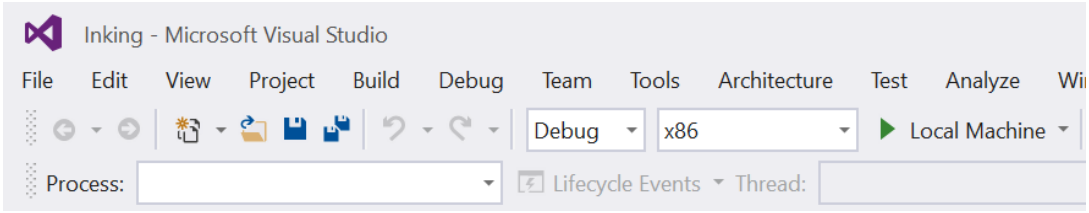

Рисунок 2

Сконфигурируйте свое приложение таким образом, чтобы оно запускалось на Локальном компьютере.

4. Скомпилируйте и запустите своё приложение. Вы увидите окно шаблона приложения со счетчиком частоты кадров, активированном по умолчанию для отладки.

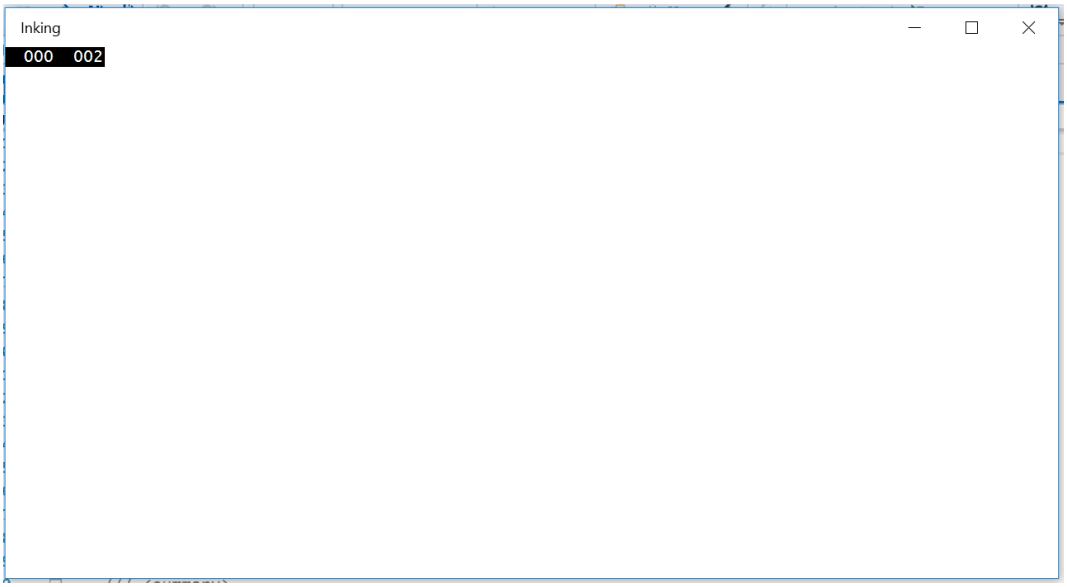

Рисунок 3

Пустое универсальное приложение, запущенное в режиме настольного компьютера.

Примечание: Счетчик частоты кадров является инструментом, используемым в процессе отладки, который помогает следить за производительностью вашего приложения. Он полезен для тех приложений, которые требуют интенсивной графической обработки, однако не подходит для простых приложений, которые будут создаваться вами в данном практическом курсе.

В шаблоне пустого приложения директива препроцессора для активации или отключения счетчика частоты кадров находится на **App.xaml.cs**. Счётчик скорости кадра может перекрывать или скрывать содержание вашего приложения, если вы оставляете его включённым. Для целей настоящего курса вы можете отключить его, установив значение **DebugSettings.EnableFrameRateCounter** как **false**.

5. Вернитесь в Visual Studio и остановите отладку.

Задача 2 – Создайте InkCanvas

Вы можете начать использование чернил в своём приложении используя всего несколько строчек кода. В этой задаче мы будем использовать большое полотно чернил и две строки вспомогательных элементов управления выше и ниже его.

1. Откройте **MainPage.xaml**. Мы настроили фон **Grid** на **LightGray (Светло-серый)** и добавили **RowDefinitions (Определения строк)** для трёх строк. Верхние и нижние строки будут содержать опции и кнопки стиля, в большой центральной строке будет находиться InkCanvas.

XAML

```
<Grid Background="LightGray">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
```

```
        <RowDefinition Height="2*" />
        <RowDefinition Height="Auto" />
    </Grid.RowDefinitions>
</Grid>
```

2. Добавьте разметку для этих трех строк. Первая строка будет содержать **Grid** шириной в два столбца, в каждом из которых будет горизонтальная StackPanel. Вторая строка будет содержать **InkCanvas**, третья строка будет состоять еще из одной StackPanel. Назовите свой InkCanvas атрибутом **x:Name**.

XAML

```
<Grid Background="LightGray">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="2*" />
        <RowDefinition Height="Auto" />
    </Grid.RowDefinitions>
    <Grid Grid.Row="0" Margin="12">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
        </Grid.ColumnDefinitions>
        <StackPanel Grid.Column="0" Orientation="Horizontal">
        </StackPanel>
        <StackPanel Grid.Column="1" Orientation="Horizontal"
HorizontalAlignment="Right">
        </StackPanel>
    </Grid>
    <Grid Grid.Row="1" Background="White" Grid.ColumnSpan="2">
        <InkCanvas x:Name="InkCanvas" />
    </Grid>
    <StackPanel Grid.Row="2" Orientation="Horizontal" Margin="12">
    </StackPanel>
</Grid>
```

3. На этом этапе ваше приложение готово к простому чернильному ввоу с помощью стилуса. Если ваш компьютер не поддерживает стилус, мы научимся использовать другие устройства ввода ниже.

4. Во вспомогательном коде MainPage добавьте пространство имен **Windows.UI.Input.Inking.**

Примечание: InkCanvas по умолчанию принимает только ввод стилусом. Вы можете использовать мышь или прикосновения в качестве устройств дополнительного ввода.

5. Добавить **мышь** и **прикосновения**, как типы устройств ввода для **InkCanvas**.

C#

```
public MainPage()
{
    this.InitializeComponent();
```

```
    InkCanvas.InkPresenter.InputDeviceTypes =
        Windows.UI.Core.CoreInputDeviceTypes.Mouse |
        Windows.UI.Core.CoreInputDeviceTypes.Pen |
        Windows.UI.Core.CoreInputDeviceTypes.Touch;
}
```

Примечание: InkCanvas содержит одно свойство обведения чернилами – InkPresenter. InkPresenter отображает чернила на полотне и позволяет устанавливать атрибуты для нанесения штрихов.

6. Скомпилируйте и запустите приложение. Сделайте рисунок на белом **InkCanvas** с помощью мышки, стилуса или прикосновений. Вы увидите цвет чернил по умолчанию и атрибуты размеров пера.

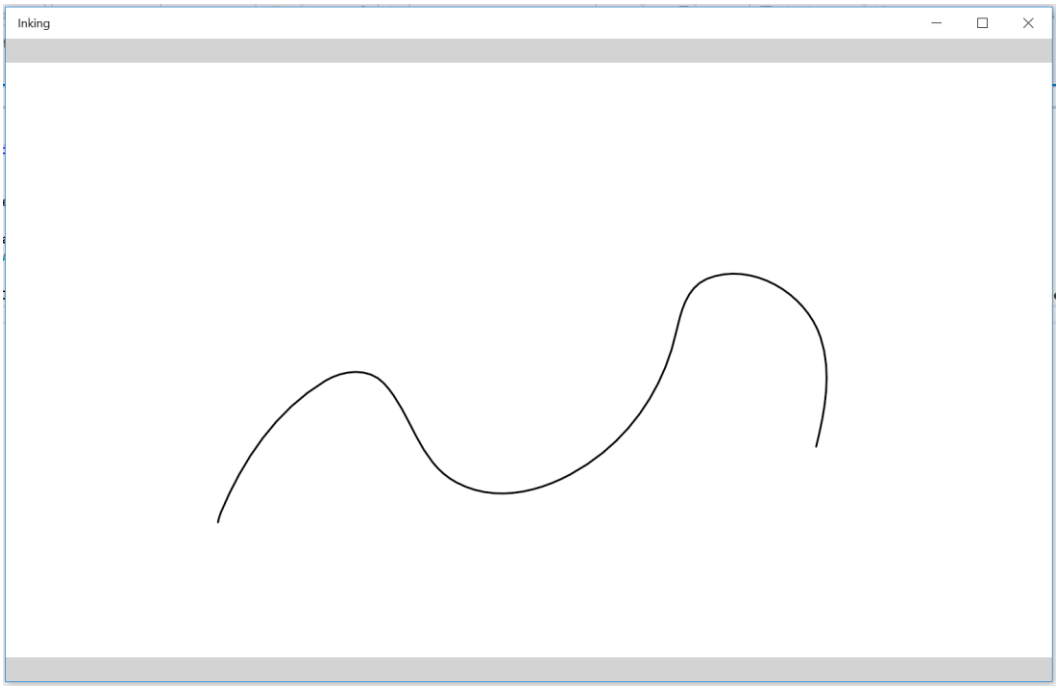

Рисунок 4

Графические атрибуты по умолчанию для InkCanvas.

7. Завершите отладку и вернитесь в Visual Studio.

Задача 3 – Настройте графические атрибуты

Вы можете настраивать графические атрибуты для **InkCanvas** с помощью экземпляра **InkDrawingAttributes**. Вы также можете по выбору пользователей программным путём обновлять атрибуты. В этой задаче вы сначала добавите атрибуты по умолчанию в стиль рукописного фрагмента, а затем кнопки, чтобы изменять атрибуты в процессе ввода.

1. Вы можете установить атрибуты сразу при создании экземпляра **InkDrawingAttributes**. Добавьте атрибуты **цвета**, **размера**, **надавливания** и **подогонки под кривую** в конструктор **MainPage**. После установки обновите графические атрибуты InkCanvas.

C#

```
public MainPage()
{
    this.InitializeComponent();

    InkDrawingAttributes drawingAttributes = new InkDrawingAttributes();

    drawingAttributes.Color = Windows.UI.Colors.Red;
    drawingAttributes.Size = new Size(4, 4);
    drawingAttributes.IgnorePressure = false;
    drawingAttributes.FitToCurve = true;


InkCanvas.InkPresenter.UpdateDefaultDrawingAttributes(drawingAttributes);

    InkCanvas.InkPresenter.InputDeviceTypes =
        Windows.UI.Core.CoreInputDeviceTypes.Mouse |
        Windows.UI.Core.CoreInputDeviceTypes.Pen |
        Windows.UI.Core.CoreInputDeviceTypes.Touch;
}
```

Примечание: Хотя IgnorePressure является опцией графических атрибутов чертежа, **InkCanvas** на данный момент не отображает чувствительные к надавливанию штрихи. Вместо этого значение надавливания фиксируется в каждой точке.

Другие графические атрибуты включают PenTip и DrawAsHighlighter. Больше информации **по классам графических атрибутов** для рисования вы можете найти по ссылке https://msdn.microsoft.com/en-us/library/windows.ui.input.inking.inkdrawingattributes.aspx

2. Скомпилируйте и запустите приложение. В этот раз ваш рукописный стиль отобразит более широкий штрих красного цвета.

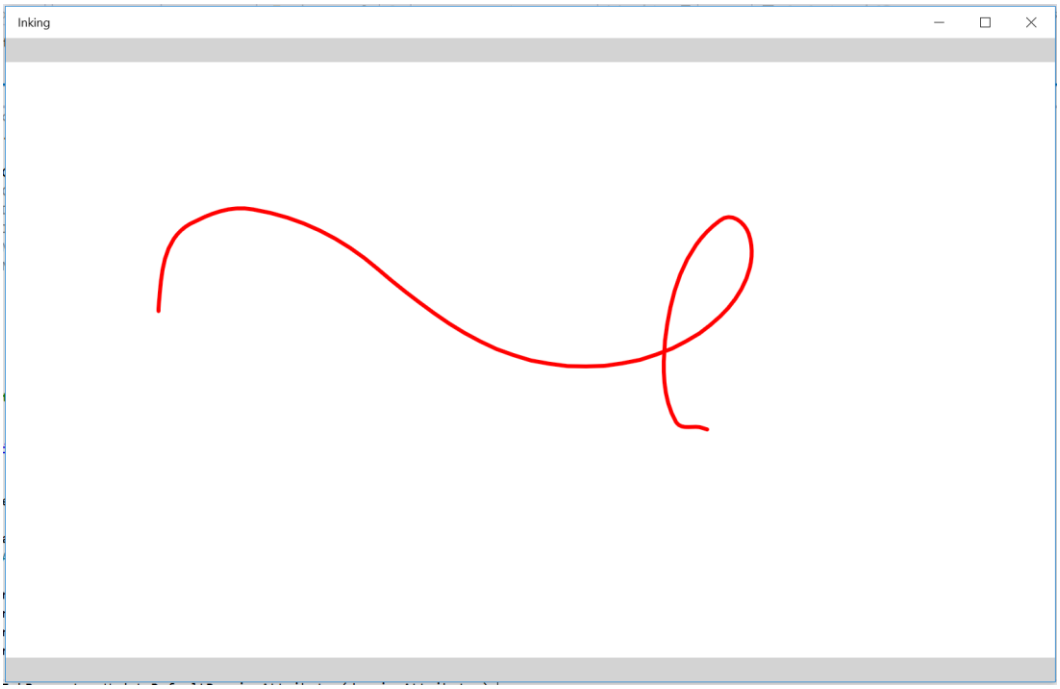

Рисунок 5

Более широкий рукописный штрих красного цвета.

3. Завершите отладку и вернитесь в Visual Studio.

4. Добавьте набор кнопок выбора цвета в StackPanel в первом ряду сетки MainPage. Установите событие Click для каждого **OnPenColorChanged** и стиль для **ColorButtonStyle StaticResource**. В одном из следующих этапов вы создадите обработчик события и ColorButtonStyle.

XAML

```
<StackPanel Grid.Column="0" Orientation="Horizontal">
    <TextBlock Text="Color" Margin="0,0,12,0" VerticalAlignment="Center" />
    <Button Background="Crimson"    Foreground="Crimson"
Click="OnPenColorChanged" Style="{StaticResource ColorButtonStyle}"
Margin="0,0,3,0"/>
    <Button Background="Orange"  Foreground="Orange"
Click="OnPenColorChanged" Style="{StaticResource ColorButtonStyle}"
Margin="0,0,3,0"/>
    <Button Background="Gold"   Foreground="Gold"
Click="OnPenColorChanged" Style="{StaticResource ColorButtonStyle}"
Margin="0,0,3,0"/>
    <Button Background="LimeGreen" Foreground="LimeGreen"
Click="OnPenColorChanged" Style="{StaticResource ColorButtonStyle}"
Margin="0,0,3,0"/>
    <Button Background="DeepSkyBlue" Foreground="DeepSkyBlue"
Click="OnPenColorChanged" Style="{StaticResource ColorButtonStyle}"
Margin="0,0,3,0"/>
    <Button Background="MediumOrchid" Foreground="MediumOrchid"
Click="OnPenColorChanged" Style="{StaticResource ColorButtonStyle}"
Margin="0,0,3,0"/>
```

```
    <Button Background="Black" Foreground="Black" Click="OnPenColorChanged"
Style="{StaticResource ColorButtonStyle}" Margin="0,0,3,0"/>
</StackPanel>
```

5. Добавьте стили в элемент **Grid.Resources** для установки высоты, ширины, полей, размера шрифта и заполнения для кнопок.

XAML

```
<Grid Background="LightGray">
    <Grid.Resources>
        <Style x:Key="ColorButtonStyle"  TargetType="Button">
            <Setter Property="MinWidth" Value="28"/>
            <Setter Property="MinHeight" Value="28"/>
            <Setter Property="Width" Value="28"/>
            <Setter Property="Height" Value="28"/>
            <Setter Property="Margin" Value="0,0,0,0"/>
            <Setter Property="FontSize" Value="0"/>
            <Setter Property="Padding" Value="0,0,0,0"/>
        </Style>
    </Grid.Resources>
```

6. В выделенном коде добавьте обработчик событий **OnPenColorChanged**. Этот метод копирует графические атрибуты по умолчанию из InkCanvas и устанавливает цвет чернил в соответствии с фоновым цветом кнопки.

C#

```
private void OnPenColorChanged(object sender, RoutedEventArgs e)
{
    if (InkCanvas != null)
        {
            InkDrawingAttributes drawingAttributes =
InkCanvas.InkPresenter.CopyDefaultDrawingAttributes();

            // Используйте фон кнопки, чтобы установить цвет нового пера
            var btnSender = sender as Button;
            var brush = btnSender.Background as SolidColorBrush;

            drawingAttributes.Color = brush.Color;
            InkCanvas.InkPresenter
 .UpdateDefaultDrawingAttributes(drawingAttributes);
    }
}
```

7. Скомпилируйте и запустите приложение. Используйте кнопки выбора цвета для изменения цвета чернил во время рисования.

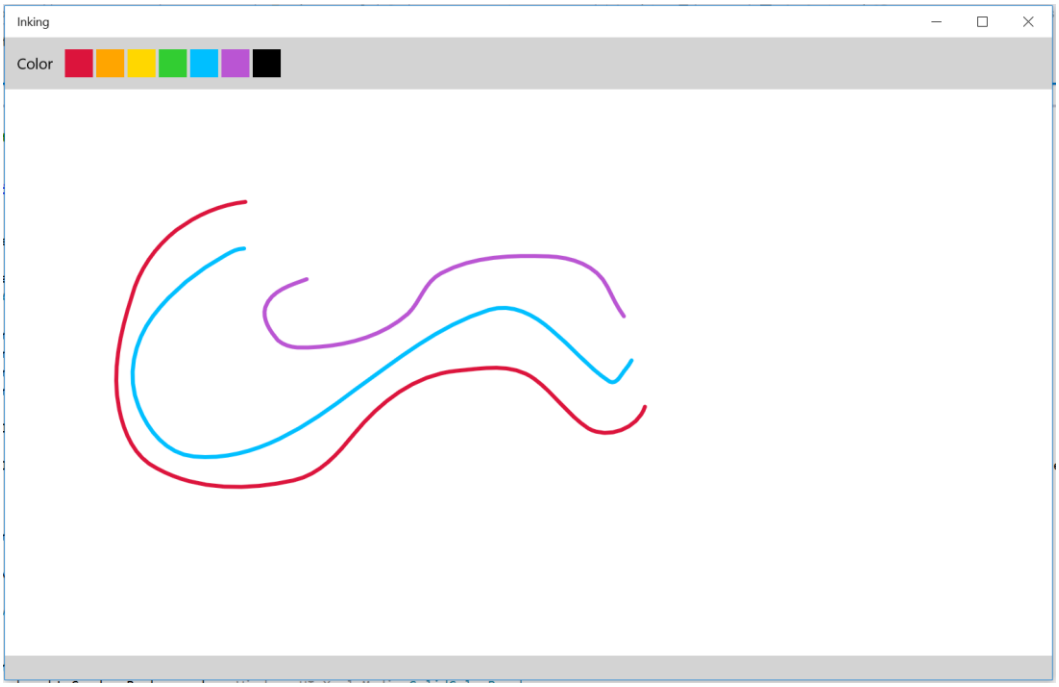

Рисунок 6

Замените цвет чернил на выбранные пользователем цвета.

8. Завершите отладку и вернитесь в Visual Studio.

Примечание: UI InkToolbar, используемый в Edge Browser Web Notes, доступен в виде расширения Visual Studio. Панель инструментов содержит в готовом виде палитру, выбор наконечников пера и режимы чернил, маркеров и ластиков. Дополнительную информацию по панели инструментов рукописного стиля см. по ссылке https://visualstudiogallery.msdn.microsoft.com/58194dfe-df44-4c4e-893a-1eca40675269

Задача 4 – Стереть и очистить

Сейчас, когда вы умеете настраивать штрихи, поговорим про стирание нарисованного. Вы можете стирать одиночные штрихи или очищать весь холст сразу.

1. **В MainPage.xaml** добавьте чекбокс внутрь **StackPanel**, который находится во втором столбце основной сетки. Отмеченная графа появится справа от кнопок выбора цвета, чтобы переключить режим стирания. Настройте события Checked и Unchecked на ErasingModeCheckBox_Checked и ErasingModeCheckBox_Unchecked соответственно. На следующем этапе вы создадите обработчики.

XAML

```
<StackPanel Grid.Column="1" Orientation="Horizontal"
HorizontalAlignment="Right">
    <CheckBox Content="Enable Erasing Mode" Margin="20,0,4,0"
        Checked="ErasingModeCheckBox_Checked"
        Unchecked="ErasingModeCheckBox_Unchecked"/>
```

```
</StackPanel>
```

2. Создайте обработчики для событий в коде страницы. В отмеченном событии **InputProcessingConfiguration** настраивается на режим **Erasing (стирания)**.

C#

```
private void ErasingModeCheckBox_Checked(object sender, RoutedEventArgs e)
{
    InkCanvas.InkPresenter.InputProcessingConfiguration.Mode =
InkInputProcessingMode.Erasing
}

private void ErasingModeCheckBox_Unchecked(object sender, RoutedEventArgs
e)
{
    InkCanvas.InkPresenter.InputProcessingConfiguration.Mode =
InkInputProcessingMode.Inking;
}
```

Примечание: Штрих, сделанный в режиме стирания, стирает любой рукописный штрих, с которым он пересекается. Весь штрих сотрется.

3. Скомпилируйте и запустите своё приложение. Рисуя в режиме обведения чернилами, запустите режим стирания путем проставления галочки в соответствующем окне. Перечеркните один из сделанных вами штрихов, чтобы увидеть, как он исчезает.

4. Остановите отладку и вернитесь в Visual Studio.

5. **В MainPage.xaml** добавьте кнопку **Очистить** в третьем ряду **StackPanel**. Настройте Click-событие на **OnClear**. На следующем этапе вы создадите обработчик события.

XAML

```
<StackPanel Grid.Row="2" Orientation="Horizontal" Margin="12">
    <Button Content="Clear" Click="OnClear" />
</StackPanel>
```

6. Добавьте обработчик **OnClear()** в код **MainPage**.

C#

```
void OnClear(object sender, RoutedEventArgs e)
{
    InkCanvas.InkPresenter.StrokeContainer.Clear();
}
```

7. Скомпилируйте и запустите приложение. Нарисуйте несколько штрихов и используйте кнопку **Очистить**, чтобы убрать их из InkCanvas в одно касание.

Рисунок 7

Режим стирания позволяет вам убирать одиночные штрихи, в то время как кнопка ***Очистить*** *стирает всё, что вы нарисовали.*

8. Завершите отладку и вернитесь в Visual Studio.

Задача 5 – Сохранить и загрузить

Режим работы с чернилами полезен тем, что он позволяет преобразовывать физические действия пользователя в пиксельные изображения на устройстве. Для повторного использования созданной информации, вы можете сохранить штрихи из InkCanvas и повторно загрузить их позднее. В этой задаче вы сохраните свои рукописные штрихи в GIF-файле и повторно загрузите их на полотно.

1. Добавьте кнопки **Сохранить** и **Загрузить** возле кнопки **Очистить** в MainPage.xaml. Настройте события щелчка на **OnSaveAsync** и **OnLoadAsync**, соответственно.

XAML

```
<StackPanel Grid.Row="2" Orientation="Horizontal" Margin="12">
    <Button Content="Clear" Click="OnClear" />
    <Button Content="Save" Margin="12,0,0,0" Click="OnSaveAsync"/>
    <Button Content="Load" Margin="12,0,0,0" Click="OnLoadAsync"/>
</StackPanel>
```

2. Во коде MainPage добавьте пространства имен **Windows.Storage** и **Windows.Storage.Streams**.

C#

```
using Windows.Storage;
```

```
using Windows.Storage.Streams;
```

3. Добавьте методы **OnSaveAsync** и **OnLoadAsync** в код страницы MainPage. Метод **OnSaveAsync** сохраняет штрихи в виде GIF-файла. Метод загрузки считывает эти штрихи обратно на InkCanvas.

```
C#
async void OnSaveAsync(object sender, RoutedEventArgs e)
{
    // Мы не хотим сохранять пустой файл
    if (InkCanvas.InkPresenter.StrokeContainer.GetStrokes().Count > 0)
    {
        var savePicker = new Windows.Storage.Pickers.FileSavePicker();
        savePicker.SuggestedStartLocation =
Windows.Storage.Pickers.PickerLocationId.PicturesLibrary;
        savePicker.FileTypeChoices.Add(
            "Gif with embedded ISF",
            new System.Collections.Generic.List<string> { ".gif" });

        Windows.Storage.StorageFile file =
            await savePicker.PickSaveFileAsync();
        if (null != file)
        {
            using (IRandomAccessStream stream =
                await file.OpenAsync(FileAccessMode.ReadWrite))
            {
                await InkCanvas.InkPresenter.StrokeContainer.SaveAsync(
                    stream);
            }
        }
    }
}

async void OnLoadAsync(object sender, RoutedEventArgs e)
{
    var openPicker = new Windows.Storage.Pickers.FileOpenPicker();
    openPicker.SuggestedStartLocation =
       Windows.Storage.Pickers.PickerLocationId.PicturesLibrary;
    openPicker.FileTypeFilter.Add(".gif");
    openPicker.FileTypeFilter.Add(".isf");
    Windows.Storage.StorageFile file = await
openPicker.PickSingleFileAsync();
    if (null != file)
    {
        using (var stream = await file.OpenSequentialReadAsync())
        {
            await InkCanvas.InkPresenter.StrokeContainer.LoadAsync(stream);
        }
    }
}
```

4. Скомпилируйте и запустите приложение. Рисуя на холсте, сохраните сделанные вами штрихи в виде GIF-файла в своей файловой системе. В процессе работы приложения очистите холст и используйте кнопку **Загрузка**, чтобы открыть браузер файлов. Выберите свой GIF-файл, чтобы перезагрузить штрихи обратно на InkCanvas.

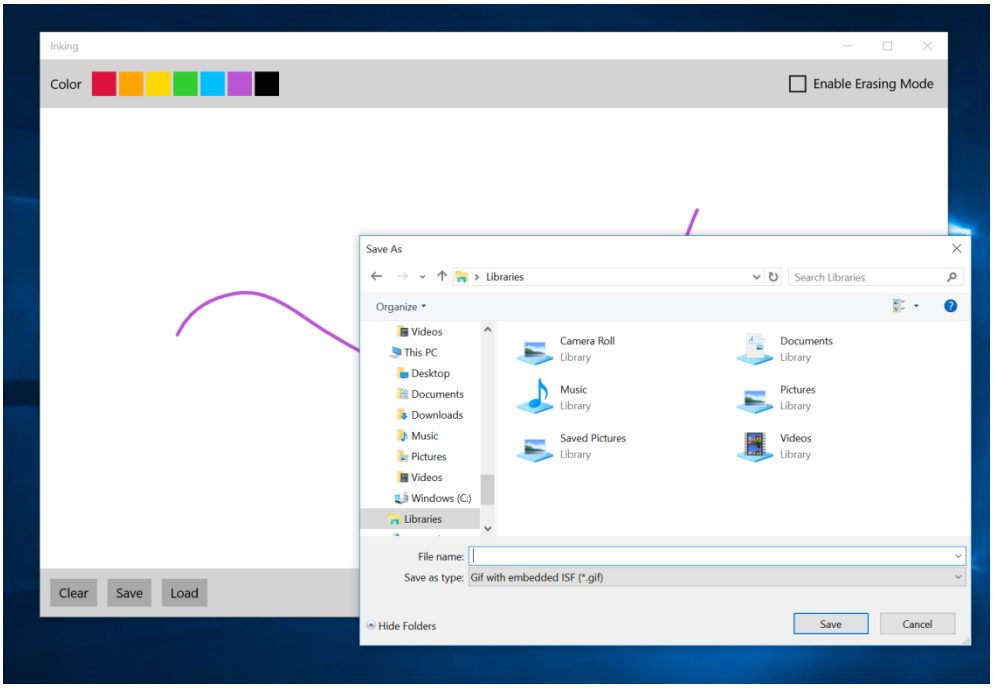

Рисунок 8

Сохраните сделанные вами штрихи в GIF-файле.

5. Остановите отладку и вернитесь в Visual Studio.

Упражнение 2: Распознавание рукописных текстов

Приложения с поддержкой рукописного ввода обычно включает чертёжи, аннотации к изображениям и функцию распознавания рукописных текстов. В этом упражнении мы рассмотрим распознавание рукописных текстов с InkCanvas, используя специфические для языка распознаватели, доступные на вашем устройстве.

Задача 1 – Создайте новое решение для проекта Распознавания рукописных текстов

Создайте новый проект для Распознавания рукописных текстов.

1. В новой версии Visual Studio 2015 выберите **File (Файл) -> New (Новый) -> Project (Проект)**, чтобы открыть диалоговое окно New Project (Новый проект). Далее **Installed**

(Установленное) > Templates (Шаблоны) > Visual C# > Windows > Universal, а затем выберите шаблон **Blank App** приложения (Universal Windows). Назовите проект **Handwriting** (Рукописный текст). Сохраните проект в папку, в которой вы храните свои работы по настоящему курсу.

2. Настройте Текущую конфигурацию решения на **отладку** и Платформу решений в соответствии с **x86**. Выберите **Локальный компьютер** из выпадающего меню Цели отладки.

3. Скомпилируйте и запустите своё приложение. Вы увидите окно шаблона приложения со счетчиком частоты кадров, активированном по умолчанию в режиме отладки.

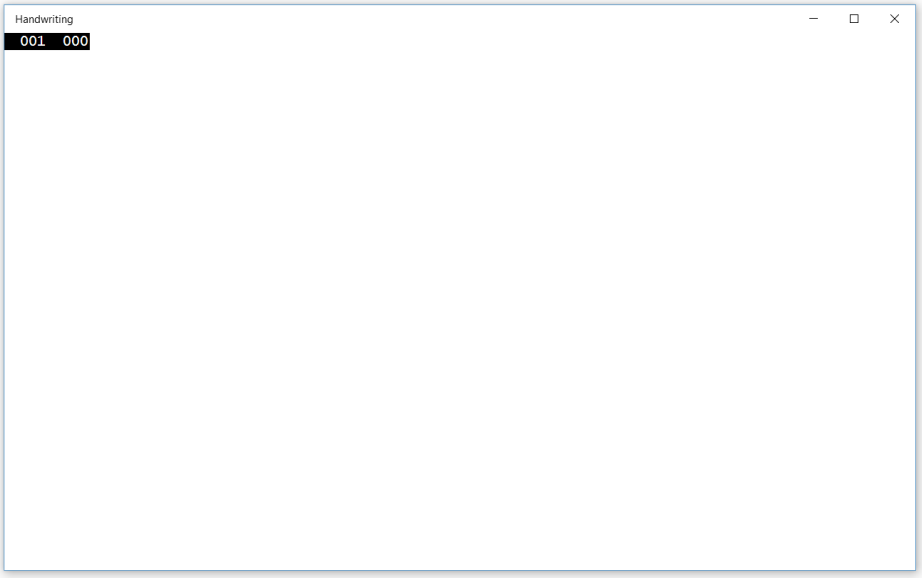

Рисунок 9

Пустое универсальное приложение, запущенное в режиме настольного компьютера.

Примечание: Директива препроцессора для активации или отключения счётчика частоты кадров находится в **App.xaml.cs**. Счётчик частоты кадра может перекрывать содержимое вашего приложения, если вы оставляете его включённым. Для целей настоящего курса вы можете отключить его, установив значение **this.DebugSettings.EnableFrameRateCounter** в **false**.

4. Вернитесь в Visual Studio и остановите отладку.

Задача 2 – Создайте дизайн окна

MainPage для этого упражнения будет выглядеть так же, как и для проекта в упражнении 1. В этой задаче мы создадим макет и установим InkCanvas.

1. Измените фон Grid на LightGray (светло-серый). Создайте такой же макет, который вы использовали в упражнении 1, используя определения строк, столбцов и элементы управления. Добавьте InkCanvas ко второй строке и назовите его с помощью **x:Name**.

XAML

```
<Grid Background="LightGray">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="2*" />
        <RowDefinition Height="Auto" />
    </Grid.RowDefinitions>
    <Grid Grid.Row="0" Margin="12">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="2*" />
            <ColumnDefinition Width="1*" />
        </Grid.ColumnDefinitions>
        <StackPanel Orientation="Horizontal" Grid.Column="0">
        </StackPanel>
        <StackPanel Grid.Column="1" HorizontalAlignment="Right"
            Orientation="Horizontal">
        </StackPanel>
    </Grid>
    <Grid Grid.Row="1" Background="White" Grid.ColumnSpan="2">
            <InkCanvas x:Name="InkCanvas" />
    </Grid>
    <StackPanel Grid.Row="2" Orientation="Horizontal" Margin="12">
    </StackPanel>
</Grid>
```

2. Добавьте пространство имен **Windows.UI.Input.Inking** в выделенный код.

C#

```
using Windows.UI.Input.Inking;
```

3. Инициализируйте графические атрибуты в конструкторе MainPage.

C#

```
public MainPage()
{
    this.InitializeComponent();

    InkDrawingAttributes drawingAttributes = new InkDrawingAttributes();
    drawingAttributes.Color = Windows.UI.Colors.Black;
    drawingAttributes.Size = new Size(4, 4);
    drawingAttributes.IgnorePressure = false;
    drawingAttributes.FitToCurve = true;

    InkCanvas.InkPresenter.UpdateDefaultDrawingAttributes(
        drawingAttributes);
    InkCanvas.InkPresenter.InputDeviceTypes =
        Windows.UI.Core.CoreInputDeviceTypes.Mouse |
        Windows.UI.Core.CoreInputDeviceTypes.Pen |
        Windows.UI.Core.CoreInputDeviceTypes.Touch;
}
```

4. Скомпилируйте и запустите своё приложение. Вы увидите простые возможности режима обведения чернилами, аналогичные тем, которые мы делали в предыдущем упражнении.

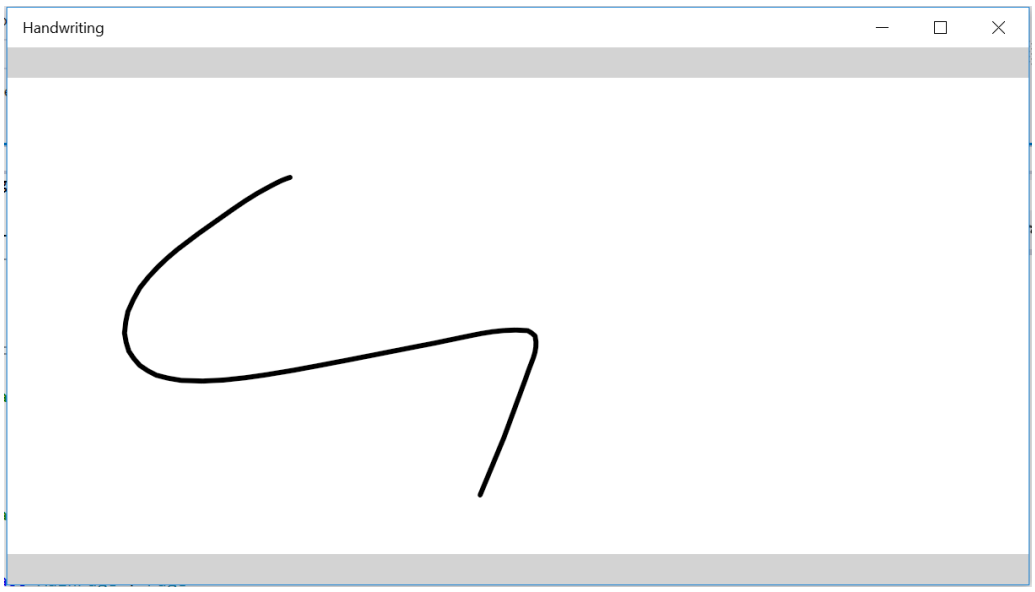

Рисунок 10
Активация приложения распознавания рукописных текстов.

5. Завершите отладку и вернитесь в Visual Studio.

Задача 3 – Настройте распознаватель чернил

InkRecognizer отвечает за все этапы распознавания рукописных текстов. В этой задаче вы настроите словарь RecognizerHelper и InkRecognizer.

1. Щёлкните правой кнопкой мыши по наименованию проекта **Рукописного текста** и выберите Add **(Добавить) > Existing Item (Существующий элемент)**. Найдите в папке **Lab Assets** и добавьте в проект файл **RecognizerHelper.cs**.

2. Откройте **RecognizerHelper.cs**. В нем описан вспомогательный класс, содержащий словарь распознавателей, которые могут быть установлены на вашем устройстве. Он переводит теги языка распознавания на более длинные, удобочитаемые имена, чтобы помочь пользователям идентифицировать распознаватель, который они выбрали. Сохраните и закройте вспомогательный элемент.

3. Добавьте метки TextBlock и ComboBox к **StackPanel** в строке 0, столбце 0 в MainPage.xaml.

XAML

```
<StackPanel Orientation="Horizontal" Grid.Column="0">
    <TextBlock Text="Available Recognizers:" Margin="0,8"/>
    <ComboBox
        x:Name="RecoName"
```

```
        MaxWidth="500"
        SelectionChanged="OnRecognizerChanged">
    </ComboBox>
</StackPanel>
```

4. Создайте строку состояния в третьей строке, используя **TextBlock**. Текст состояния отобразит результаты распознавания рукописных текстов или сообщение, если в системе не было обнаружено никаких распознавателей.

XAML

```
<StackPanel Grid.Row="2" Orientation="Horizontal" Margin="12">
    <TextBlock x:Name="Status" Margin="20,0,0,0" />
</StackPanel>
```

5. Добавьте пространство имен **Windows.Globalization** в код **MainPage**.

C#

```
using Windows.Globalization;
```

6. Создайте экземпляры **InkRecognizers** и список **InkRecognizerContainer**, доступный только для чтения.

C#

```
Рукописный текст пространства имён
{
    public sealed partial class MainPage : Page
    {
        InkRecognizerContainer inkRecognizerContainer = null;
        private IReadOnlyList<InkRecognizer> recoView = null;
```

7. Загрузите доступные распознаватели системы в конструкторе MainPage.

C#

```
public MainPage()
{
    this.InitializeComponent();
    InkDrawingAttributes drawingAttributes = new
        InkDrawingAttributes();
    drawingAttributes.Color = Windows.UI.Colors.Black;
    drawingAttributes.Size = new Size(4, 4);
    drawingAttributes.IgnorePressure = false;
    drawingAttributes.FitToCurve = true;

    inkRecognizerContainer = new InkRecognizerContainer();
    recoView = inkRecognizerContainer.GetRecognizers();
    if (recoView.Count > 0)
    {
        foreach (InkRecognizer recognizer in recoView)
        {
            RecoName.Items.Add(recognizer.Name);
        }
```

```
    }
    else
    {
        RecoName.IsEnabled = false;
        RecoName.Items.Add("No Recognizer Available");
    }
    RecoName.SelectedIndex = 0;

    InkCanvas.InkPresenter.UpdateDefaultDrawingAttributes(
        drawingAttributes);
    InkCanvas.InkPresenter.InputDeviceTypes =
    Windows.UI.Core.CoreInputDeviceTypes.Mouse |
    Windows.UI.Core.CoreInputDeviceTypes.Pen |
    Windows.UI.Core.CoreInputDeviceTypes.Touch;
}
```

8. Создайте обработчик **OnRecognizerChanged** для выбора значения распознавателя из ComboBox и метод **SetRecognizerByName** для установки распознавателя по имени.

C#

```
void OnRecognizerChanged(object sender, RoutedEventArgs e)
{
    string selectedValue = (string)RecoName.SelectedValue;
    SetRecognizerByName(selectedValue);
}

bool SetRecognizerByName(string recognizerName)
{
    bool recognizerFound = false;

    foreach (InkRecognizer reco in recoView)
    {
        if (recognizerName == reco.Name)
        {
            inkRecognizerContainer.SetDefaultRecognizer(reco);
            recognizerFound = true;
            break;
        }
    }

    if (!recognizerFound)
    {
        Status.Text = "Could not find target recognizer.";
    }

    return recognizerFound;
}
```

9. Скомпилируйте и запустите приложение. Используйте ComboBox для просмотра списка распознавателей, установленных в системе.

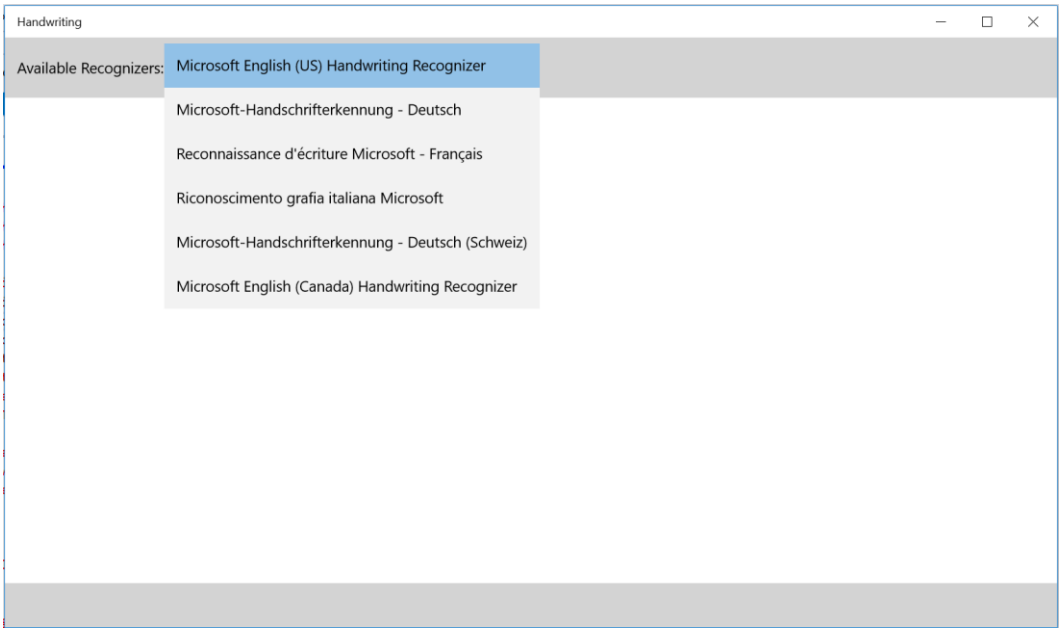

Рисунок 11

ComboBox предлагает распознаватели, доступные на устройстве.

10. Остановите отладку и вернитесь в Visual Studio.

Задача 4 – Примените OnRecognizeAsync()

Теперь, когда настроены распознаватели, вы можете осуществить распознавание рукописного текста в своём приложении.

1. Добавьте кнопки **Распознать** и **Очистить** ко второму столбцу первой строки в MainPage.

XAML

```
<StackPanel Grid.Column="1" HorizontalAlignment="Right"
Orientation="Horizontal">
    <Button x:Name="RecognizeBtn" Content="Recognize" Width="100"
        Margin="0,0,4,0" Click="OnRecognizeAsync"/>
    <Button x:Name="ClearBtn" Content="Clear" Width="65" Margin="0,0,4,0"
        Click="OnClear"/>
</StackPanel>
```

2. Добавьте обработчик **OnClear()** в код MainPage. Этот метод идентичен методу **OnClear**(), который мы использовали в упражнении 1.

C#

```
void OnClear(object sender, RoutedEventArgs e)
{
    InkCanvas.InkPresenter.StrokeContainer.Clear();
}
```

3. Добавьте обработчик **OnRecognizeAsync()** в выделенный код. Этот метод асинхронно распознает штрихи, нанесенные в InkCanvas, с использованием выбранного

распознавателя. Если будут результаты, они отображаются в строке состояния. Если результаты будут ошибочными, вы получите сообщение **нет распознанных текстов**.

C#

```
async void OnRecognizeAsync(object sender, RoutedEventArgs e)
{
    IReadOnlyList<InkStroke> currentStrokes =
InkCanvas.InkPresenter.StrokeContainer.GetStrokes();
    if (currentStrokes.Count > 0)
    {
        RecognizeBtn.IsEnabled = false;
        ClearBtn.IsEnabled = false;
        RecoName.IsEnabled = false;

        var recognitionResults = await
inkRecognizerContainer.RecognizeAsync(
            InkCanvas.InkPresenter.StrokeContainer,
            InkRecognitionTarget.All);

        if (recognitionResults.Count > 0)
        {
            // Показать результат распознавания
            string str = "Recognition result:";
            foreach (var r in recognitionResults)
            {
                str += " " + r.GetTextCandidates()[0];
            }
            Status.Text=str;
        }
        else
        {
                    Status.Text = "No text recognized.";
        }

        RecognizeBtn.IsEnabled = true;
        ClearBtn.IsEnabled = true;
        RecoName.IsEnabled = true;
    }
    else
    {
        Status.Text="Must first write something.";
    }
}
```

4. Скомпилируйте и запустите приложение. Выберите распознаватель из списка и запишите слово из соответствующего языка на полотне. Используйте кнопку распознавания для просмотра результатов

Рисунок 12

Успешное распознавание рукописного текста.

5. Завершите отладку и вернитесь в Visual Studio.

Краткий обзор

Чернила – естественный и удобный способ взаимодействия с пользователем на устройствах Windows 10, которые имеют опции ввода данных с помощью мыши, прикосновений или пера. Рисование чернилами быстро и легко реализовать в приложениях UWP. В этой работе мы рассмотрели выбор настройки атрибутов чернил и стилей, очистки и стирания с полотна и сохранения и загрузки штрихов. Мы также применили распознавание рукописных текстов для языковых пакетов, установленных на устройстве.