

Лабораторный­ практикум

Курс: Обведение чернилами

Сентябрь 2015 года

Содержание

**No table of contents entries found.**

Обзор

Обведение чернилами является мощным средством преобразования естественного движения в пиксели. Поскольку все больше устройств поддерживает сенсорные экраны, то обведение чернилами играет большую роль во взаимодействиях пользователей. Традиционно стилус является основным устройством ввода чернил. В UWP, стилусе, касании и мышке Xbox обведение чернилами может быть внедрено всего с несколькими строками кода. InkCanvas упростил работу, которая сводится к применению чернил в вашем приложении.

В этом курсе вы создадите InkCanvas, который принимает ввод данных через мышку, прикосновение 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

* 1. Пройдя этот курс, вы сможете:
  + Создавать InkCanvas в приложении UWP
  + Применять для ввода данных стилус, прикосновение и мышь
  + Изменять цвет чернил по выбору пользователя
  + Стирайте единочные штрихи
  + Очищать холст
  + Сохранять и загружать штрихи
  + Распознавать рукописный текст

# Системные требования

* 1. Чтобы выполнить этот курс, необходимо обладать следующим набором программных инструментов:
  + Microsoft Windows 10
  + Microsoft Visual Studio 2015

# Настройка

* 1. Вам следует выполнить следующие действия для подготовки компьютера:
  2. Установить Microsoft Windows 10.
  3. Установить Microsoft Visual Studio 2015.

# Упражнения

* 1. Этот лабораторный курс включает следующие упражнения:
  2. Рисование с помощью InkCanvas
  3. Распознавание рукописных текстов
  4. Расчетное время для завершения практикума: **45-60 минут**.

Упражнение 1: Рисование с помощью InkCanvas

1. 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 **"Обведение чернилами"** и выберите местоположение файловой системы, в котором будет осуществлено хранение результатов прохождения Лабораторного курса. Мы создали папку на диске **C** и переименовали ее в **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 обновлять версию своей работы или использовать инструмент Application Insights. Нажмите **OK** для создания проекта.

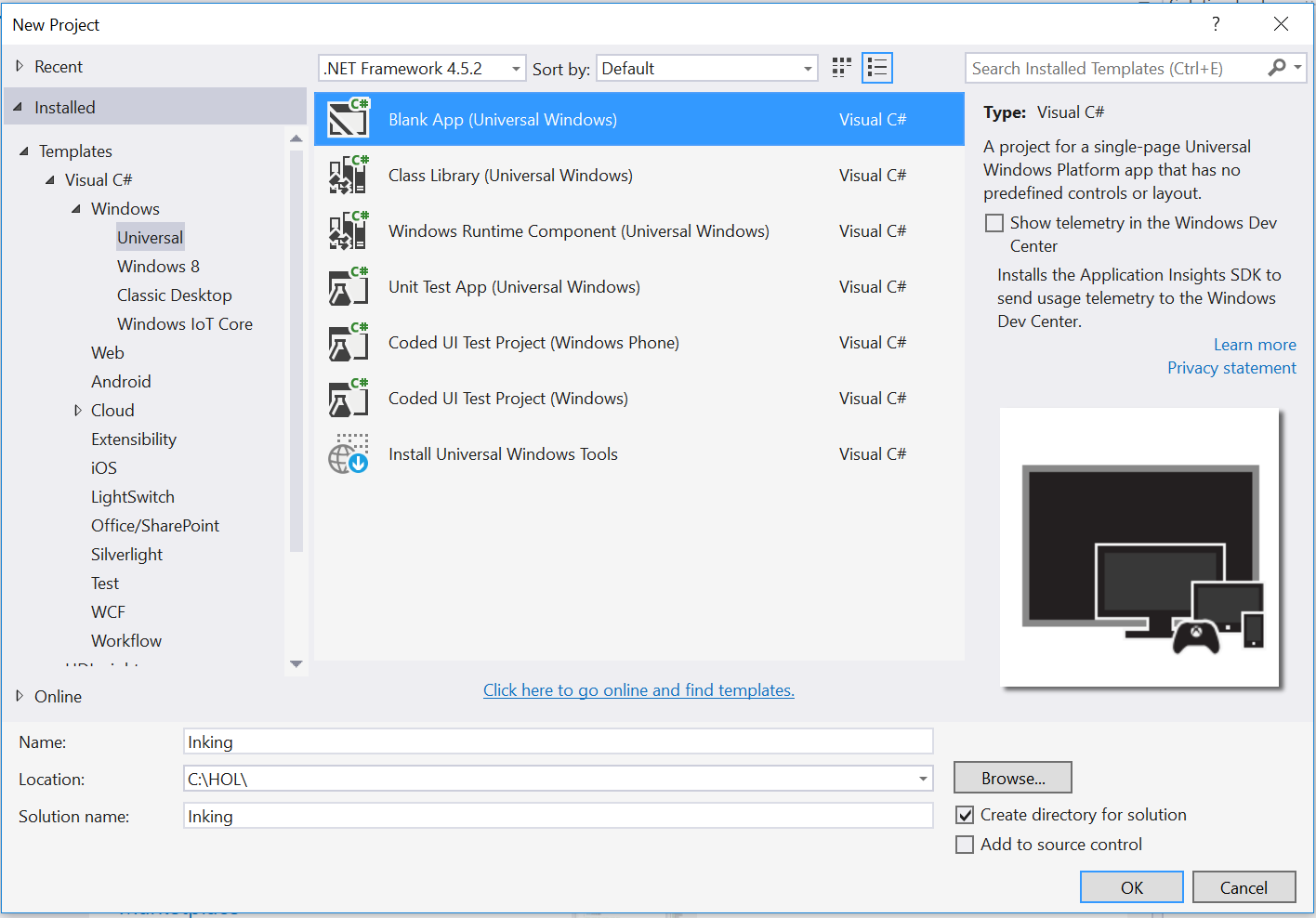


Рисунок 1

Создание нового проекта приложения в Visual Studio 2015.

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

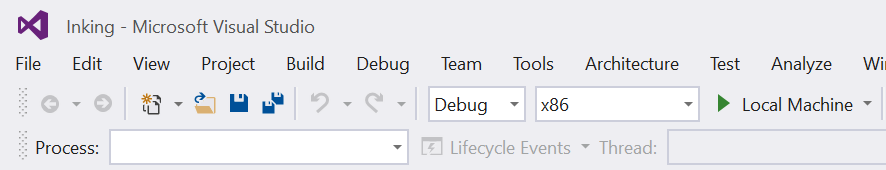


Рисунок 2

* + 1. Сконфигурируйте свое приложение таким образом, чтобы оно запускалось на Локальном компьютере.

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

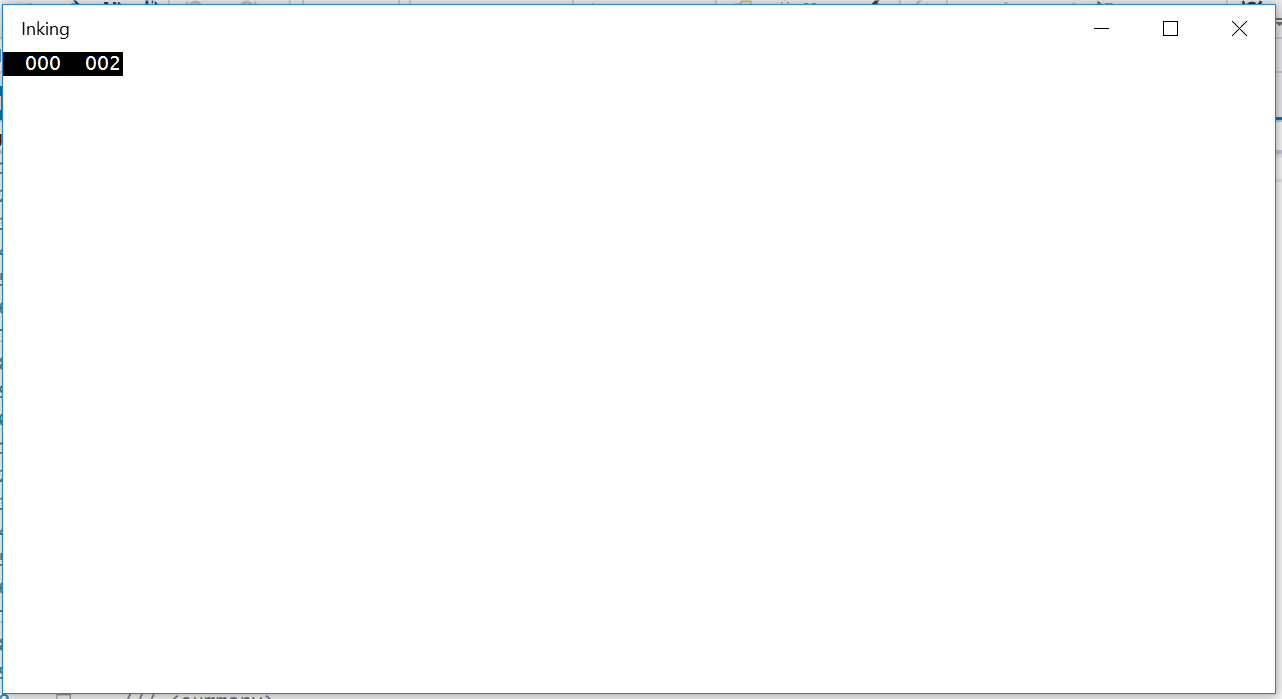


Рисунок 3

Пустое универсальное приложение, запущенное в режиме настольного компьютера.

* 1. **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.
  2. 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 настоящих курсов вы можете отключить его, установив значение **DebugSettings.EnableFrameRateCounter** как **ложное**.

1. Вернитесь в Visual Studio и отключите отладку.

Задача 2 – Создайте InkCanvas

* 1. Вы можете начать обведение чернилами в своём приложении в нескольких строках кода. В этой задаче мы установим структуру оформления для поддержки большого полотна чернил со строками выше и ниже его для хранения опций и кнопок стиля.

1. Откройте **MainPage.xaml**. Мы настроили фон в **Сетке** на **LightGray (Светло-серый)** и добавили **RowDefinitions (Определения строк)** для трёх строк. Верхние и нижние строки будут содержать опции и кнопки стиля, и в большой центральной строке будет находиться InkCanvas.
   * 1. XAML
   1. <Grid Background="LightGray">
   2. <Grid.RowDefinitions>
   3. <RowDefinition Height="Auto" />
   4. <RowDefinition Height="2\*" />
   5. <RowDefinition Height="Auto" />
   6. </Grid.RowDefinitions>
   7. </Grid>
2. Добавьте структуры к этим трём строкам. Первая строка будет содержать два столбца, каждый будет поддерживать горизонтальную StackPanel. Вторая строка будет содержать **InkCanvas**, и третья строка будет состоять из другой StackPanel. Назовите свой InkCanvas атрибутом  **x:Name**.
   * 1. XAML
   1. <Grid Background="LightGray">
   2. <Grid.RowDefinitions>
   3. <RowDefinition Height="Auto" />
   4. <RowDefinition Height="2\*" />
   5. <RowDefinition Height="Auto" />

</Grid.RowDefinitions>

* 1. <Grid Grid.Row="0" Margin="12">
  2. <Grid.ColumnDefinitions>
  3. <ColumnDefinition Width="1\*" />
  4. <ColumnDefinition Width="1\*" />
  5. </Grid.ColumnDefinitions>
  6. <StackPanel Grid.Column="0" Orientation="Horizontal">
  7. </StackPanel>
  8. <StackPanel Grid.Column="1" Orientation="Horizontal" HorizontalAlignment="Right">
  9. </StackPanel>
  10. </Grid>
  11. <Grid Grid.Row="1" Background="White" Grid.ColumnSpan="2">
  12. <InkCanvas x:Name="InkCanvas" />
  13. </Grid>
  14. <StackPanel Grid.Row="2" Orientation="Horizontal" Margin="12">
  15. </StackPanel>
  16. </Grid>

1. На этом этапе ваше приложение готово к простому обведению чернилами с помощью стилуса. Если ваша проявочная машина не поддерживает стилус, мы предоставим другие устройства ввода.
2. Во вспомогательном коде MainPage добавьте пространство имени **Windows.UI.Input.Inking.**
   1. **Примечание:** InkCanvas принимает только ввод стилусом по умолчанию. Вы можете использовать мышку или прикосновение в качестве устройств дополнительного ввода.
3. Добавить **мышку** и **прикосновение**, как типы устройств ввода для **InkCanvas**.
   * 1. C#
   1. public MainPage()
   2. {
   3. this.InitializeComponent();
   4. InkCanvas.InkPresenter.InputDeviceTypes =
   5. Windows.UI.Core.CoreInputDeviceTypes.Mouse |   
       Windows.UI.Core.CoreInputDeviceTypes.Pen |   
       Windows.UI.Core.CoreInputDeviceTypes.Touch;
   6. }
   7. Примечание: InkCanvas является элементом управления, содержащим одно свойство обведения чернилами– InkPresenter. InkPresenter отображает чернила на полотне и обрабатывает атрибуты для нанесения штрихов.
4. Скомпилируйте и запустите своё приложение. Сделайте рисунок на белом **InkCanvas** с помощью мышки, стилуса или прикосновения. Вы увидите цвет чернил по умолчанию и атрибуты размеров пера.

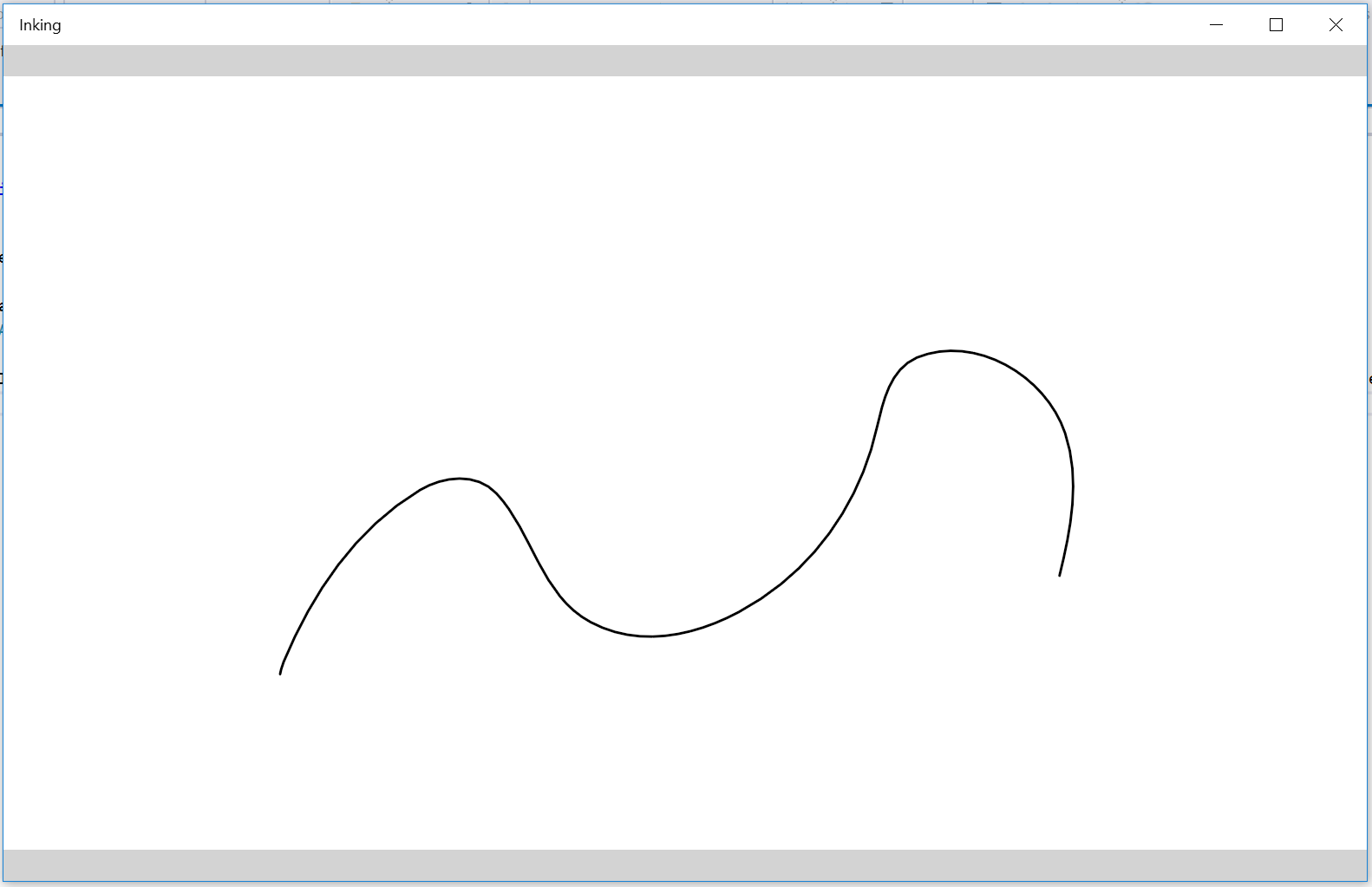


Рисунок 4

Графические атрибуты по умолчанию для InkCanvas.

1. Отключите отладку и вернитесь в Visual Studio.

Задача 3 – Настройте графические атрибуты

Если вы захотите использовать один стиль рукописного фрагмента, вы сможете настраивать графические атрибуты для **InkCanvas**, при создании экземпляра **InkDrawingAttributes**. Вы также можете по выбору пользователей программным путём обновить атрибуты . В этой задаче вы сначала будете добавлять атрибуты по умолчанию в стиль рукописного фрагмента, затем добавлять кнопки, чтобы моментально изменить атрибуты.

1. Вы можете установить атрибуты сразу при создании экземпляра **InkDrawingAttributes**. Добавьте атрибуты **Цвета**, **Размера**, **Надавливания** и **FitToCurve (Подогнать под кривую)** в конструктор **MainPage (Главной страницы)**. После установки обновите графические атрибуты на InkCanvas, войдя в свой экземпляр **графических атрибутов**.
   * 1. C#
   1. public MainPage()
   2. {
   3. this.InitializeComponent();
   4. InkDrawingAttributes drawingAttributes = new InkDrawingAttributes();
   6. drawingAttributes.Color = Windows.UI.Colors.Red;
   7. drawingAttributes.Size = new Size(4, 4);
   8. drawingAttributes.IgnorePressure = false;
   9. drawingAttributes.FitToCurve = true;
   10. InkCanvas.InkPresenter.UpdateDefaultDrawingAttributes(drawingAttributes);
   11. InkCanvas.InkPresenter.InputDeviceTypes =
   12. Windows.UI.Core.CoreInputDeviceTypes.Mouse |   
        Windows.UI.Core.CoreInputDeviceTypes.Pen |   
        Windows.UI.Core.CoreInputDeviceTypes.Touch;
   13. }
   14. **Примечание:** Хотя "IgnorePressure = ложно" является опцией графических атрибутов чертежа, **InkCanvas** на данный момент не отображает чувствительные к надавливанию штрихи. Вместо этого значение надавливания фиксируется в каждой точке.
   15. 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 своё приложение. В этот раз ваш рукописный стиль отобразит более широкий штрих красного цвета.

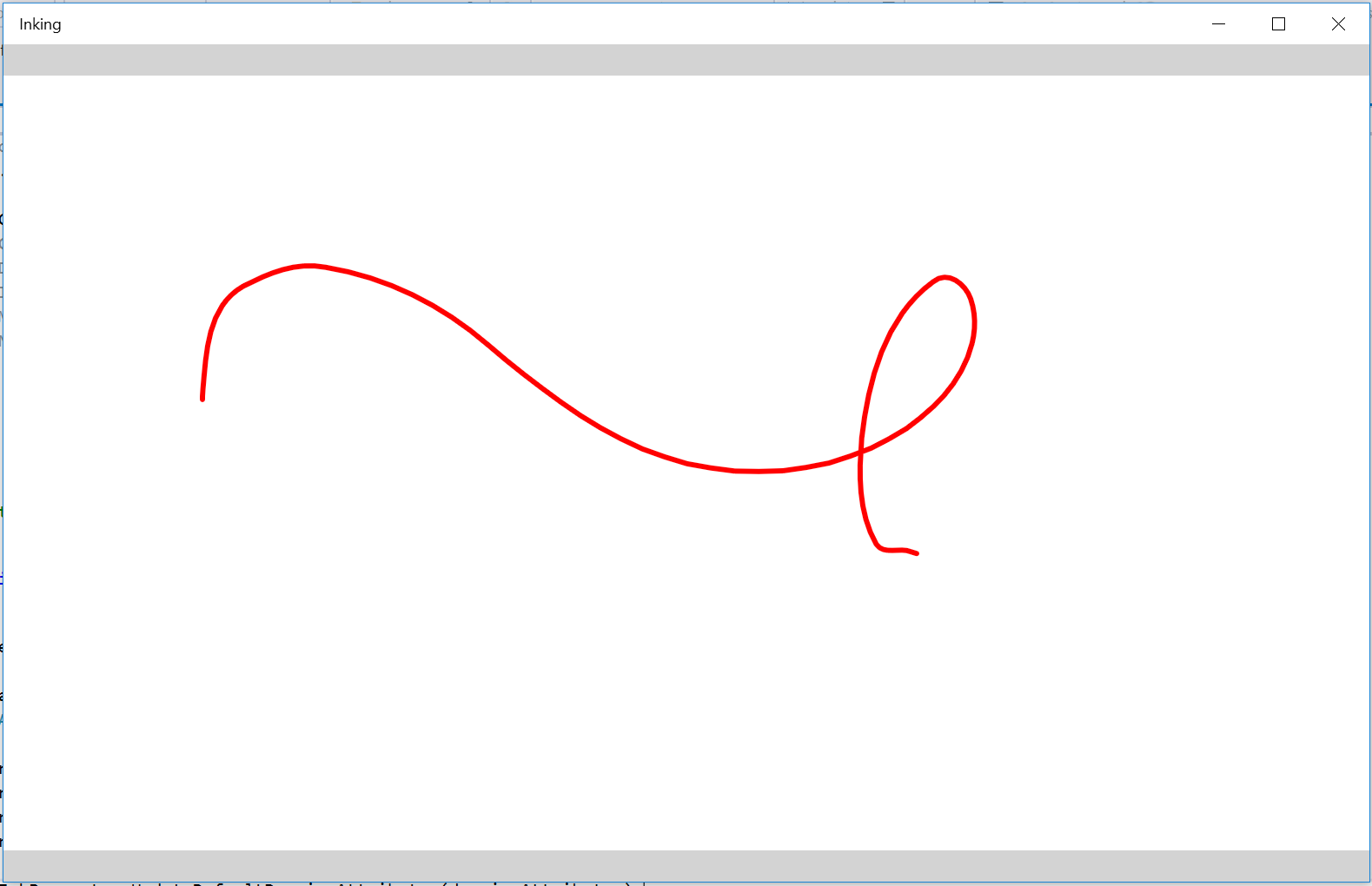


Рисунок 5

Более широкий рукописный штрих красного цвета.

1. Отключите отладку и вернитесь в Visual Studio.
2. Добавьте набор кнопок выбора цвета 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.
   * 1. XAML
   1. <StackPanel Grid.Column="0" Orientation="Horizontal">
   2. <TextBlock Text="Color" Margin="0,0,12,0" VerticalAlignment="Center" />
   3. <Button Background="Crimson" Foreground="Crimson" Click="OnPenColorChanged" Style="{StaticResource ColorButtonStyle}" Margin="0,0,3,0"/>
   4. <Button Background="Orange" Foreground="Orange" Click="OnPenColorChanged" Style="{StaticResource ColorButtonStyle}" Margin="0,0,3,0"/>
   5. <Button Background="Gold" Foreground="Gold" Click="OnPenColorChanged" Style="{StaticResource ColorButtonStyle}" Margin="0,0,3,0"/>
   6. <Button Background="LimeGreen" Foreground="LimeGreen" Click="OnPenColorChanged" Style="{StaticResource ColorButtonStyle}" Margin="0,0,3,0"/>
   7. <Button Background="DeepSkyBlue" Foreground="DeepSkyBlue" Click="OnPenColorChanged" Style="{StaticResource ColorButtonStyle}" Margin="0,0,3,0"/>
   8. <Button Background="MediumOrchid" Foreground="MediumOrchid" Click="OnPenColorChanged" Style="{StaticResource ColorButtonStyle}" Margin="0,0,3,0"/>
   9. <Button Background="Black" Foreground="Black" Click="OnPenColorChanged" Style="{StaticResource ColorButtonStyle}" Margin="0,0,3,0"/>
   10. </StackPanel>
3. Добавьте стили в элемент **Grid.Resources** для установки высоты, ширины, полей, размера шрифта и заполнения для кнопок.
   * 1. XAML
   1. <Grid Background="LightGray">
   2. <Grid.Resources>
   3. <Style x:Key="ColorButtonStyle" TargetType="Button">
   4. <Setter Property="MinWidth" Value="28"/>
   5. <Setter Property="MinHeight" Value="28"/>
   6. <Setter Property="Width" Value="28"/>
   7. <Setter Property="Height" Value="28"/>
   8. <Setter Property="Margin" Value="0,0,0,0"/>
   9. <Setter Property="FontSize" Value="0"/>
   10. <Setter Property="Padding" Value="0,0,0,0"/>
   11. </Style>
   12. </Grid.Resources>
4. В выделенном коде добавьте обработчик событий **OnPenColorChanged**. Этот метод копирует графические атрибуты по умолчанию из InkCanvas и устанавливает цвет чернил до фонового цвета кнопки.
   * 1. C#
   1. private void OnPenColorChanged(object sender, RoutedEventArgs e)
   2. {
   3. if (InkCanvas != null)
   4. {
   5. InkDrawingAttributes drawingAttributes = InkCanvas.InkPresenter.CopyDefaultDrawingAttributes();
   6. // Используйте фон кнопки, чтобы установить цвет нового пера
   7. var btnSender = sender as Button;
   8. var brush = btnSender.Background as SolidColorBrush;
   9. drawingAttributes.Color = brush.Color;
   10. InkCanvas.InkPresenter  
        .UpdateDefaultDrawingAttributes(drawingAttributes);
   11. }
   12. }
5. Скомпилируйте и запустите своё приложение. Используйте кнопки выбора цвета для изменения цвета чернил во время рисования.

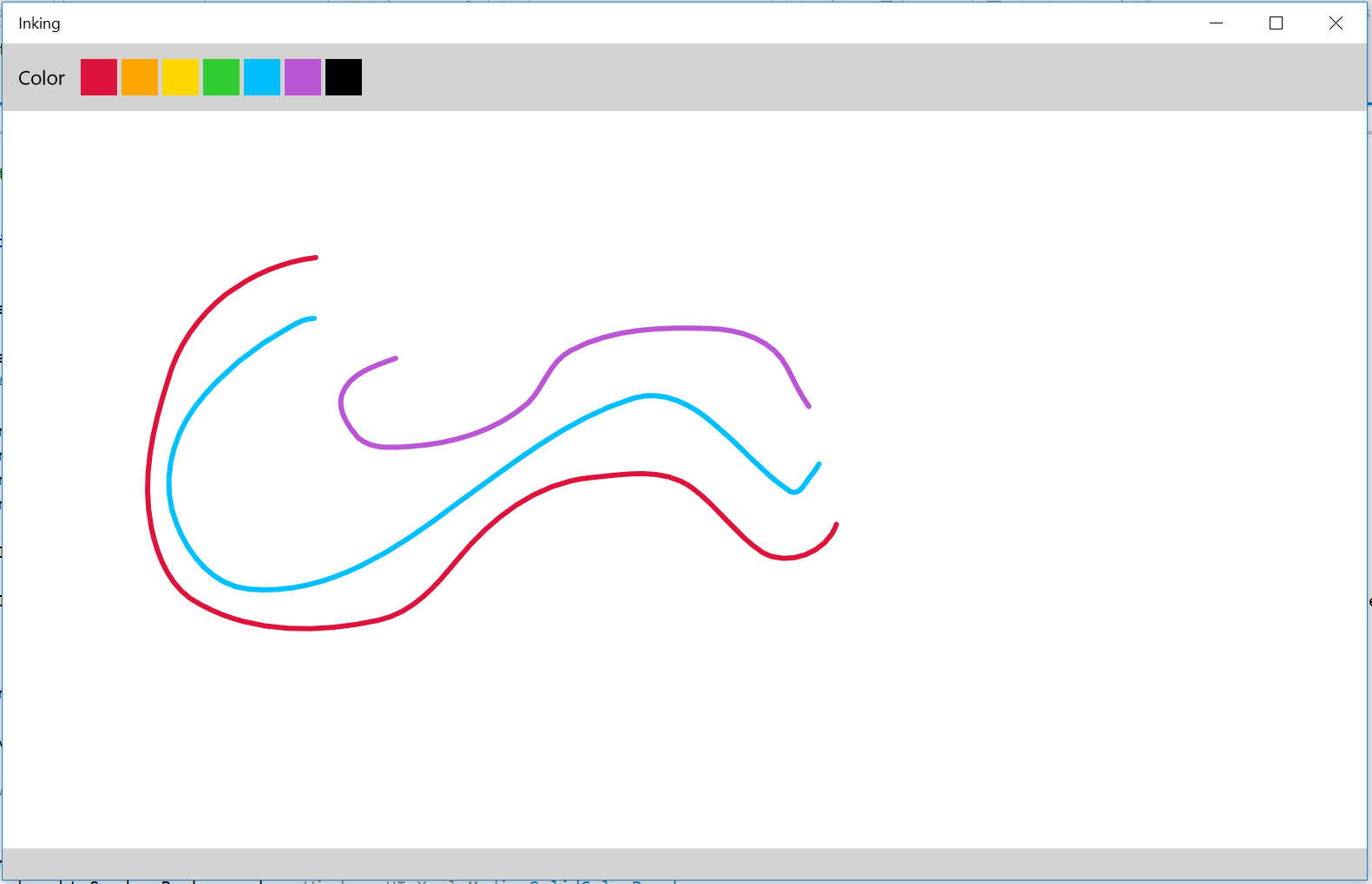


Рисунок 6

Замените цвет чернил на выделенные пользователями цвета.

1. Отключите отладку и вернитесь в Visual Studio.
   1. **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 можете настраивать штрихи, которые вы наносите, вы также можете пожелать стереть нарисованное. 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 **Отмеченное событие** на ErasingModeCheckBox\_Checked и Неотмеченное событие на ErasingModeCheckBox\_Unchecked. На следующем этапе вы создадите обработчик.
   * 1. XAML
   1. <StackPanel Grid.Column="1" Orientation="Horizontal" HorizontalAlignment="Right">
   2. <CheckBox Content="Enable Erasing Mode" Margin="20,0,4,0"   
       Checked="ErasingModeCheckBox\_Checked"   
       Unchecked="ErasingModeCheckBox\_Unchecked"/>
   3. </StackPanel>
2. Создайте обработчики для отмеченных и неотмеченных событий в выделенном коде. В отмеченном событии **InputProcessingConfiguration** настраивается на режим **Erasing (Стирания)**.
   * 1. C#
   1. private void ErasingModeCheckBox\_Checked(object sender, RoutedEventArgs e)
   2. {
   3. InkCanvas.InkPresenter.InputProcessingConfiguration.Mode =   
      InkInputProcessingMode.Erasing
   4. }
   5. private void ErasingModeCheckBox\_Unchecked(object sender, RoutedEventArgs e)
   6. {
   7. InkCanvas.InkPresenter.InputProcessingConfiguration.Mode =   
      InkInputProcessingMode.Inking;
   8. }
   9. **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, исчез ли он.
4. Отключите 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.
   * 1. XAML
   1. <StackPanel Grid.Row="2" Orientation="Horizontal" Margin="12">
   2. <Button Content="Clear" Click="OnClear" />
   3. </StackPanel>
6. Добавьте обработчик **OnClear()** в выделенный код **MainPage**.
   * 1. C#
   1. void OnClear(object sender, RoutedEventArgs e)
   2. {
   3. InkCanvas.InkPresenter.StrokeContainer.Clear();
   4. }
7. Скомпилируйте и запустите своё приложение. Нарисуйте несколько штрихов и используйте **кнопку** Очистить, чтобы убрать их из InkCanvas в одно действие.

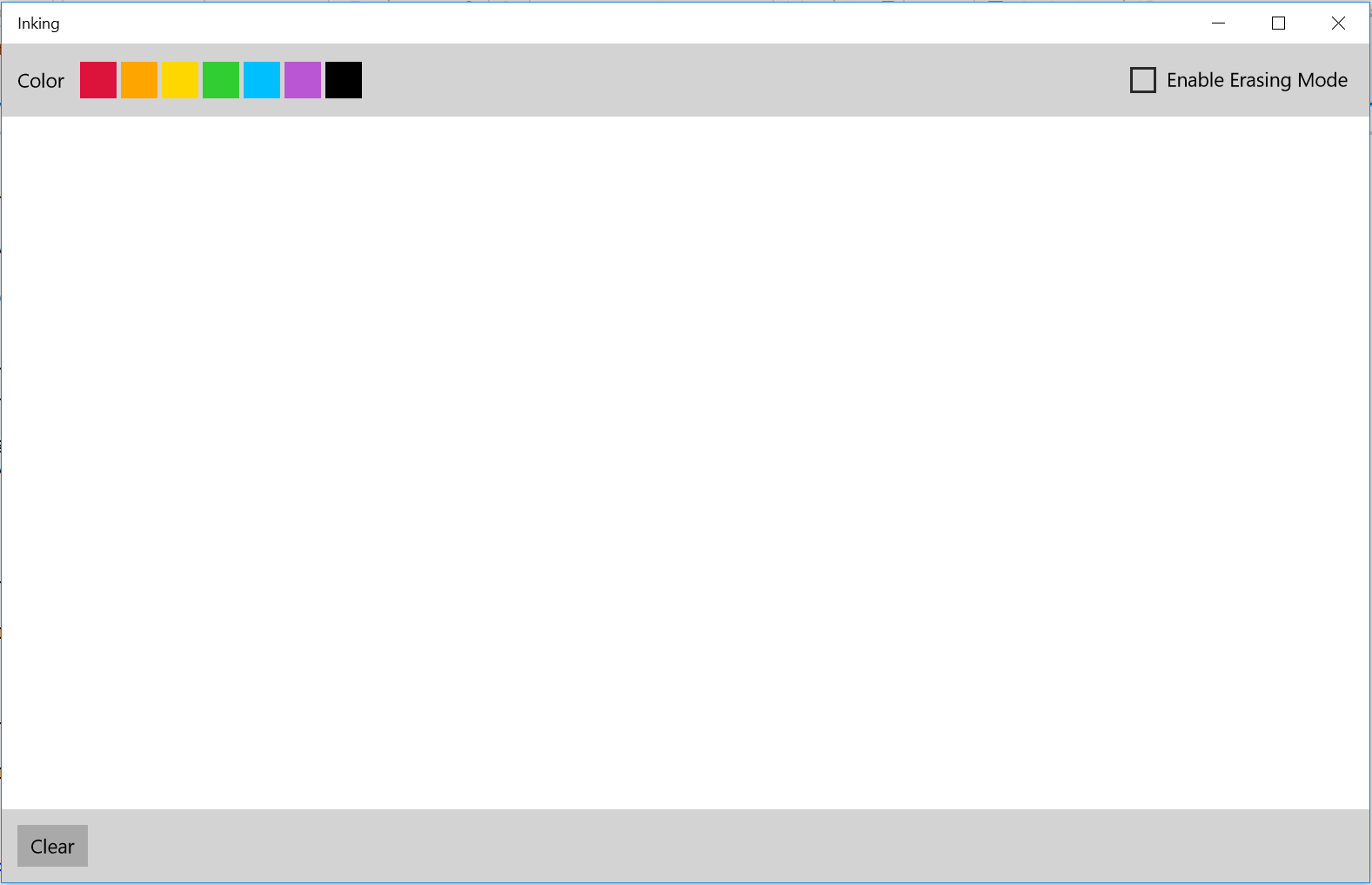


Рисунок 7

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

1. Отключите отладку и вернитесь в Visual Studio.

Задача 5 – Сохранить и загрузить

Режим Обведения чернилами полезен тем, что позволяет преобразовывать физические действия 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 перезагрузите их на полотно.

1. Добавьте кнопки **Сохранить** и **Загрузить** возле **кнопки** Очистить в MainPage.xaml. Настройте события щелчка на **OnSaveAsync** и **OnLoadAsync**, соответственно.
   * 1. XAML
   1. <StackPanel Grid.Row="2" Orientation="Horizontal" Margin="12">
   2. <Button Content="Clear" Click="OnClear" />
   3. <Button Content="Save" Margin="12,0,0,0" Click="OnSaveAsync"/>
   4. <Button Content="Load" Margin="12,0,0,0" Click="OnLoadAsync"/>
   5. </StackPanel>
2. Во вспомогательном коде MainPage добавьте пространства имен **Windows.Storage** и **Windows.Storage.Streams**.
   * 1. C#
   1. using Windows.Storage;
   2. using Windows.Storage.Streams;
3. Добавьте методы **OnSaveAsync** и **OnLoadAsync** в выделенный код. Метод **OnSaveAsync** сохраняет штрихи в виде GIF-файла. Метод загрузки считывает эти штрихи обратно на InkCanvas.
   * 1. C#
     2. async void OnSaveAsync(object sender, RoutedEventArgs e)
     3. {
     4. // Мы не хотим сохранять пустой файл
     5. if (InkCanvas.InkPresenter.StrokeContainer.GetStrokes().Count > 0)
     6. {
     7. var savePicker = new Windows.Storage.Pickers.FileSavePicker();
     8. savePicker.SuggestedStartLocation =   
        Windows.Storage.Pickers.PickerLocationId.PicturesLibrary;
     9. savePicker.FileTypeChoices.Add(  
         "Gif with embedded ISF",   
         new System.Collections.Generic.List<string> { ".gif" });
     10. Windows.Storage.StorageFile file =   
          await savePicker.PickSaveFileAsync();
     11. if (null != file)
     12. {
     13. using (IRandomAccessStream stream =   
          await file.OpenAsync(FileAccessMode.ReadWrite))
     14. {
     15. await InkCanvas.InkPresenter.StrokeContainer.SaveAsync(  
          stream);
     16. }
     17. }
     18. }
     19. }
     20. async void OnLoadAsync(object sender, RoutedEventArgs e)
     21. {
     22. var openPicker = new Windows.Storage.Pickers.FileOpenPicker();
     23. openPicker.SuggestedStartLocation =   
          Windows.Storage.Pickers.PickerLocationId.PicturesLibrary;
     24. openPicker.FileTypeFilter.Add(".gif");
     25. openPicker.FileTypeFilter.Add(".isf");
     26. Windows.Storage.StorageFile file = await openPicker.PickSingleFileAsync();
     27. if (null != file)
     28. {
     29. using (var stream = await file.OpenSequentialReadAsync())
     30. {
     31. await InkCanvas.InkPresenter.StrokeContainer.LoadAsync(stream);
     32. }
     33. }
     34. }
4. Скомпилируйте и запустите своё 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.

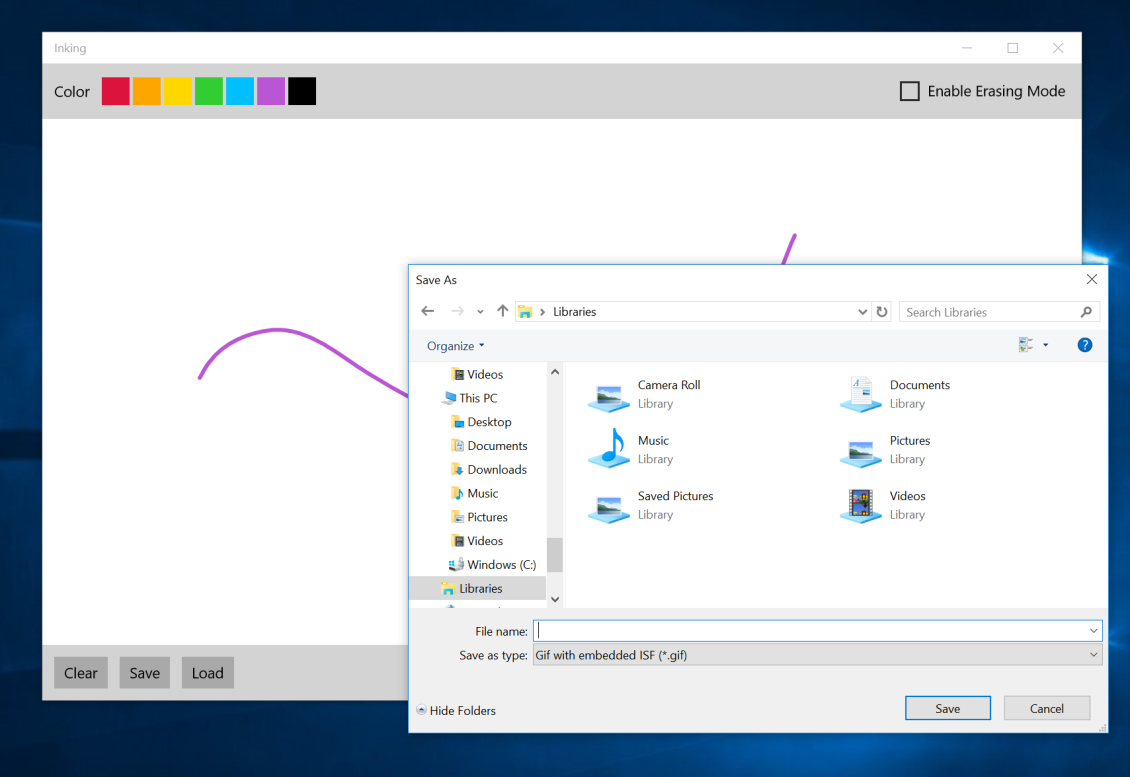


Рисунок 8

Сохраните сделанные вами штрихи на GIF-файле.

1. Отключите отладку и вернитесь в Visual Studio.

Упражнение 2: Распознавание рукописных текстов

* 1. Приложения для Рукописного стиля включают чертёжи, аннотации к изображениям и распознавание 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 для отладки.

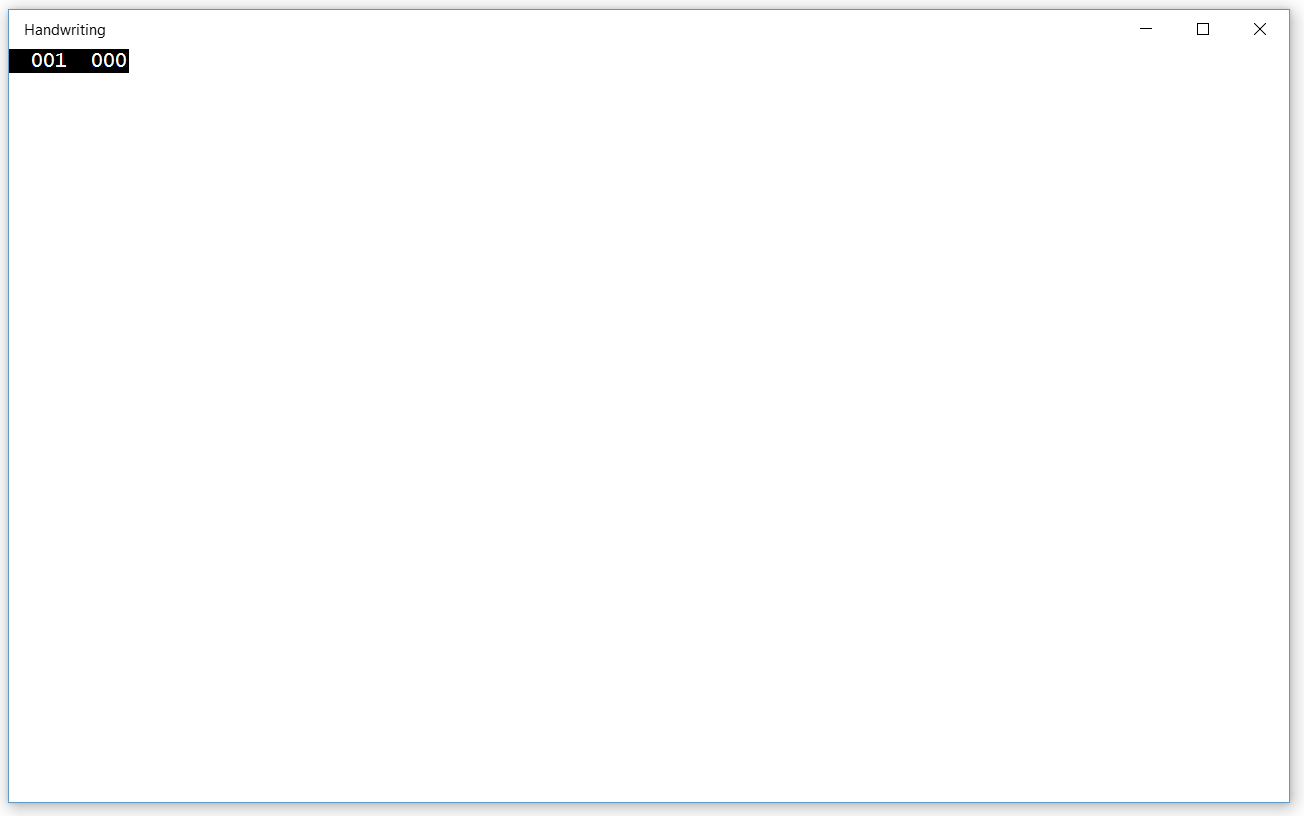


Рисунок 9

Пустое универсальное приложение, запущенное в режиме настольного компьютера.

* 1. **Примечание:** Директива препроцессора для активации или отключения счётчика скорости кадра 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 **this.DebugSettings.EnableFrameRateCounter** как **ложное**.

1. Вернитесь в Visual Studio и отключите отладку.

Задача 2 – Создайте вид

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

1. Измените Фон сетки на LightGray (светло-серый). Создайте такой же макет, который вы использовали, в Упражнении 1, используя определения строк, определения столбцов и элементы управления макетами. Добавьте InkCanvas ко второй строке с помощью **x: Name InkCanvas**.
   * 1. XAML
   1. <Grid Background="LightGray">
   2. <Grid.RowDefinitions>
   3. <RowDefinition Height="Auto" />
   4. <RowDefinition Height="2\*" />
   5. <RowDefinition Height="Auto" />
   6. </Grid.RowDefinitions>
   7. <Grid Grid.Row="0" Margin="12">
   8. <Grid.ColumnDefinitions>
   9. <ColumnDefinition Width="2\*" />
   10. <ColumnDefinition Width="1\*" />
   11. </Grid.ColumnDefinitions>
   12. <StackPanel Orientation="Horizontal" Grid.Column="0">
   13. </StackPanel>
   14. <StackPanel Grid.Column="1" HorizontalAlignment="Right"   
        Orientation="Horizontal">
   15. </StackPanel>
   16. </Grid>
   17. <Grid Grid.Row="1" Background="White" Grid.ColumnSpan="2">
   18. <InkCanvas x:Name="InkCanvas" />
   19. </Grid>
   20. <StackPanel Grid.Row="2" Orientation="Horizontal" Margin="12">
   21. </StackPanel>
   22. </Grid>
2. Добавьте пространство имен **Windows.UI.Input.Inking** в выделенный код.
   * 1. C#
   1. using Windows.UI.Input.Inking;
3. Инициализируйте графические атрибуты в конструкторе MainPage.
   * 1. C#
   1. public MainPage()
   2. {
   3. this.InitializeComponent();
   4. InkDrawingAttributes drawingAttributes = new InkDrawingAttributes();
   5. drawingAttributes.Color = Windows.UI.Colors.Black;
   6. drawingAttributes.Size = new Size(4, 4);
   7. drawingAttributes.IgnorePressure = false;
   8. drawingAttributes.FitToCurve = true;
   9. InkCanvas.InkPresenter.UpdateDefaultDrawingAttributes(  
       drawingAttributes);
   10. InkCanvas.InkPresenter.InputDeviceTypes =   
        Windows.UI.Core.CoreInputDeviceTypes.Mouse |   
        Windows.UI.Core.CoreInputDeviceTypes.Pen |   
        Windows.UI.Core.CoreInputDeviceTypes.Touch;
   11. }
4. Скомпилируйте и запустите своё приложение. Вы увидите простые возможности режима обведения чернилами, аналогичные тем, которые были в предыдущем упражнении.

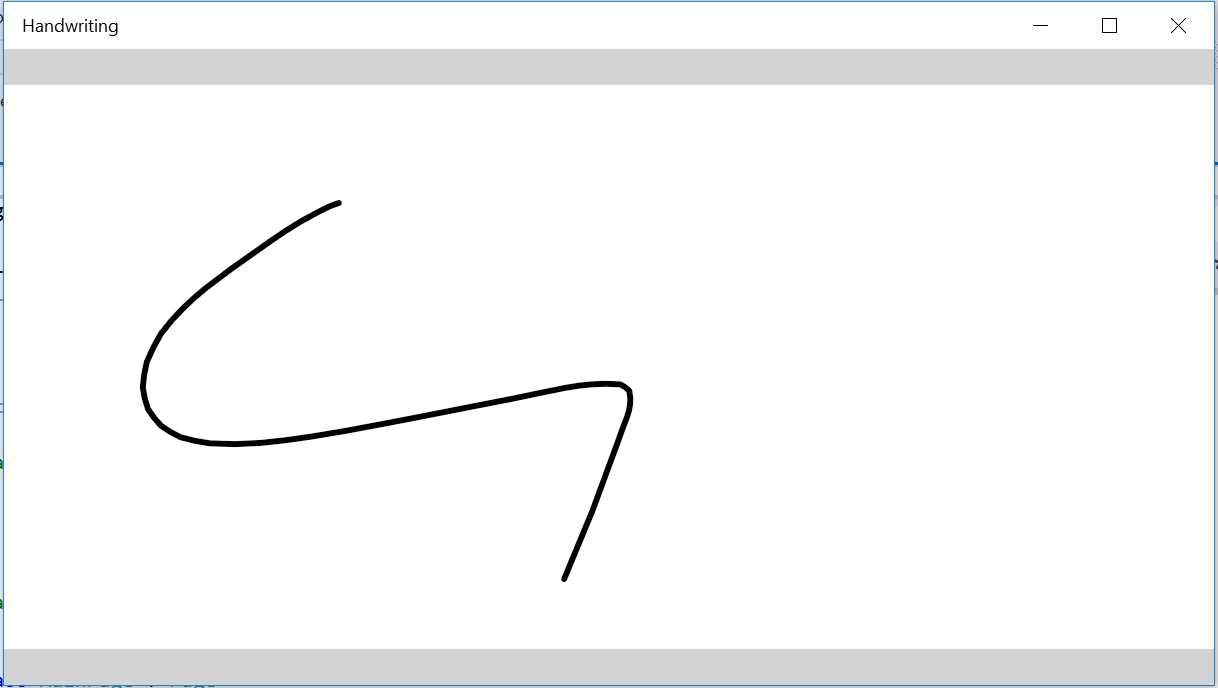


Рисунок 10

Активация приложения Распознавания рукописных текстов.

1. Отключите отладку и вернитесь в Visual Studio.

Задача 3 – Настройте распознаватель чернил

* 1. 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)**. Загрузите в курс  **папку Lab Assets** и добавьте **RecognizerHelper.cs**.
2. Откройте **RecognizerHelper.cs**. Этот вспомогательный элемент содержит 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.
   * 1. XAML
   1. <StackPanel Orientation="Horizontal" Grid.Column="0">
   2. <TextBlock Text="Available Recognizers:" Margin="0,8"/>

<ComboBox  
 x:Name="RecoName"  
 MaxWidth="500"  
 SelectionChanged="OnRecognizerChanged">  
 </ComboBox>

* 1. </StackPanel>

1. Создайте **текстовый блок** состояния в третьей строке. Текст состояния отобразит результаты распознавания рукописных текстов или сообщение, если в системе не было обнаружено никаких распознавателей.
   * 1. XAML
   1. <StackPanel Grid.Row="2" Orientation="Horizontal" Margin="12">
   2. <TextBlock x:Name="Status" Margin="20,0,0,0" />
   3. </StackPanel>
2. Добавьте пространство имен**Windows.Globalization** **в выделенный код** MainPage.
   * 1. C#
   1. using Windows.Globalization;
3. Создайте экземпляры **InkRecognizers** и список**InkRecognizerContainer** доступный только для чтения в выделенном коде.
   * 1. C#
   1. Рукописный текст пространства имён
   2. {
   3. /// <Резюме>
   4. /// Пустая страница, которая может использоваться самостоятельно или к которой можно перейти в пределах кадра.
   5. /// </Резюме>
   6. public sealed partial class MainPage : Page
   7. {
   8. InkRecognizerContainer inkRecognizerContainer = null;
   9. private IReadOnlyList<InkRecognizer> recoView = null;
4. Загрузите доступные распознаватели системы в конструктор MainPage.
   * 1. C#
   1. public MainPage()
   2. {
   3. this.InitializeComponent();
   4. InkDrawingAttributes drawingAttributes = new
   5. InkDrawingAttributes();
   6. drawingAttributes.Color = Windows.UI.Colors.Black;
   7. drawingAttributes.Size = new Size(4, 4);
   8. drawingAttributes.IgnorePressure = false;
   9. drawingAttributes.FitToCurve = true;
   10. inkRecognizerContainer = новый InkRecognizerContainer();
   11. recoView = inkRecognizerContainer.GetRecognizers();
   12. if (recoView.Count > 0)
   13. {
   14. foreach (InkRecognizer recognizer in recoView)
   15. {
   16. RecoName.Items.Add(recognizer.Name);
   17. }
   18. }
   19. else
   20. {
   21. RecoName.IsEnabled = false;
   22. RecoName.Items.Add("No Recognizer Available");
   23. }
   24. RecoName.SelectedIndex = 0;
   25. InkCanvas.InkPresenter.UpdateDefaultDrawingAttributes(  
        drawingAttributes);
   26. InkCanvas.InkPresenter.InputDeviceTypes =   
        Windows.UI.Core.CoreInputDeviceTypes.Mouse |   
        Windows.UI.Core.CoreInputDeviceTypes.Pen |   
        Windows.UI.Core.CoreInputDeviceTypes.Touch;
   27. }
5. Создайте обработчик **OnRecognizerChanged** для настройки значения ComboBox на выбранный распознаватель и метод **SetRecognizerByName** для отслеживания обнаруженных в системе распознавателей.
   * 1. C#
   1. void OnRecognizerChanged(object sender, RoutedEventArgs e)
   2. {
   3. string selectedValue = (string)RecoName.SelectedValue;
   4. SetRecognizerByName(selectedValue);
   5. }
6. bool SetRecognizerByName(string recognizerName)
7. {
8. bool recognizerFound = false;
9. foreach (InkRecognizer reco in recoView)
10. {
11. if (recognizerName == reco.Name)
12. {
13. inkRecognizerContainer.SetDefaultRecognizer(reco);
14. recognizerFound = true;
15. break;
16. }
17. }
18. if (!recognizerFound)
19. {
20. Status.Text = "Could not find target recognizer.";
21. }
22. return recognizerFound;
23. }
24. Скомпилируйте и запустите своё приложение. Используйте ComboBox для просмотра списка распознавателей, установленных в вашей системе.

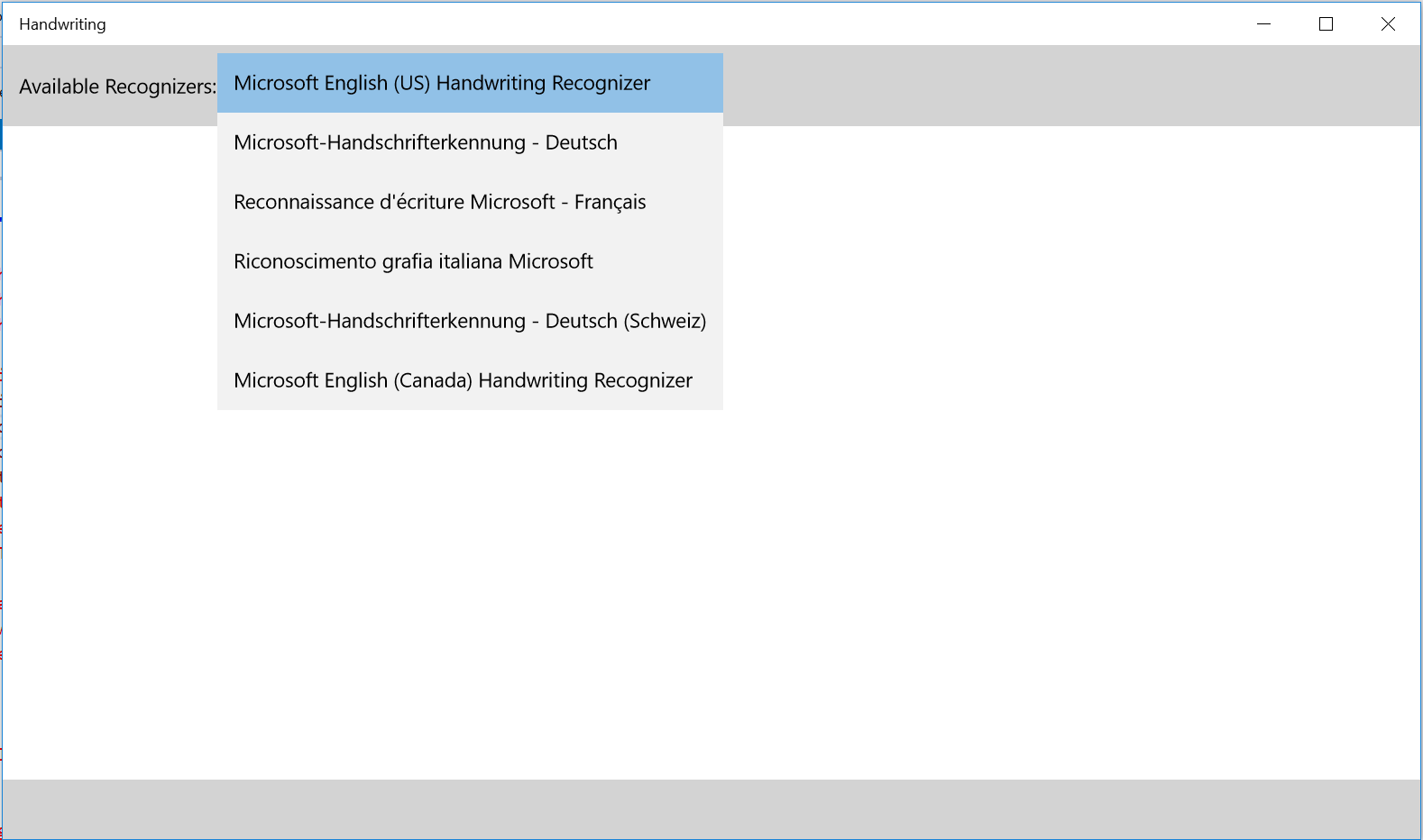


Рисунок 11

ComboBox предлагает распознаватели доступные на устройстве.

1. Отключите отладку и вернитесь в Visual Studio.

Задача 4 – Примените OnRecognizeAsync()

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

1. Добавляет кнопки Распознать и Очистить ко второму столбцу первой строки в просмотре MainPage.
   * 1. XAML
   1. <StackPanel Grid.Column="1" HorizontalAlignment="Right" Orientation="Horizontal">
   2. <Button x:Name="RecognizeBtn" Content="Recognize" Width="100"   
       Margin="0,0,4,0" Click="OnRecognizeAsync"/>
   3. <Button x:Name="ClearBtn" Content="Clear" Width="65" Margin="0,0,4,0"   
       Click="OnClear"/>
   4. </StackPanel>
2. Добавьте обработчик **OnClear()** в выделенный код MainPage. Этот метод идентичен методу **OnClear**(), который мы использовали в Упражнении 1.
   * 1. C#
   1. void OnClear(object sender, RoutedEventArgs e)
   2. {
   3. InkCanvas.InkPresenter.StrokeContainer.Clear();
   4. }
3. Добавьте обработчик **OnRecognizeAsync()** в выделенный код. Этот метод ожидает результатов по штрихам, нанесенным в InkCanvas, и в выбранном распознавателе. Если будут результаты, они будут отображены в пакете текстов **Состояния**. Если результаты будут неокончательными, **Состояние** выдаст сообщение **Нет никаких распознанных текстов**.
   * 1. C#
   1. async void OnRecognizeAsync(object sender, RoutedEventArgs e)
   2. {
   3. IReadOnlyList<InkStroke> currentStrokes =   
      InkCanvas.InkPresenter.StrokeContainer.GetStrokes();
   4. if (currentStrokes.Count > 0)
   5. {
   6. RecognizeBtn.IsEnabled = false;
   7. ClearBtn.IsEnabled = false;
   8. RecoName.IsEnabled = false;
   9. var recognitionResults = await inkRecognizerContainer.RecognizeAsync(  
       InkCanvas.InkPresenter.StrokeContainer,   
       InkRecognitionTarget.All);
   10. if (recognitionResults.Count > 0)
   11. {
   12. // Показать результат распознавания
   13. string str = "Recognition result:";
   14. foreach (var r in recognitionResults)
   15. {
   16. str += " " + r.GetTextCandidates()[0];
   17. }
   18. Status.Text=str;
   19. }
   20. else
   21. {
   22. Status.Text = "No text recognized.";
   23. }
   24. RecognizeBtn.IsEnabled = true;
   25. ClearBtn.IsEnabled = true;
   26. RecoName.IsEnabled = true;
   27. }
   28. else
   29. {
   30. Status.Text="Must first write something.";
   31. }

}

1. Скомпилируйте и запустите своё приложение. Выберите распознаватель из списка и запишите слово от того языка на полотне. Используйте кнопку **распознавания** для просмотра результатов

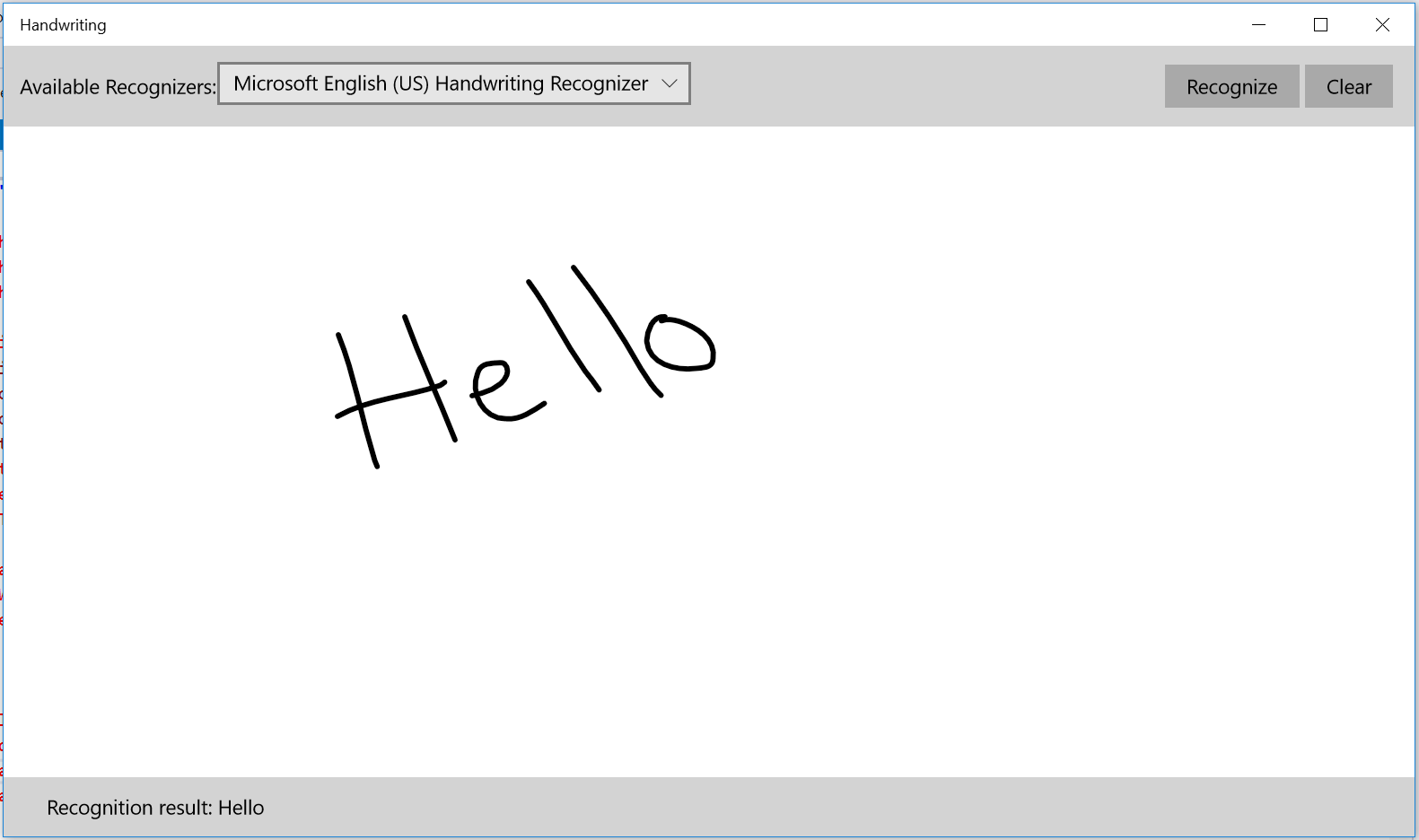


Рисунок 12

Успешное распознавание рукописного текста.

1. Отключите отладку и вернитесь в Visual Studio.

Краткий обзор

* 1. Чернила – естественный и удобный способ взаимодействия с приложениями на устройствах Windows 10, которые имеют опции ввода данных с помощью мышки, прикосновения, пера или контроллера. Рисование чернилами быстро и легко выполнять в приложениях UWP. В этом курсе 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.