# 电子墨水（Digital Ink） 绘图板应用

## 概述

在Windows 10平台上，用户可以通过触控笔或手指这种更加自然的输入方式与应用互动，实现单人绘图、多人绘图同步、手写笔迹识别等丰富的场景。Windows 10为开发者提供了实现这些场景所需要的基础API，可以让用户在应用内设置画笔粗细、笔触颜色、类型，进行笔触编辑，并且可以通过官方提供的远程通信接口，轻松构建远程分布式应用，实现多人绘图、实时同步的功能。

## 通过示例您将学会

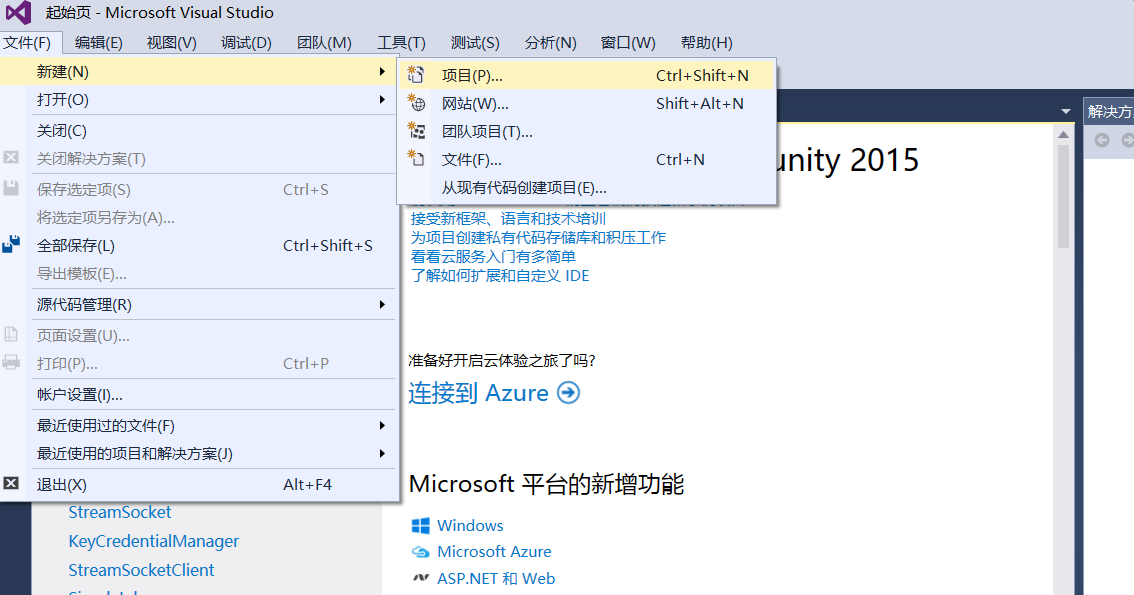
* 标准画板：画笔类型、画笔粗细、画笔颜色、加载图片及开启/关闭触控模式的设置。
* 手写识别：文本识别器的构建和手写文字识别。
* 墨水编辑：绘图内容的剪切、复制、粘贴操作。
* 设备间同步：多设备协同绘图。
* 其他：应用程序多语言支持。

## 挑战

我们将使用Visual Studio 2015 创建一个新的项目，来学习绘图板相关API的使用方法。

— 从桌面或开始菜单运行Visual Studio 2015。

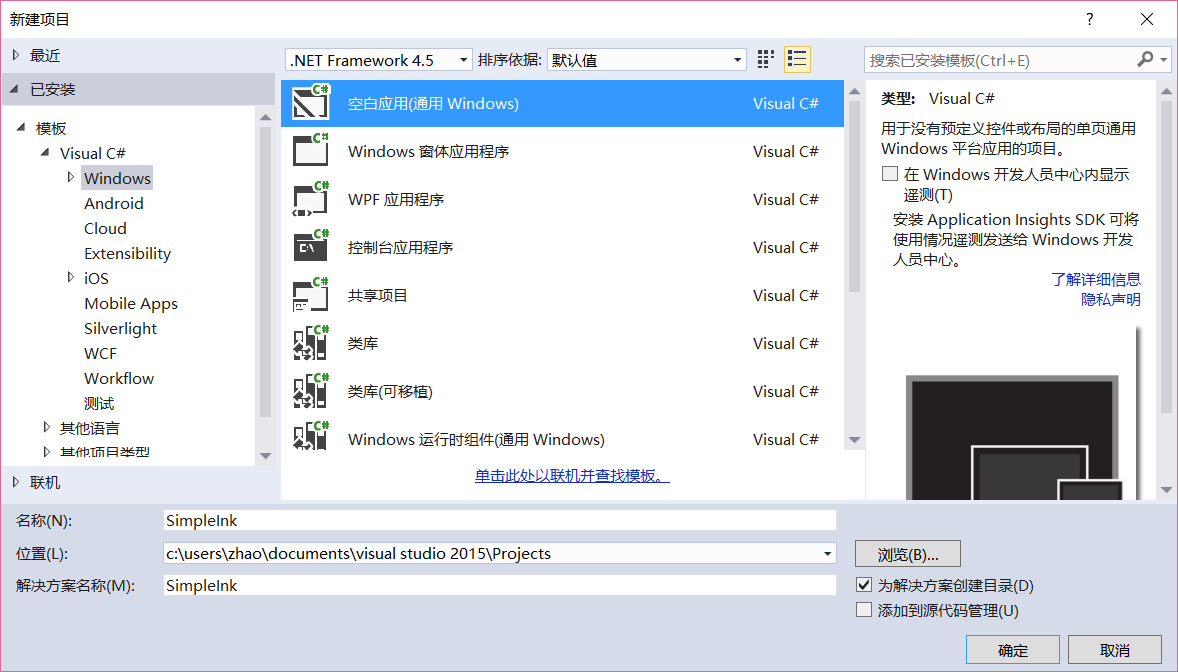
## 创建新工程



从所提供的项目模板中选择空白应用（通用Windows）类型项目。

该模板位于已安装**→模板→Visual C#→Windows→空白应用（通用Windows）**

一个空白应用（通用Windows）是一个最简单的窗口应用程序，它可以运行在多个Windows 10设备上面。我们把这个项目命名为SimpleInk。



## 创建第一个场景

### 场景描述

在这个场景里面，主要演示了如何创建绘图板工具。用户可以通过触控笔或鼠标左键在画板上进行创作。同时，用户可以设置画笔类型、粗细、颜色，实现加载本地图片、开启/关闭触控操作等功能。

### 知识点

**InkCanvas Control**：简单来说InkCanvas是实现墨迹的布局控件，它主要通过鼠标或触控笔来捕捉笔迹。是实现该场景的基础控件。

**InkDrawingAttributes Class**：画笔属性，包括画笔粗细、颜色及文本识别等属性；

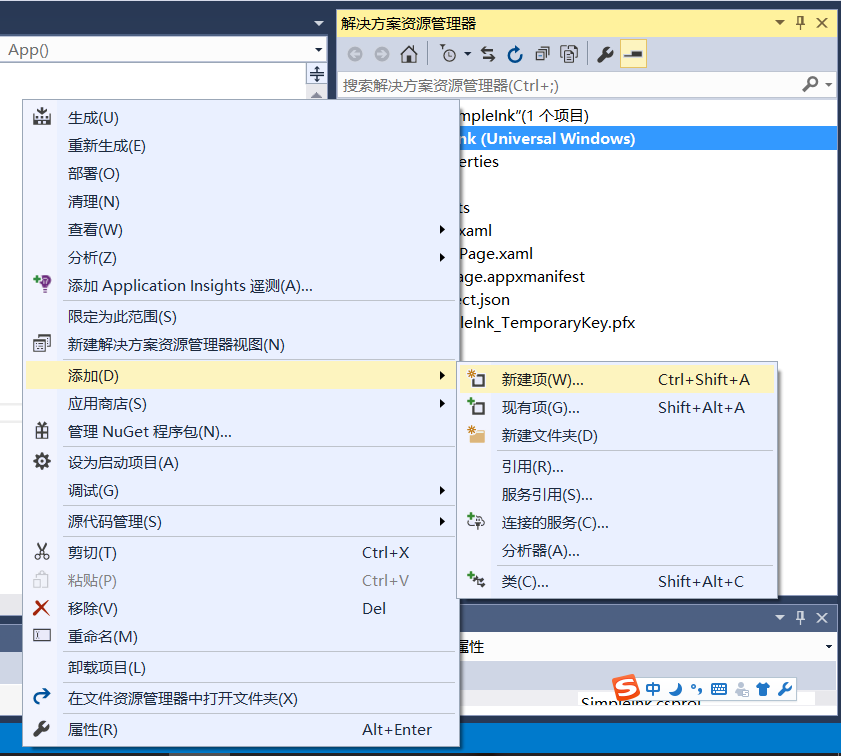
**FileSavePicker Class**：文件选择器，主要用于将文件保存在用户机器中。

**FileOpenPicker Class**;文件选择器，主要用于选择一个或多个用户机器中的文件。

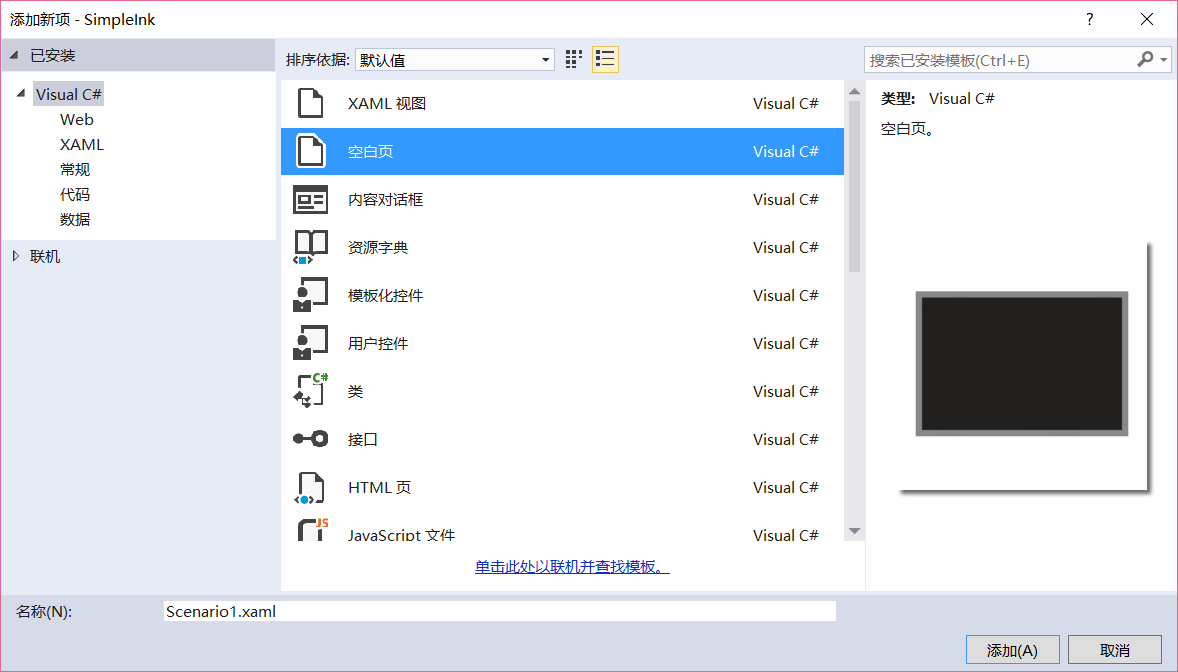
**StorageFile Class**:表示一个文件。提供有关文件及其内容的信息，以及处理它们的方法。

### 创建XAML视图文件

在解决方案资源管理器中，右键单击该项目SimpleInk（除非你有其他的命名）。从“添加”菜单中，选择“新建项”



在左侧Visual C# 模板中，可以选择文件类型，我们添加一个“空白页”用来展示我们的第一个应用场景，命名为Scenario1。



打开Scenario1的xaml文件，添加InkCanvas控件，代码样章如下：

<Grid x:Name="outputGrid" Grid.Row="1" HorizontalAlignment="Stretch"

Background="{ThemeResource SystemControlBackgroundChromeWhiteBrush}"

BorderBrush="{StaticResource SplitViewPaneBackgroundColor}" BorderThickness="2">

<!-- Inking area -->

<InkCanvas x:Name="inkCanvas"/>

</Grid>

打开Scenario1.xaml.cs文件，在页面构造函数中添加如下代码：

InkDrawingAttributes drawingAttributes = new InkDrawingAttributes();

drawingAttributes.Color = Windows.UI.Colors.Red;

drawingAttributes.Size = new Size(penSize, penSize);

drawingAttributes.IgnorePressure = false;

drawingAttributes.FitToCurve = true;

//set drawingAttributes

inkCanvas.InkPresenter.UpdateDefaultDrawingAttributes(drawingAttributes);

inkCanvas.InkPresenter.InputDeviceTypes =Windows.UI.Core.CoreInputDeviceTypes.Mouse | Windows.UI.Core.CoreInputDeviceTypes.Pen |Windows.UI.Core.CoreInputDeviceTypes.Touch;

绘图板的渲染，主要是通过InkCanvas控件的InkPresenter属性来实现的。

1. **InkPresenter**：它是Windows.UI.Input.Inking命名空间下的一个类，主要用于显示墨迹笔画的一个画布，InkCanvas的墨迹捕捉实际上也是由它来完成。

**InkDrawingAttributes**：墨迹绘制属性，InkPresenter主要根据InkDrawingAttributes中的设置来采集、捕捉笔迹。InkDrawingAttributes主要属性如下：

* **InkDrawingAttributes.Color**：画笔颜色。
* **InkDrawingAttributes.Size**：画笔粗细。
* **InkDrawingAttributes.DrawAsHighlighter**：是否采用荧光笔绘制。
* **InkDrawingAttributes.FitToCurve**：是否使用贝塞尔曲线
* **InkDrawingAttributes.IgnorePressure**：是否忽略压力
* **InkDrawingAttributes.PenTip**：笔尖形状
* **InkDrawingAttributes.Matrix3x2**：3\*2转转矩阵

## 创建第二个场景

### 场景描述

在这个场景里面，主要演示了手写笔迹的识别。

### 知识点

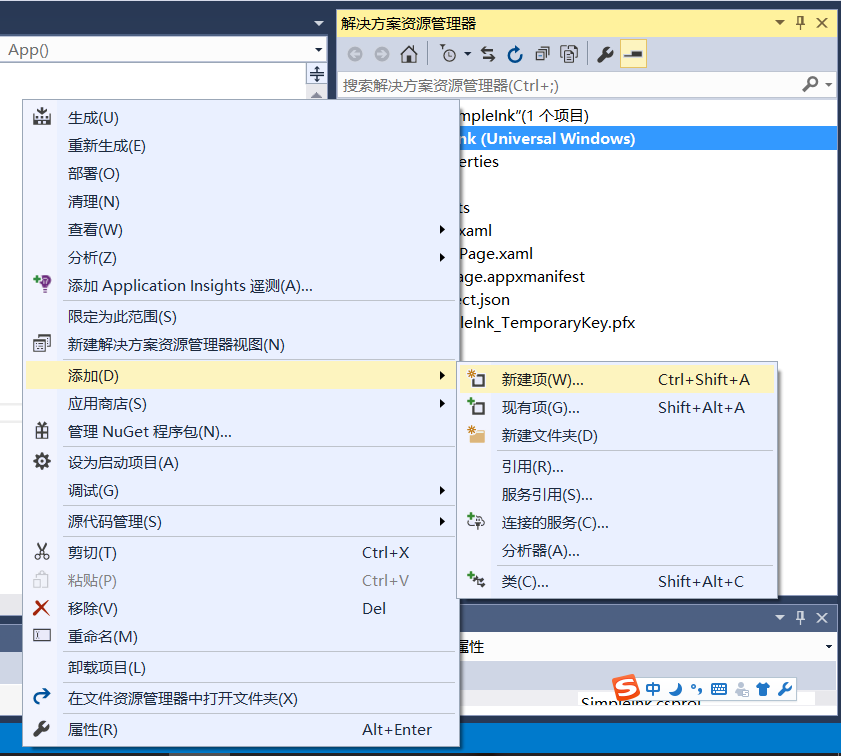
**InkCanvas Control**：详见第一个场景里的描述。

**InkDrawingAttributes**：详见第一个场景里的描述

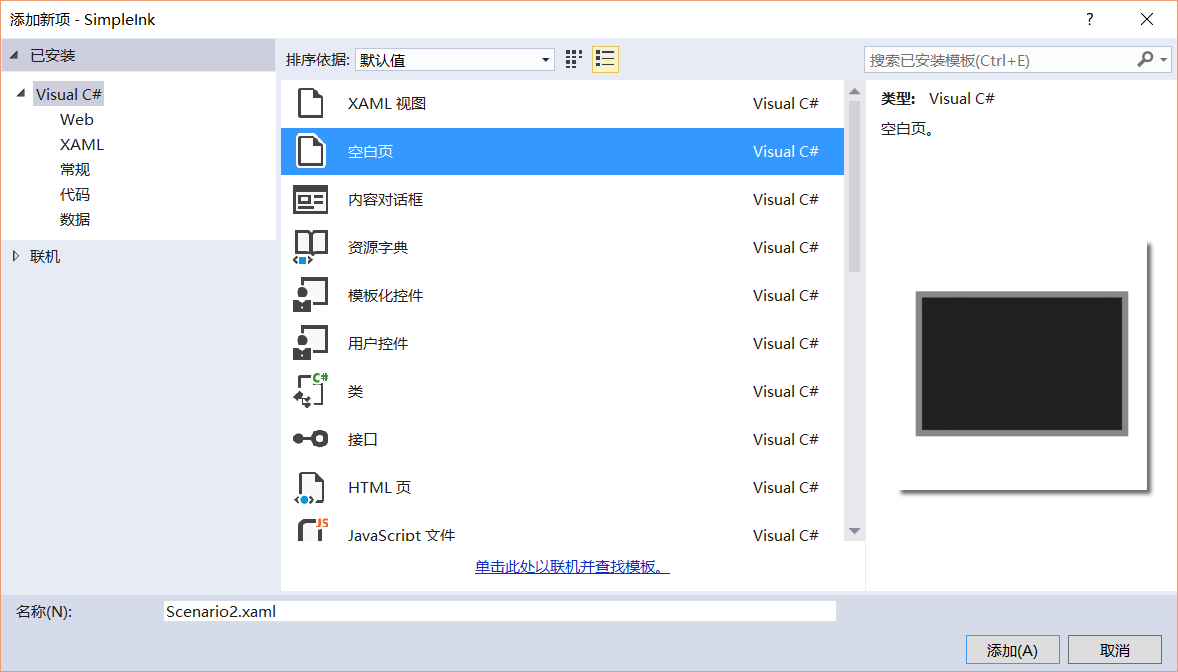
**InkRecognizerContainer**：该类是Windows提供给开发者的一个识别引擎，它可以通过一组画笔墨迹来识别出文本。

### 创建XAML视图文件

在解决方案资源管理器中，右键单击该项目SimpleInk（除非你有其他的命名）。从“添加”菜单中，选择“新建项”



在左侧Visual C# 模板中，可以选择文件类型，我们添加一个“空白页”用来展示我们的第二个应用场景。命名为Scenario2。



打开Scenario2的xaml文件，添加InkCanvas控件，代码样章如下：

<Grid x:Name="Output" Grid.Row="3" HorizontalAlignment="Stretch"

Background="{ThemeResource SystemControlBackgroundChromeWhiteBrush}"

BorderBrush="{StaticResource SplitViewPaneBackgroundColor}" BorderThickness="2">

<!-- Inking area -->

<InkCanvas x:Name="inkCanvas"/>

</Grid>

文本识别代码如下：

//Recognize Text

IReadOnlyList<InkStroke> currentStrokes = inkCanvas.InkPresenter.StrokeContainer.GetStrokes();

if (currentStrokes.Count > 0)

{

inkRecognizerContainer.RecognizeAsync(inkCanvas.InkPresenter.StrokeContainer, InkRecognitionTarget.All);

if (recognitionResults.Count > 0)

{

// Display recognition result

string str = "";

foreach (var r in recognitionResults)

{

str += " " + r.GetTextCandidates()[0];

}

this.textShow.Text = str;

}

else

{

rootPage.ShowMessage("No text recognized.");

}

}

InkRecognizerContainer主要的函数描述如下：

1. **GetRecognizers()**：获取当前应用设置的识别器。（识别器需要Bcp47支持，如"zh-CN", "Microsoft 中文(简体)手写识别器"）；
2. **SetDefaultRecognizer(InkRecognizer recognizer)**：设置默认的识别器，设置成功后，识别引擎会根据识别器的语言，来识别文本。
3. **RecognizeAsync(InkStrokeContainer** strokeCollection**, InkRecognitionTarget** recognitionTarget**)**：文本识别，调用示例如下：

Var recognitionResults = await inkRecognizerContainer.RecognizeAsync(inkCanvas.InkPresenter.StrokeContainer, InkRecognitionTarget.All);

该函数返回一组对象来标识识别结果。集合中的每个项表示一个书面语。

## 创建第三个场景

### 场景描述

在这个场景里，演示了画板图画的编辑。用户可以通过长按右鼠标右键，来选取墨迹，并且可以对墨迹进行剪切、复制、粘贴等操作（对于Phone用户，我们提供选取墨迹的功能按钮）。

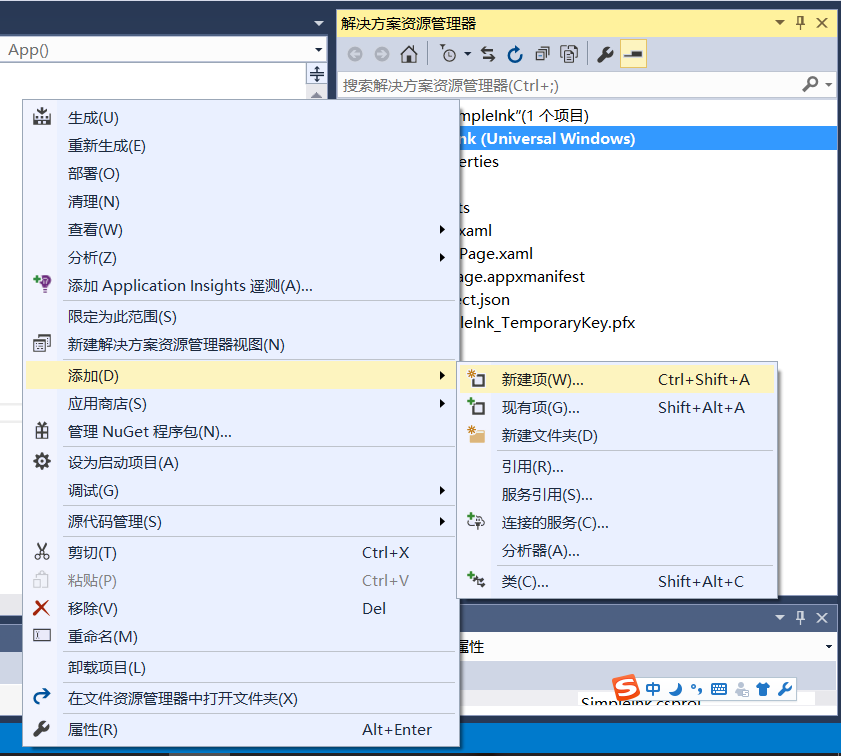
### 知识点

**InkCanvas**：详见第一个场景里的描述。

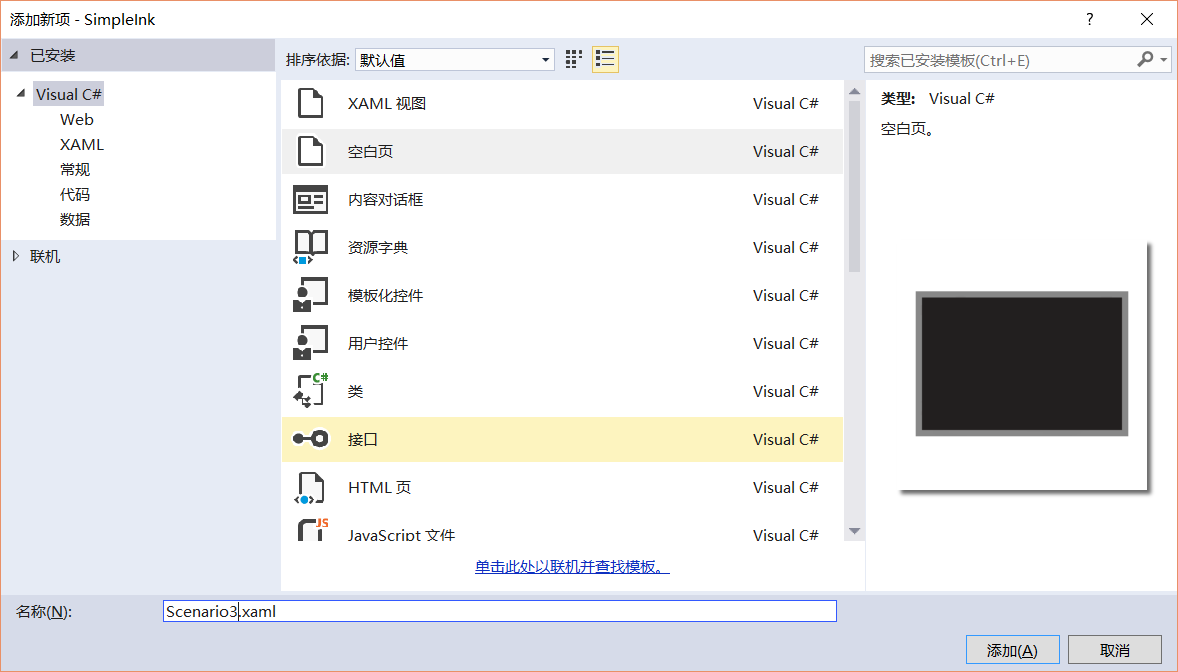
**Polyline**：PolyLine类主要用于绘制一系列相互连接的直线，简单来说就是一个矩形或正方形。

### 创建XAML视图文件

在解决方案资源管理器中，右键单击该项目SimpleInk（除非你有其他的命名）。从“添加”菜单中，选择“新建项”。



在左侧Visual C# 模板中，可以选择文件类型，我们添加一个“空白页”用来展示我们的第三个应用场景。命名为Scenario3。



打开Scenario3的xaml文件，添加InkCanvas控件，代码样章如下：

<Grid x:Name="outputGrid" Grid.Row="1" HorizontalAlignment="Stretch"

Background="{ThemeResource SystemControlBackgroundChromeWhiteBrush}"

BorderBrush="{StaticResource SplitViewPaneBackgroundColor}" BorderThickness="2">

<!-- Inking area -->

<Canvas x:Name="selectionCanvas"/>

<InkCanvas x:Name="inkCanvas" Height="Auto"/>

</Grid>

**注意：**在这个场景里面，比第一个和第二个场景中多出一个Canvas。它是一个透明的浮层，主要用于显示选取图片的框。<Canvas x:Name="selectionCanvas"/>

在这个场景中，主要使用InkCanvas.InkPresenter中的StrokeContainer属性，它是一个InkStrokeContainer类型，主要使用到的函数如下：

1. **Clear()**：从InkStrokeContainer中删除所有的信息。即清除墨迹。
2. **DeleteSelected()**：取消InkStrokeContainer中InkStroke内容的选取。
3. **SelectWithPolyLine(IEnumerable<Point> polyline)**:选中InkStrokerContainer中InkStroke内容。即选中所有的墨迹。
4. **CopySelectedToClipboard()**：将选中InkStroke对象，复制到剪贴板。
5. **PasteFromClipboard(Point position)**：将剪切板中的InkStroke对象，粘贴到InkStrokeContainer对象，以便呈现该画笔。
6. **AddStroke(InkStroke stroke)**：添加墨迹到绘图板中。
7. **MoveSelected(Point translation)**：根据坐标参数，移动选中的Stroke。

## 创建第四个场景

### 场景描述

在这个场景里，主要演示了多设备间的同步绘图场景。在填写远程设备IP地址后，可以与远程设备建立Socket连接, 实时共享绘图板。

### 知识点

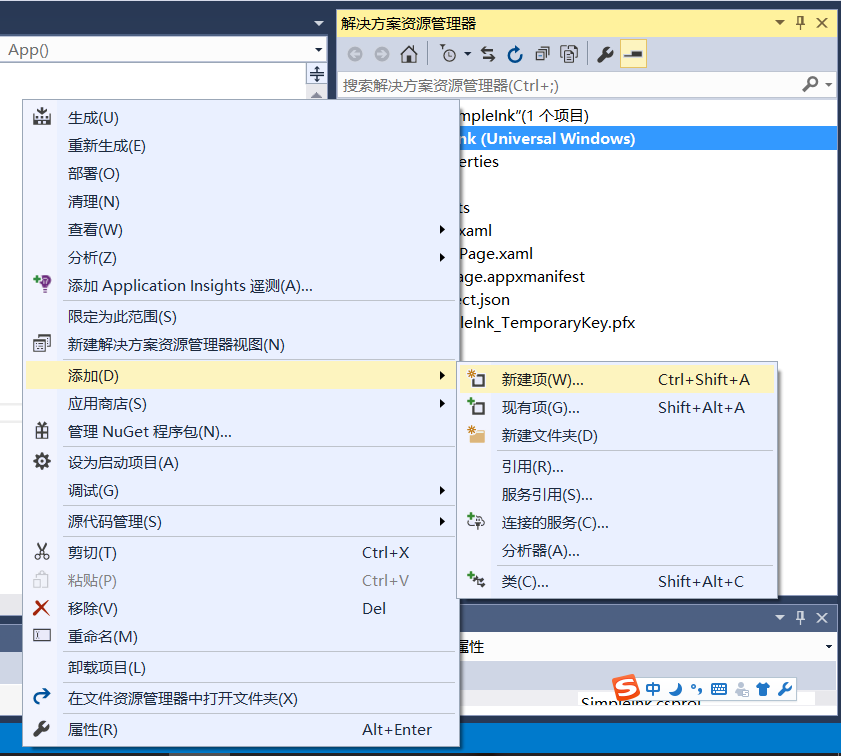
**StreamSocketListener**:通过 StreamSocketListener 实现 tcp 通信的服务端的 socket 监听。

**StreamSocket**：通过 StreamSocket 实现 tcp 通信的客户端 socket

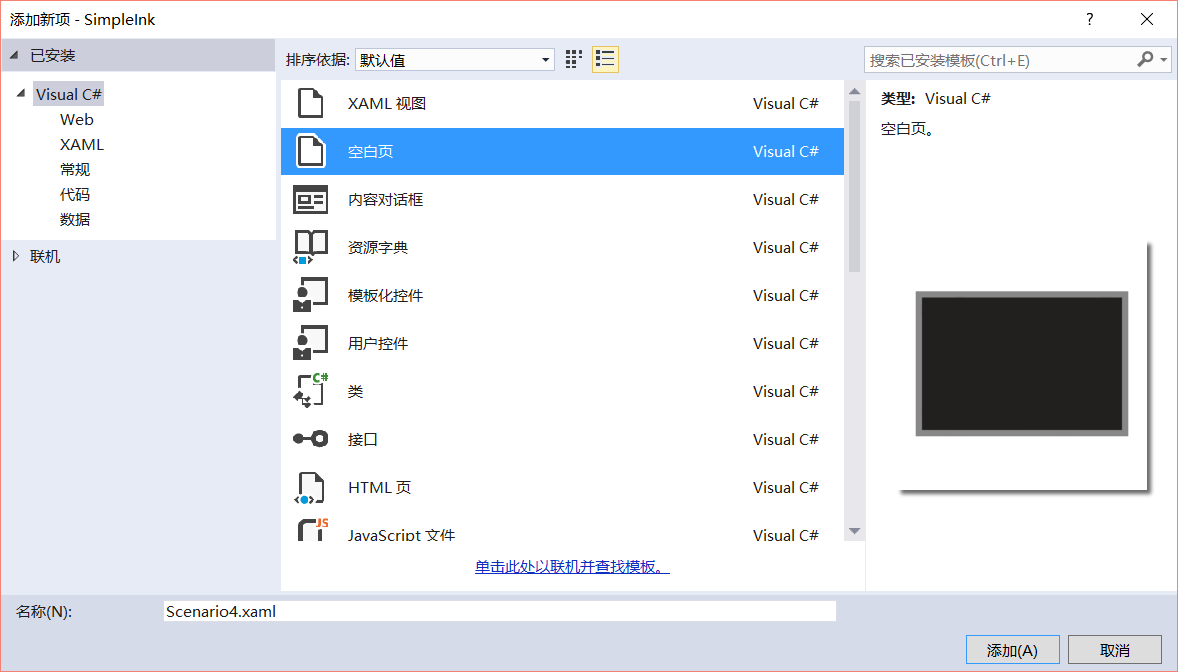
**InkStrokeContainer：**LoadAsync(IInputStream inputStream)、SaveAsync(IOutputStream outputStream)

### 创建XAML视图文件

在解决方案资源管理器中，右键单击该项目SimpleInk（除非你有其他的命名）。从“添加”菜单中，选择“新建项”。



在左侧Visual C# 模板中，可以选择文件类型，我们添加一个“空白页”用来展示我们的第四个应用场景。命名为Scenario4。



Socket监听代码样章：

try

{

StreamSocketListener listener = new StreamSocketListener();

listener.ConnectionReceived += OnConnection;

listener.Control.KeepAlive = false;

await listener.BindEndpointAsync(new HostName(txtBoxAddress.Text), ServiceName);

rootPage.ShowMessage("Listening");}

catch (Exception exception)

{

if (SocketError.GetStatus(exception.HResult) == SocketErrorStatus.Unknown)

{

rootPage.ShowMessage(

"Start listening failed with error: " + exception.Message); }

}

Socket ConnectionReceived事件代码：

private async void OnConnection(StreamSocketListener sender,StreamSocketListenerConnectionReceivedEventArgs args)

{

DataReader reader = new DataReader(args.Socket.InputStream);

try

{

while (true)

{

uint sizeFieldCount = await reader.LoadAsync(sizeof(uint));

if (sizeFieldCount != sizeof(uint))

{

return;

}

uint stringLength = reader.ReadUInt32();

uint actualStringLength = await reader.LoadAsync(stringLength);

if (stringLength != actualStringLength)

{

return;

}

byte[] dataArray = new byte[actualStringLength];

reader.ReadBytes(dataArray);

await this.Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.Normal, async () =>

{

inkCanvas.InkPresenter.StrokeContainer.Clear();

InMemoryRandomAccessStream randomAccessStream = new InMemoryRandomAccessStream();

await randomAccessStream.WriteAsync(dataArray.AsBuffer());

randomAccessStream.Seek(0);

await inkCanvas.InkPresenter.StrokeContainer.LoadAsync(randomAccessStream);

}).AsTask();

}

}

catch (Exception exception)

{

if (SocketError.GetStatus(exception.HResult) == SocketErrorStatus.Unknown)

{

rootPage.ShowMessage(

"Start Receive failed with error: " + exception.Message);

}

}

}

Socket连接代码：

private async void btnConnect\_Tapped(object sender, TappedRoutedEventArgs e)

{

HostName hostName;

try

{

hostName = new HostName(this.txtBoxAddress.Text);

}

catch (ArgumentException)

{

rootPage.ShowMessage("Error: Invalid host name.");

return;

}

this.socketSend = new StreamSocket();

this.socketSend.Control.KeepAlive = false;

try

{

await this.socketSend.ConnectAsync(hostName, ServiceName);

isConnect = true;

}

catch (Exception exception)

{

if (SocketError.GetStatus(exception.HResult) == SocketErrorStatus.Unknown)

{

rootPage.ShowMessage("Connect failed with error: " + exception.Message);

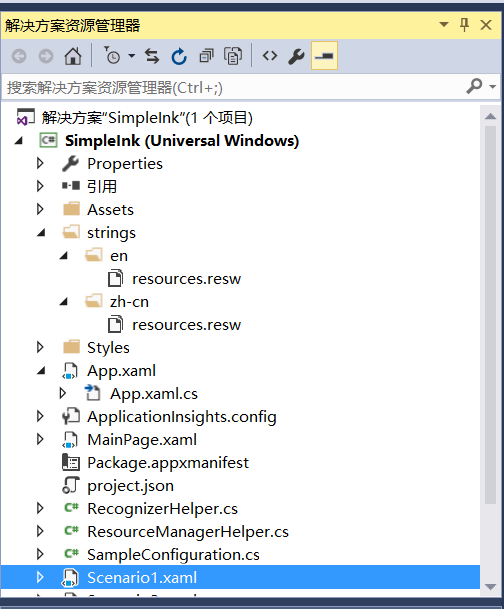
}

}

}

1. 在这个场景里面，远端设备绘图板的显示是通过InkCanvas.InkPresenter属性的StrokeContainer类，LoadAsync(IInputStream inputStream)函数和SaveAsync(IOutputStream outputStream)函数来完成的。
2. LoadAsync 函数：从内存流中加载InkStroke对象到InkStrokeContainer,以便于画笔显示。
3. SaveAsync 函数：将InkStrokeContainer中的InkStroke对象写入内存流。

## 多语言支持



在解决方案中，添加strings文件夹，里面分别创建英文和中文的资源文件，en/resources.resw、zh-cn/resources.resw。用于存储UI元素的英文显示和中文显示。

在Windows 10 中，当前语言设置可以使用如下代码：

Windows.Globalization.ApplicationLanguages.PrimaryLanguageOverride = "zh-cn"; or

Windows.Globalization.ApplicationLanguages.PrimaryLanguageOverride = "en-US";

**PrimaryLanguageOverride**：为当前应用程序的首选语言。