Ink, Touch, and Windows Presentation Foundation

Lecture #4: Ink and WPF
Joseph J. LaViola Jr.
Fall 2015

From Last Time

- Windows Presentation Foundation (WPF)
  - integration of
    - Ink
    - multi-touch
    - 2D Graphics
    - 3D Graphics
    - video and audio
  - uses visual tree model
    - component based
- XAML and C# code
- Important control – InkCanvas
Ink Environment Choices

- Microsoft traditional dev stream: Visual Studio, C#, WPF or (Mono+Xamarin)
  - **Pros**: Mature platform with many online resources
  - **Cons**: Can be deployed only on Windows ecosystem
  - *Using Mono+Xamarin can be deployed cross-platform*

- Microsoft new dev stream: Visual Studio, JavaScript(or Typescript),HTML
  - **Pros**: Can be deployed cross-platform(windows, android or IOS)
  - **Cons**: Relatively new computing environment with new API

- Other alternatives, such as Python/Kivy, IOS, Android.
  - **Pros**: Open-source with many online collaboration.
  - **Cons**: Have little support on inking

Traditional Ink and Touch Ink SDK

Before Windows 8.1:
- The inking API is `System.Windows.Ink`
- Online Sample code: *Adventures into Ink API using WPF*

After Windows 8.1:
- The inking API is `Windows.UI.Input.Inking`
- Online Sample code: *Input:Simplied ink sample*

You can find more articles or sample code resources from Microsoft *Windows Dev Center or Code project*
Important Ink Components

- **InkCanvas – System.Windows.Controls**
  - receives and displays ink strokes
  - starting point for ink applications
  - stores ink in Strokes

- **System.Windows.Ink Namespace**
  - contains classes to interact with and manipulate ink
  - examples
    - Stroke
    - GestureRecognizer
  - InkAnalyzer now separate (only on 32 bit)
    - needs IACore.dll, IAWinFX.dll and IALoader.dll

Dealing with InkCanvas

- *InkCanvas* collects Strokes
- Strokes contain *StylusPoints*
- *StylusPoints* contain X,Y, Pressure
  - can also be converted into Geometry objects
- Strokes contain
  - digitizer packets
  - drawing attributes
  - application-defined data
- *InkCanvas* has several stylus level events
  - StrokeCollected, StylusInAirMove, …
Strokes and Geometry

- Strokes
  - perform hit tests
  - get geometry, bounds, Bezier points
  - add properties
  - transformations

- Geometry
  - lose pressure and stylus specific data
  - Within scope of 2D graphics API
  - get area
  - create shapes

- No Cusp or self-intersection detection

More InkCanvas Features

- Enough support to implement Windows Journal

- Modes
  - Ink
  - InkandGesture
  - GestureOnly
  - EraseByStroke
  - EraseByPoint
  - Select
  - None
## Drawing Attributes

- Can access on stroke level using *Drawing Attributes* property
- Can access on global level using the *InkCanvas DefaultDrawingAttributes* property
- Example attributes
  - color
  - Bezier curves
  - height and width of ink stroke
  - ignoring pressure

---

### InkCanvas Example

```xml
<Window x:Class="WpfApplication4.InkTest"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="InkTest" Height="300" Width="300"
    Visibility='Visible'>
  <Grid>
    <InkCanvas
      Name='_ink'
      StrokeCollected='Collected'
      Background='Beige' />
    <Canvas Name='_overlay' />
  </Grid>
</Window>
```

```csharp
private void Collected(object sender, InkCanvasStrokeCollectedEventArgs e)
{
    _overlay.Children.Clear();
    Brush fill = new SolidColorBrush(Color.FromArgb(120, 255, 0, 0));
    foreach (StylusPoint pt in e.Stroke.StylusPoints)
    {
        double markerSize = pt.PressureFactor * 35.0;
        Ellipse marker = new Ellipse();
        Canvas.SetLeft(marker, pt.X - markerSize / 2);
        Canvas.SetTop(marker, pt.Y - markerSize / 2);
        marker.Width = marker.Height = markerSize;
        marker.Fill = fill;
        _overlay.Children.Add(marker);
    }
}
```

Creating Your Own InkCanvas

- InkCanvas handles approx. 90-95% of what you need
- Can develop custom InkCanvas
  - InkPresenter – System.Windows.Controls
  - Stylus events
- See Windows SDK documentation

Stylus Descriptions

- Other data besides x,y points and pressure
  - xtilt, ytilt
  - Barrel button
- Can request data globally using `DefaultStylusPointDescription` on `InkCanvas`
- Per stroke with `Reformat` method on `StylusPointCollection`
Stylus Description Example

```csharp
public InkTest() {
    InitializeComponent();
    _ink.DefaultStylusPointDescription = new StylusPointDescription{
        new StylusPointPropertyInfo[] {
            new StylusPointPropertyInfo(StylusPointProperties.X),
            new StylusPointPropertyInfo(StylusPointProperties.Y),
            new StylusPointPropertyInfo(StylusPointProperties.NormalPressure),
            new StylusPointPropertyInfo(StylusPointProperties.BarrelButton),
        }
    );
}
```

Asks for information on x, y, pressure, and if the barrel button is pressed.

Gesture Recognition

- Built in Gesture recognition engine
  - handwriting recognition and ink analysis are separate (outside of InkCanvas)
- 41 distinct gestures (found in ApplicationGesture enum)
  - check
  - square
  - triangle
  - arrows
  - scratchout
  - etc…
Gesture Recognition Example

```xml
<Window x:Class="WpfApplication6.Window1"
     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
     Title="GestureTester">
    <StackPanel>
        <InkCanvas Height='200' Name='_ink'
                   Gestures='InkGesture'
                   EditingMode='InkAndGesture' />
        <ListBox Name='_seen' />
    </StackPanel>
</Window>
```

```csharp
public partial class Window1 : Window {
    public Window1() {
        InitializeComponent();
        _ink.SetEnabledGestures(new ApplicationGesture[] {
            ApplicationGesture.AllGestures, });
    }

    private void InkGesture(object sender, InkCanvasGestureEventArgs e) {
        _seen.Items.Add(e.GetGestureRecognitionResults()[0].ApplicationGesture);
    }
}
```

Collecting Timing Information

```csharp```
// Create a guid for the date/timestamp.
Guid dtGuid = new Guid("03457307-3475-3450-3035-640435034540");
DateTime now = DateTime.Now;
// Check whether the property is already saved
if ((thisStroke.ContainsPropertyData(dtGuid)) { }
// Check whether the existing property matches the current date/timestamp
if (oldDT != now) { }
// Update the current date and time
thisStroke.AddPropertyData(dtGuid, now);
```

This snippet works on a Stroke by Stroke basis. Can you think of how to do this on a point by point basis?
Multi-Touch in WPF 4.5

- Basic Touch events
  - TouchEnter, TouchLeave
  - TouchMove, PreviewTouchMove
  - TouchDown, TouchUp
- Events get raised for each finger independently
- Equivalent mouse events for first finger

- Manipulation events
  - ManipulationStarting
  - ManipulationDelta
  - ManipulationCompleted

- Operations
  - Translation
  - Scale
  - Rotation
  - Expansion
- Supports inertia
Assignments

- Readings
  - WPF Unleashed -- Chapters 9-11, 15-18
  - Windows SDK documentation (Windows 8.1)
    - System.Windows.Control.InkCanvas
    - System.Windows.Input.Stylus
  - Windows SDK documentation (after Windows 8.1)
    - Windows.UI.Input.Inking

- Assignment 1 – MiniJournal posted soon