Introduction to C#, Visual Studio and Windows Presentation Foundation

Lecture #3: C#, Visual Studio, and WPF
Joseph J. LaViola Jr.
Fall 2013

C#

- Combination of C++ and Java
  - no pointer manipulation
  - built in data structures – Lists, Hash tables
  - some higher level constructs
    - foreach
  - C# not difficult
  - .NET high learning curve
  - Intellisense makes things much easier
- **Quick C# Reference**
Visual Studio 2010

- Good IDE
  - debugging
  - intellisense
- Handles WPF well
- Visual UI designer
  - Integrates with XAML

Windows Presentation Foundation (WPF)

- UI development platform from MSFT
- Integration of
  - INK!!!!
  - Multi-touch
  - 2D graphics
  - 3D graphics
  - video/audio/animation
- Declarative/Procedural programming model
  - XAML
  - C#/Visual Basic/etc…
- Uses retained mode
  - implies scenegraph

www.markmywords.org
blogs.msdn.com/mgrayson/
WPF Features and Machinery

- **Control library**
  - buttons, sliders, menus, toolbars
  - tool tips, popups, scroll bars, etc...
  - user defined as well
- **Layout panels**
  - canvas, stack, wrap, doc panels
  - grid – most flexible
- **Actions**
  - events
  - commands
  - triggers
- **Styles, skins, themes, templates**

Logical and Visual Trees in WPF

- UIs are constructed from a tree of objects (logical tree)
- Visual tree expands logical tree
  - nodes broken down into visual components
  - not all logical tree nodes appear in visual tree
**Extensible Application Markup Language (XAML)**

- Set of semantics on top of XML
- Tags always defined in context namespace
- Easy to read like and write
  - a.k.a. HTML
  - declarative
  - want to integrate graphic designers
- Independent of WPF
- Ideal for rapid UI prototyping
  - set up UI then write procedural code

**WPF Example – Button**

```xml
<Window
xmlns='http://schemas.microsoft.com/winfx/2006/xaml/presentation'
xmlns:x='http://schemas.microsoft.com/winfx/2006/xaml'
Title='Hello World'>
<Button>Howdy!</Button>
</Window>
```

WPF Example – Stack Panel

```xml
<Window
    xmlns='http://schemas.microsoft.com/winfx/2006/xaml/presentation'
    Title='Hello World' >
    <StackPanel>
        <Button>Howdy!</Button>
        <Button>A second button</Button>
    </StackPanel>
</Window>
```

WPF Example – More Controls

```xml
<Window
    xmlns='http://schemas.microsoft.com/winfx/2006/xaml/presentation'
    Title='Hello World' >
    <StackPanel>
        <Button>Howdy!</Button>
        <Button>A second button</Button>
        <TextBox>An editable text box</TextBox>
        <CheckBox>A check box</CheckBox>
        <Slider Width='75' Minimum='0' Maximum='100' Value='50' /> 
    </StackPanel>
</Window>
```
**WPF Example – Wrap Layout**

```xml
<Window
    xmlns='http://schemas.microsoft.com/winfx/2006/xaml/presentation'
    Title='Hello World'>
    <WrapPanel>
        <Button>Howdy!</Button>
        <Button>A second button</Button>
        <TextBox>An editable text box</TextBox>
        <CheckBox>A check box</CheckBox>
        <Slider Width='75' Minimum='0' Maximum='100' Value='50' />
    </WrapPanel>
</Window>
```

**WPF Example – Events**

```csharp
using System;
using System.Windows.Controls;
using System.Windows;
namespace CAP6105Test
{
    public partial class MyWindow : Window
    {
        public MyWindow()
        {
            InitializeComponent();
        }
        private void HowdyClicked(object sender, RoutedEventArgs e)
        {
            _text1.Text = "Hello from C#";
        }
    }
}
```
WPF Example – Resource Binding

```xml
<Window.Resources>
  <SolidColorBrush x:Key='bg' Color='Red' />
</Window.Resources>
<WrapPanel>
  <Button Background='{{StaticResource bg}}'
    Click="HowdyClicked"> Howdy!</Button>
  <Button Background='{{StaticResource bg}}'>A second button</Button>
<WrapPanel>
```

WPF Example – Property Binding

```xml
  <Button Background='{{StaticResource bg}}'>A second button</Button>
  <TextBox x:Name='_text1'>An editable text box</TextBox>
  <CheckBox Content='{{Binding ElementName=_text1, Path=Text}}' />
```
**WPF Actions**

- 3 principles
  - element composition
  - loose coupling
  - declarative actions
- Uses events, commands, and triggers
- Utilize routed events – traverse visual tree
- Semantic events vs. physical events
  - Click vs. MouseDown

**WPF Events**

- Declare events in XAML/implement in code
- Routed events
  - Direct – fire on single source
  - Tunneling – travel from root to target element
  - Bubbling – opposite of tunneling
- Tunneling version prefixed with *Preview*
- *Handled* property can break traversal
Event Ordering Example

```xml
<Window ...
    PreviewMouseRightButtonDown='WindowPreviewRightButtonDown'
    MouseRightButtonDown='WindowRightButtonDown' >
    <GroupBox
        PreviewMouseRightButtonDown='GroupBoxPreviewRightButtonDown'
        MouseRightButtonDown='GroupBoxRightButtonDown' >
        <StackPanel>
            <Button>One</Button>
            <Button
                PreviewMouseRightButtonDown='ButtonTwoPreviewRightButtonDown'
                MouseRightButtonDown='ButtonTwoRightButtonDown' > Two </Button>
        </StackPanel>
    </GroupBox>
</Window>
```

Ordering
1. `Window PreviewMouseRightButtonDown`
2. `GroupBox PreviewMouseRightButtonDown`
3. `Button PreviewMouseRightButtonDown`
4. `Button MouseRightButtonDown`
5. `GroupBox MouseRightButtonDown`
6. `Window MouseRightButtonDown`

Commands

- Provide single name to signify an action
  - define command
  - define command implementation
  - create trigger for command
- Uses `ICommand` interface

```csharp
public interface ICommand {
    event EventHandler CanExecuteChanged;
    bool CanExecute(object parameter);
    void Execute(object parameter);
}
```
**Command Example**

```csharp
public class Exit : ICommand {
    public bool CanExecute(object parameter) {
        return true;
    }

    public event EventHandler CanExecuteChanged;

    public void Execute(object parameter) {
        Application.Current.Shutdown();
    }
}
```

```xml
<MenuItem Header='_File'>
    <MenuItem Header='E_xit'>
        <MenuItem.Command>
            <l:Exit />
        </MenuItem.Command>
    </MenuItem>
</MenuItem>
```

```csharp
public partial class Window1 : Window {
    public static readonly ICommand ExitCommand = new Exit();
    ...
}
```

**Triggers**

- Designed for markup
- Signaled by
  - state of a display property (Trigger)
  - state of a data property (DataTrigger)
    - used only within a data template
  - an event (EventTrigger)
- Cause set of actions when signaled
- MultiTrigger and MultiDataTrigger
Event Trigger Example

```
<EventTrigger RoutedEvent='FrameworkElement.Loaded'>
  <EventTrigger.Actions>
    <BeginStoryboard>
      <Storyboard>
        <DoubleAnimation
          From='-25'
          To='25'
          Storyboard.TargetName='rotation'
          Storyboard.TargetProperty='Angle'
          AutoReverse='True'
          Duration='0:0:2.5'
          RepeatBehavior='Forever'/>
      </Storyboard>
      <BeginStoryboard.Storyboard>
        <Storyboard>
          <DoubleAnimation
            From='-25'
            To='25'
            Storyboard.TargetName='rotation'
            Storyboard.TargetProperty='Angle'
            AutoReverse='True'
            Duration='0:0:2.5'
            RepeatBehavior='Forever'/>
        </Storyboard>
      </BeginStoryboard.Storyboard>
    </BeginStoryboard>
  </EventTrigger.Actions>
</EventTrigger>
```

Readings

- WPF 4.0 Unleashed by Nathan
  - Chapters 1-8