Windows Presentation Foundation

Jim Fawcett

CSE687 – Object Oriented Design

Spring 2018

References

- *Pro C# 5 and the .Net 4.5 Platform*, Andrew Troelsen, Apress, 2012
- *Programming WPF*, 2nd edition, Sells & Griffiths, O'Reilly, 2007
- *Windows Presentation Foundation Unleashed*, Adam Nathan, SAMS, 2007
- *Essential Windows Presentation Foundation*, Chris Anderson, Addison-Wesley, 2007
- http://msdn2.microsoft.com/en-us/library/aa970268.aspx
- http://msdn2.microsoft.com/en-us/library/ms754130.aspx

WPF Blogs and Tutorials

[Josh Smith Blog](#)

[WPFpedia](#)

https://www.wpftutorial.net/

Introduction

- ## What is WPF?
  - A graphical user interface technology
    - Desktop
    - Little brother Silverlight is used for web applications
  - Uses markup and code
    - Together or separately, much like [ASP.Net](ASP.Net)
  - Easy to produce different styles
    - Web browser like navigation and placement
    - Traditional forms
    - Animated graphics

Markup

- XAML
  - eXtensible Application Markup Language
  - Tags are names of .Net 3.5 classes
  - Attributes are class properties and events

    ```
    <Grid>
      <Ellipse Fill="blue" />
      <TextBlock>
        Name: <TextBlock Text="{Binding Name}" />
      </TextBlock>
    </Grid>
    ```

Code Behind

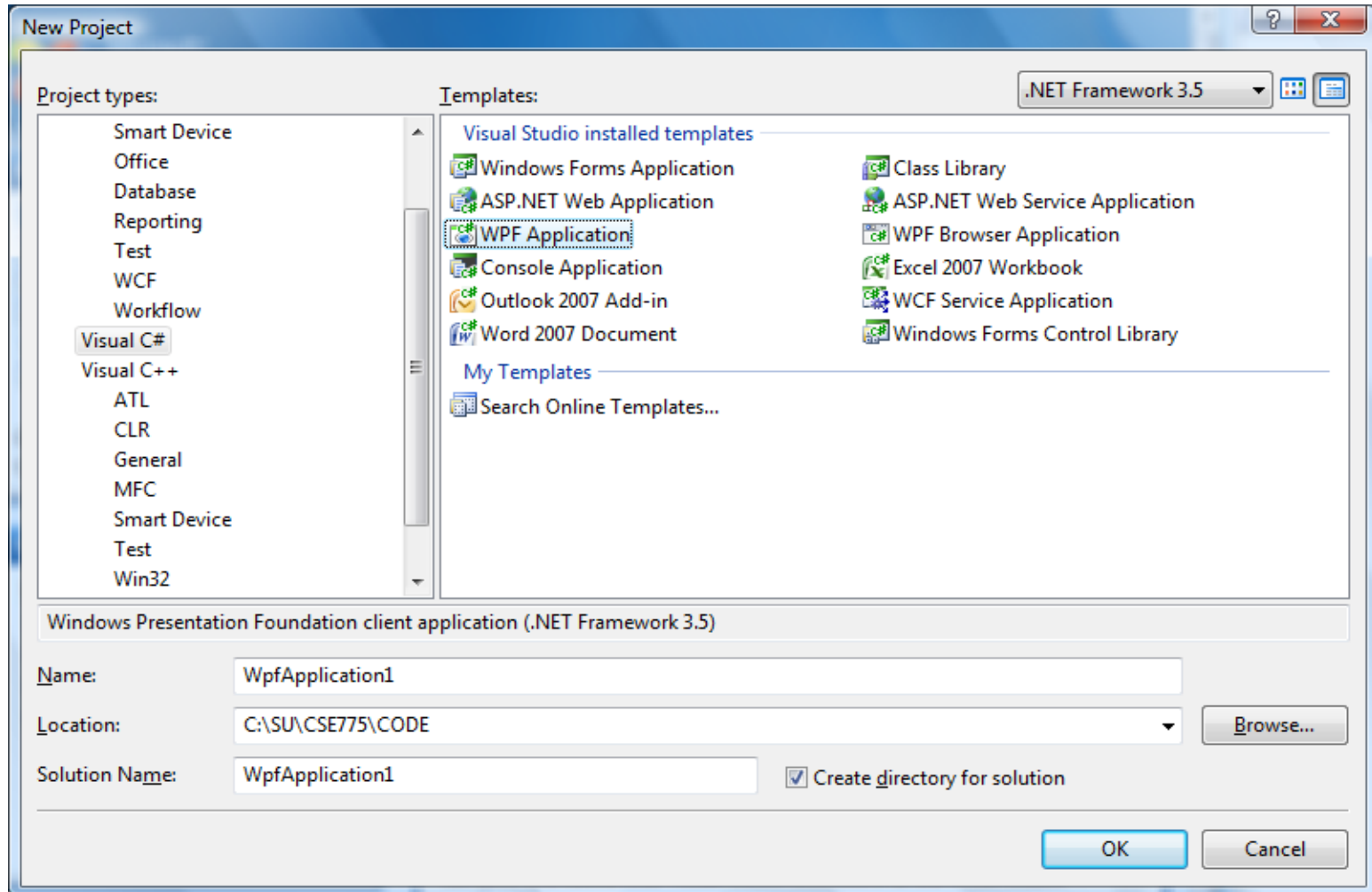- Often, code provides processing for control events, bound in XAML, like this:
  - XAML in Window.Xaml

    ```
    <Button
        x:Name="button"
        Width="200"
        Height="25"
        Click="button_Click">Submit</Button>
    ```

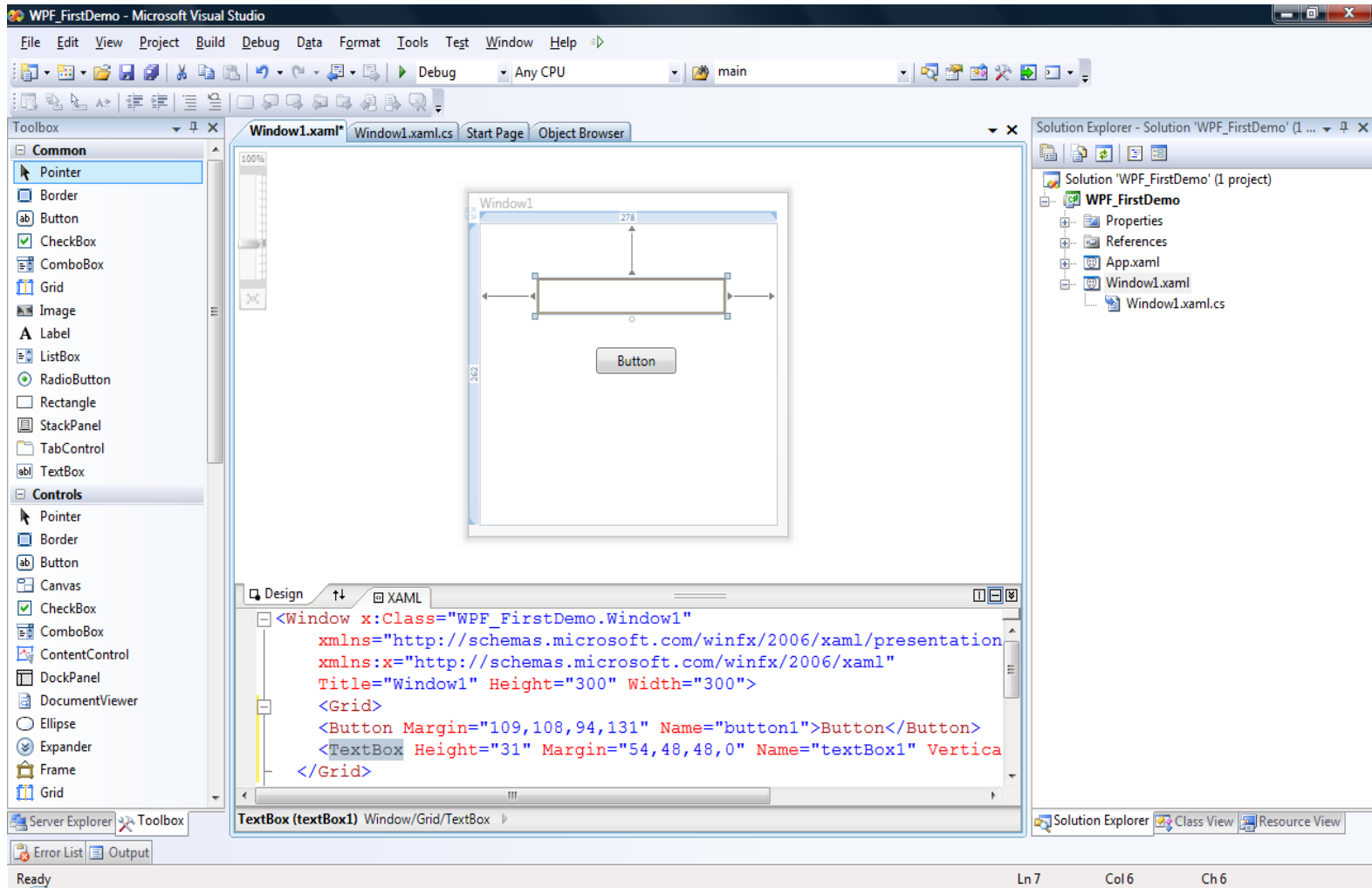 - C# code in Window.Xaml.cs

    ```
    Void button_Click(object sender, RoutedEventsArgs e) {
        MessageBox.Show(...) }
    ```
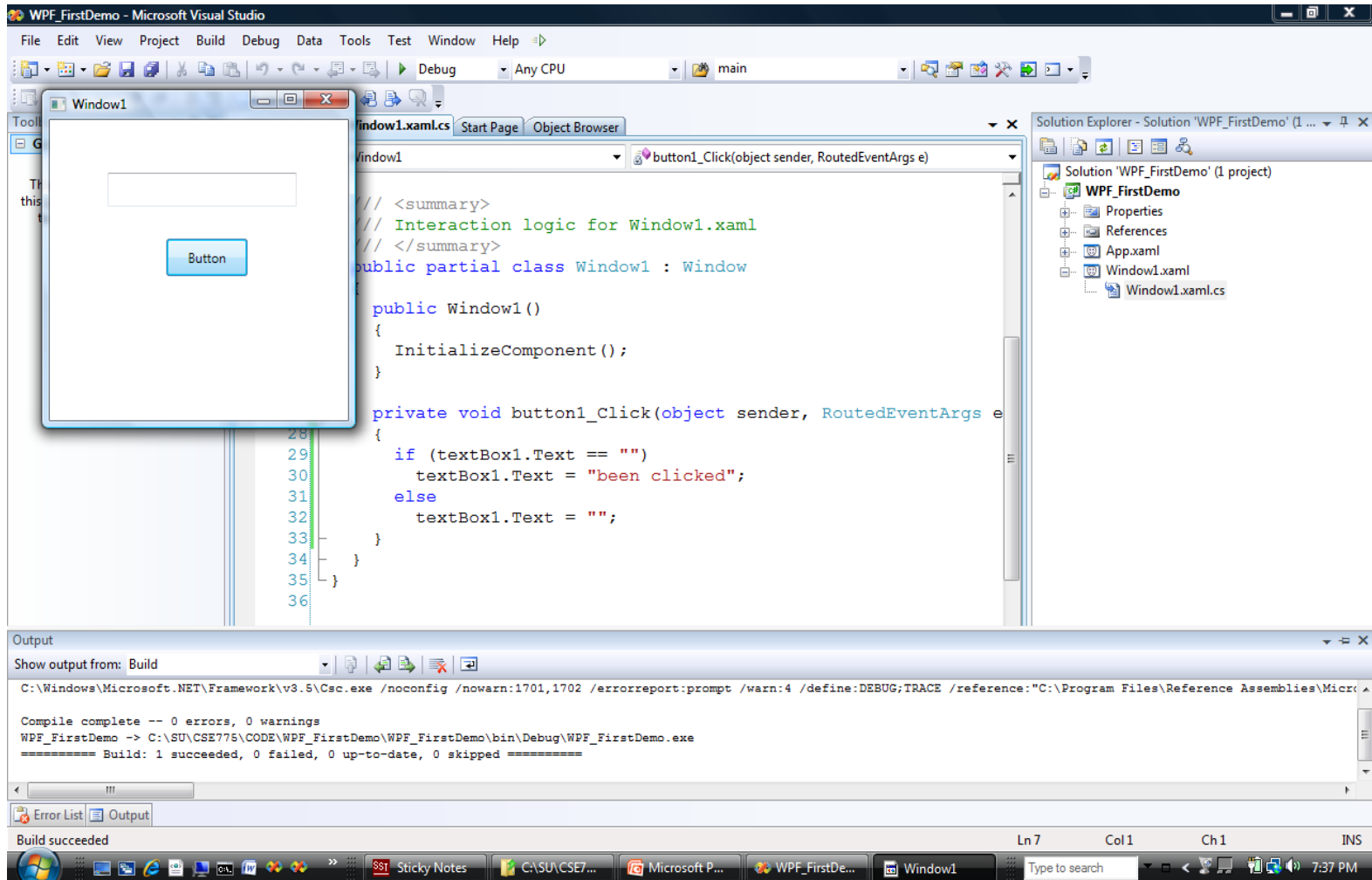
C# Wizard

Default Grid Panel

Like WinForms, But …

It's Easy to Do More Interesting Things

Panels

- Layouts, like the previous page can use:
  - Canvas
    - Simplest, placement relative to two edges
  - StackPanel
    - Horizontal or vertical stacking
  - Grid
    - Uses rows and columns
  - DockPanel
    - Dock to top, right, bottom, left, and all else fills remaining space
  - WrapPanel
    - Horizontal stacking with wrap on overflow
  - All of these can be nested, any one in another

Vector Graphics

- In WPF there is only (usually) one window
  - Controls are not windows!
  - No handles—really, no handles
  - A button is a shape with border, fill, text, animation, and events, like click.
  - There is a Button class, but it is not a .Net control in the traditional sense nor an ActiveX control.
    - Just markup, lines, fills, and events.

Parse Tree

- XAML gets rendered into a parse tree, just like XML—it is XML
  - Inherited properties are based on parent-child relationships in the markup tree
  - Events bubble based on those relationships as well
  - You have direct and simple control over that structure
    - The world is yours!

What Makes WPF Unique?

- Vector graphics with parse-tree structure derived from markup

- Routed events bubble up the parse tree

- Pervasive publish-and-subscribe model
  - Data binding
  - Dependency properties

- Layered on top of DirectX
  - Strong 2D and 3D graphics
  - Animation

- Layout and styles model similar to the best of the web

Auto Parse ▶ Refresh | 🖨 | Courier New ▾ 12 ▾ 🔲 📰 | 25% ▾

Click me!

Click me!

Click me!

```
            TextureCoordinates="0,1 0,0 1,0 1,1 1,1 -0,1 0,-0 1,0 1,1 -0,1 0,-0 1,0 1,0 1,1 -0,1 0,-0 -0,0 1,-0 1,1 0,1 1,-0 1,1 0,1 -0,0"/>
          </GeometryModel3D.Geometry>
        </GeometryModel3D>
      </ModelVisual3D.Content>
    </ModelVisual3D>
  </Viewport3D.Children>
</Viewport3D>
          </ControlTemplate>
        </Setter.Value>
      </Setter>
    </Style>
  </Page.Resources>

  <DockPanel>
    <Button>Click me!</Button>
  </DockPanel>
</Page>
```

3D Hit Testing

3D Perspective Camera
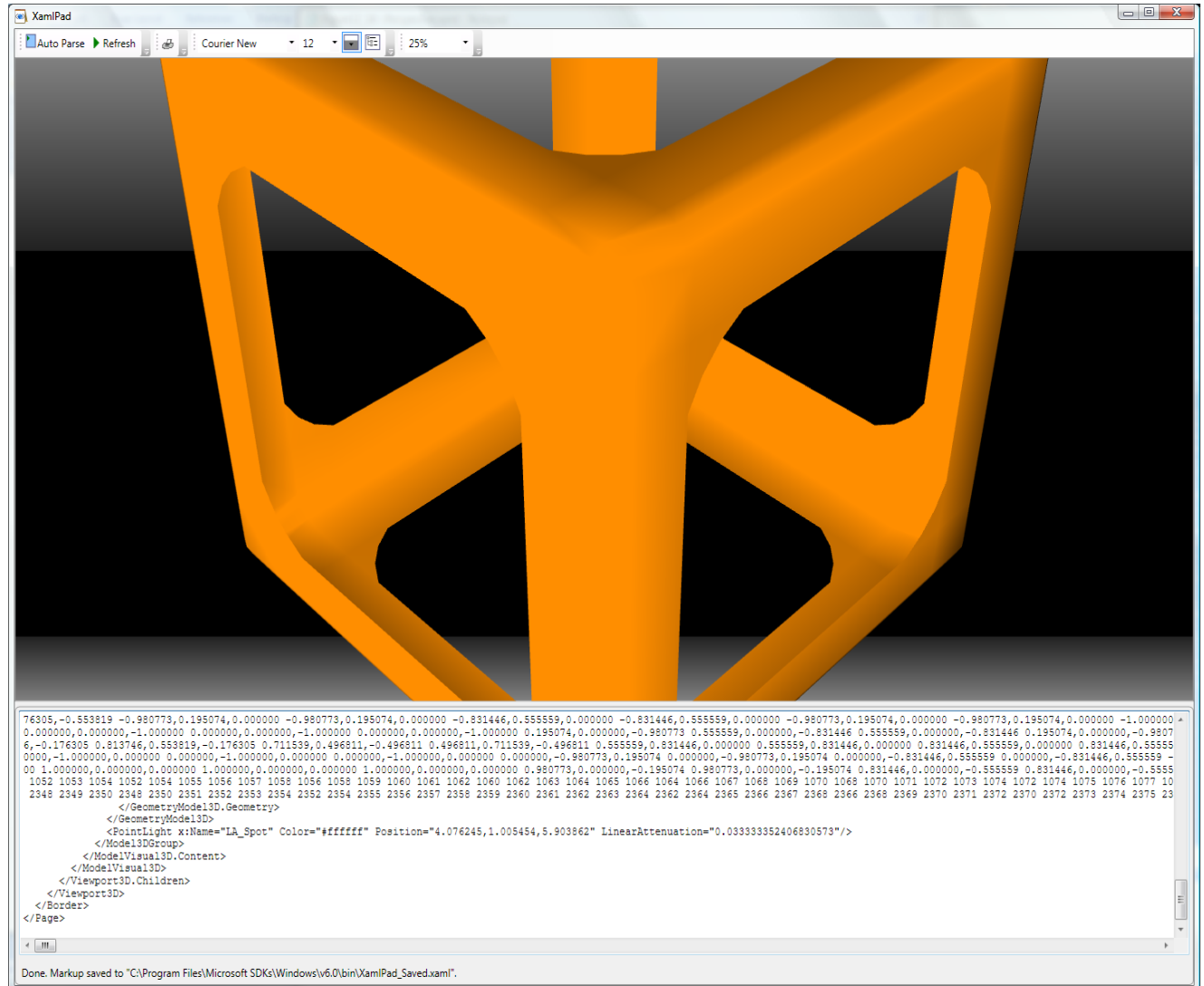
Famous Teapot
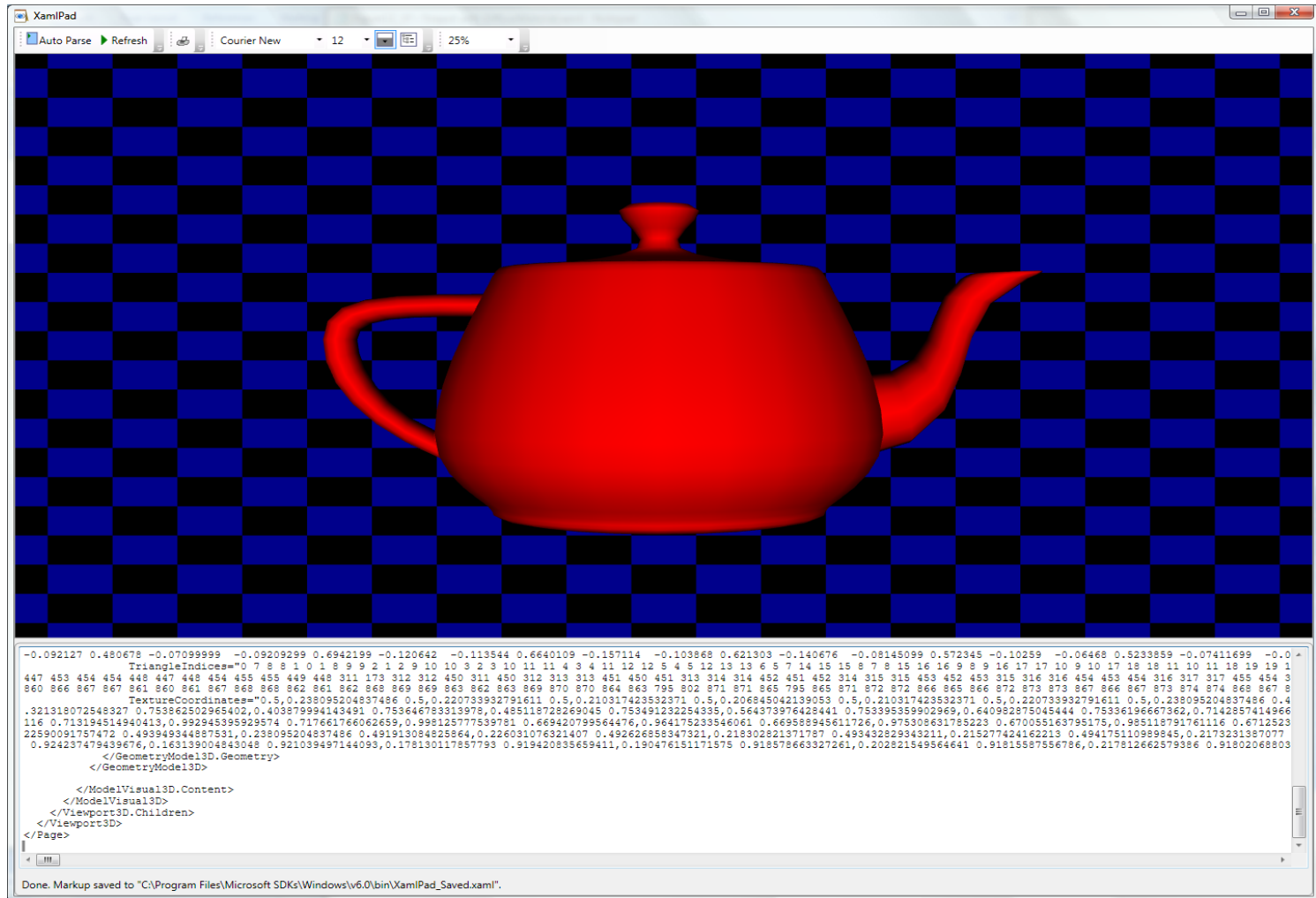
```xml
<Page
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <Page.Background>
    <DrawingBrush Viewport="0,0,0.05,0.05" TileMode="FlipXY">
      <DrawingBrush.Drawing>
        <DrawingGroup>
          <GeometryDrawing Brush="Black" Geometry="M0,0 L1,0 L1,1 L0,1" />
          <GeometryDrawing Brush="DarkBlue" Geometry="M0,0.5 L0.5,0.5 L0.5,1 L0,1" />
          <GeometryDrawing Brush="DarkBlue" Geometry="M0.5,0 L1,0 L1,0.5 L0.5,0.5" />
        </DrawingGroup>
      </DrawingBrush.Drawing>
    </DrawingBrush>
  </Page.Background>
  <Viewport3D>
    <Viewport3D.Camera>
      <PerspectiveCamera Position="0,0,7" LookDirection="0,0,-1"/>
    </Viewport3D.Camera>
    <Viewport3D.Children>
      <ModelVisual3D x:Name="Light">
        <ModelVisual3D.Content>
          <DirectionalLight/>
        </ModelVisual3D.Content>
      </ModelVisual3D>
      <ModelVisual3D>
        <ModelVisual3D.Transform>
          <x:Static Member="Transform3D.Identity"/>
        </ModelVisual3D.Transform>
        <ModelVisual3D.Content>

          <GeometryModel3D x:Name="Teapot">
            <GeometryModel3D.Material>
              <DiffuseMaterial Brush="Red" />
            </GeometryModel3D.Material>
            <GeometryModel3D.BackMaterial>
              <DiffuseMaterial Brush="Red" />
            </GeometryModel3D.BackMaterial>
            <GeometryModel3D.Geometry>
              <MeshGeometry3D
                Positions="0.6788729 0.330678 0  0.669556 0.358022 0  0.6710029 0.374428 0  0.6804349 0.379897 0  0.69502
809  0.164722 0.374428 0.6677769  0.167255 0.379897 0.6768779  0.171187 0.374428 0.6910039  0.175771 0.358022 0.7074749
022 0.350969  -0.62291 0.374428 0.351704  -0.6293589 0.379897 0.356498  -0.6411459 0.374428 0.363938  -0.6555929 0.35802
0.374428 -0.491412  -0.5192369 0.379897 -0.498109  -0.5296319 0.374428 -0.5085049  -0.5417529 0.358022 -0.5206259  -0.55
99  0.329842 0.358022 -0.5975689  0.330577 0.374428 -0.5988199  0.33537 0.379897 -0.6069819  0.34281 0.374428 -0.6196489
39 -0.183211 0.258573  0.9341959 -0.30387 0.265877  0.9436879 -0.419322 0.268519  0.6813349 0.199602 0.412576  0.7319039
0.8397459  -0.268668 -0.058384 0.8894389  -0.279701 -0.183211 0.929081  -0.287004 -0.30387 0.9553229  -0.289646 -0.41932
```
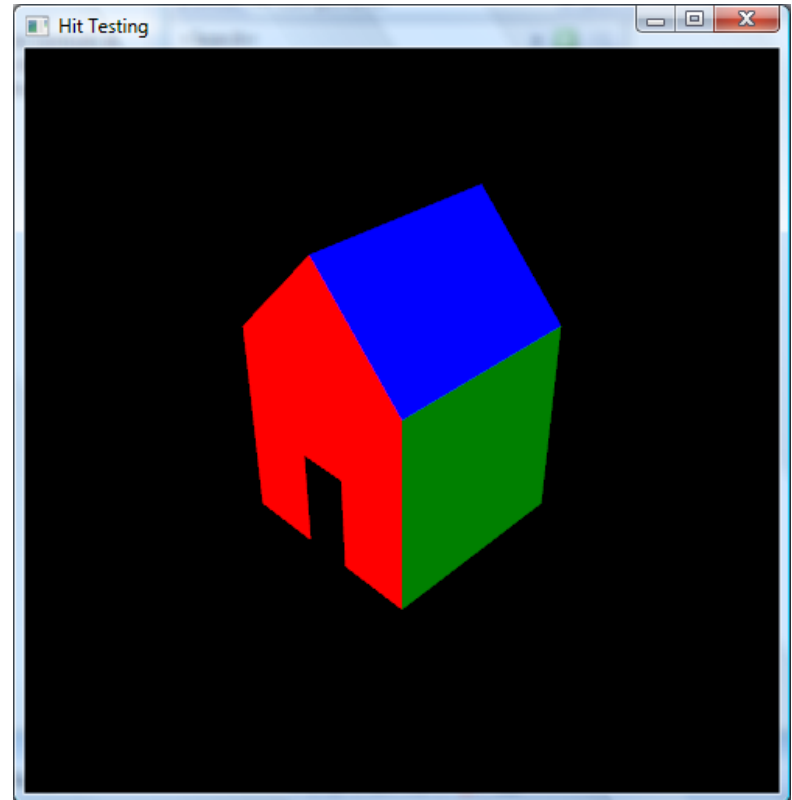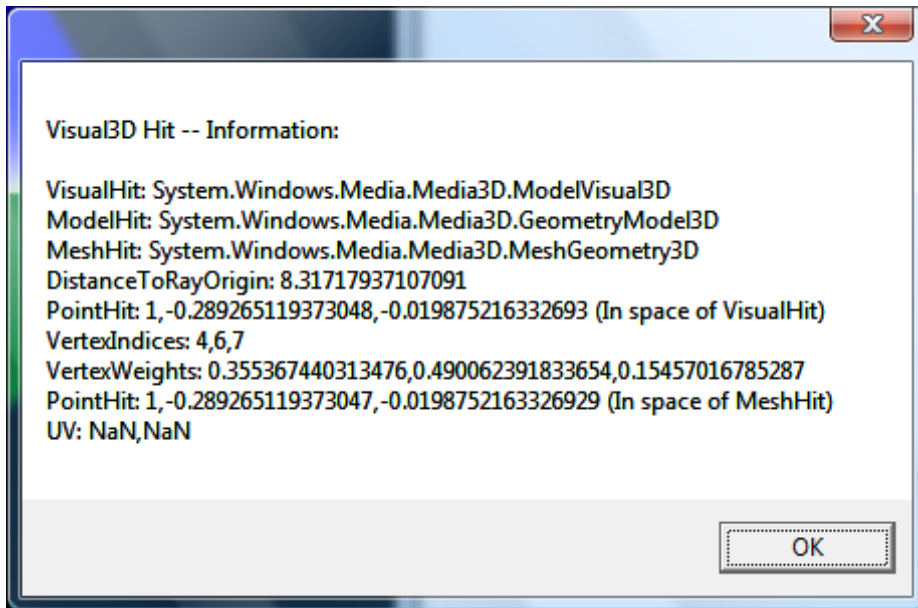
Routed Events

- WPF maps markup elements to UIElements, which derive from ContentControl
  - That means that almost everything can hold content—only one thing unless it's a panel.
  - How does a mouse click *event* on any one of a control's content elements get *routed* to the control?
    - By walking the XAML parse tree until it finds a parent that handles that event.

Adding Event Handlers

- You will find that property sheets show events as well as properties
  - Click on the lightning bolt to see the event sheet.
  - You subscribe by clicking on an event entry.
- You can also add event handlers quickly in XAML:
  - Go to the XAML, type a space after the tag for the element you want to handle the event
    - That gets you a context menu (via IntelliSense) and you just double-click on the desired event, which adds an event attribute

Attached Properties

- Buttons, ListBoxes, Images, etc., do not have Dock properties.

- However, when you place one of these in a DockPanel, you find that it has had Dock **properties attached**.

<Image Source="./help.png"
  DockPanel.Dock="Top" Height="213"
  ImageFailed="Image_ImageFailed" />

DependencyObject Class

- Attached properties work because all WPF controls derive from the DependencyObject class.

  - DependencyObject class supports adding an arbitrary number of dependency properties.

- App
- □ Window1
  - Base Types
    - ○ IComponentConnector
    - □ Window
      - □ ContentControl
        - □ Control
          - □ FrameworkElement
            - □ ○ IFrameworkInputElement
              - ○ IInputElement
            - ○ IInputElement
            - ○ ISupportInitialize
            - □ UIElement
              - ○ IAnimatable
              - ○ IInputElement
              - □ Visual
                - □ DependencyObject
                  - □ DispatcherObject
                    - Object
      - ○ IAddChild

- GetValueEntry(System.Windows.EntryIndex, System.Windows.DependencyProperty, Syst
- GetValueSource(System.Windows.DependencyProperty, System.Windows.PropertyMeta
- GetValueSource(System.Windows.DependencyProperty, System.Windows.PropertyMeta
- HasAnyExpression()
- HasExpression(System.Windows.EntryIndex, System.Windows.DependencyProperty)
- InvalidateProperty(System.Windows.DependencyProperty)
- InvalidateSubProperty(System.Windows.DependencyProperty)
- LookupEntry(int)
- NotifyPropertyChange(System.Windows.DependencyPropertyChangedEventArgs)
- NotifySubPropertyChange(System.Windows.DependencyProperty)
- OnInheritanceContextChanged(System.EventArgs)

Dependency Properties

- A dependency property is a property that is registered with the WPF dependency property system. Two uses:
  - Backing an object property with a dependency property, provides support for data binding, styling, and animation. Examples include Background and FontSize properties.
  - Creating attached properties. Attached properties are properties that can be set on ANY DependencyObject types. An example is the Dock property.
- You can find an example of the definition and use of a custom dependency property here.
- Dependency properties are a publish-and-subscribe system.

Dependency Property Links

Josh Smith's Blog

Switch on the Code Blog

Learn WPF site

Property Syntax

- Two syntax forms:
  - XAML attribute:
    `<button ToolTip="Button Tip />`
  - Property element syntax:
    ```
    <Button>
      <Button.Background>
        <SolidColorBrush Color="#FF4444FF" />
      </Button.Background>
      Some Button Text
    </Button>
    ```

Markup Extensions

- Sometimes you need to assign a property from some source at run-time. For that you use markup extensions:

```
<Button Foreground="{x:static
       SystemColors.ActiveCaptionBrush}" >
   Some text
</Button>
```

Inline Styles

- Collections of property values:
  - <Button.Style>
    <Style>
    <Setter Property="Button.FontSize" Value="32pt" />
    <Setter Property="Button.FontWeight" Value="Bold" />
    </Style>
    </Button.Style>

Named Styles

- Collections of property values:
  - ```
    <Window.Resources>
      <Style x:Key="myStyle" TargetType="{x:Type Control}">
        <Setter Property="FontSize" Value="32pt" />
        <Setter Property="FontWeight" Value="Bold" />
      </Style>
    </Window>
    ```

Binding

- Binding infrastructure allows you to set up a one-way or two-way updating of property values that happens when the source changes.
- This requires two things:
  - A dependency object
    - Has its own dispatcher thread
  - Support for INotifyPropertyChanged interface

Binding

- Objects that implement INotifyPropertyChanged interface raise events when the property has changed.

- Data binding is the process of registering two properties with the data-binding engine and letting the engine keep them synchronized.

- You will find an example in the Wpf_AttachedProperties demo code.

Binding Links

[MSDN Article by John Papa](#)

[CodeProject article by Josh Smith (part of a tutorial series)](#)

[Bea (Costa) Stollnitz](#)

Control Templates

- With control templates you can change the look and feel of existing controls and support making your own controls:
  - <Button.Template>
    <ControlTemplate>
    <Grid><Rectangle /></Grid>
    </ControlTemplate>
    </Button.Template>

Navigation

- You can use instances of the Page and Frame classes to set up a navigation structure resembling web applications.
  - Pages go in NavigationWindow instances and Frames go in Windows and Pages.
  - This is a good alternative to tabbed displays.

Special Classes

- ContentControl
  - All UIElements derive from this.
  - Content can be text, a tree of elements, or a .Net object which can be displayed using a data template
- Dependency object
  - Derives from DispatcherObject
  - Supports data binding, styling, animation, property inheritance, and property change notifications
- WindowsFormsHost
  - Supports hosting controls based on HWNDs

Special UI Elements

- ViewBox
  - Resizes content to fit available space
- UserControl
  - Way to build custom controls as collections of elements on a panel
- Animatable
  - Provides hooks for DirectX to change elements properties over time, e.g., position, size, color, …
- FlowDocument
  - FlowDocumentScrollViewer
  - FlowDocumentPageViewer
- MediaElement
  - Play media on load or on request, e.g., wma, wmv, mp3, …

The End