Getting started on the Surface

Doug Blank
Bryn Mawr College
Programming Paradigms
Spring 2010

parts based on Microsoft tutorials

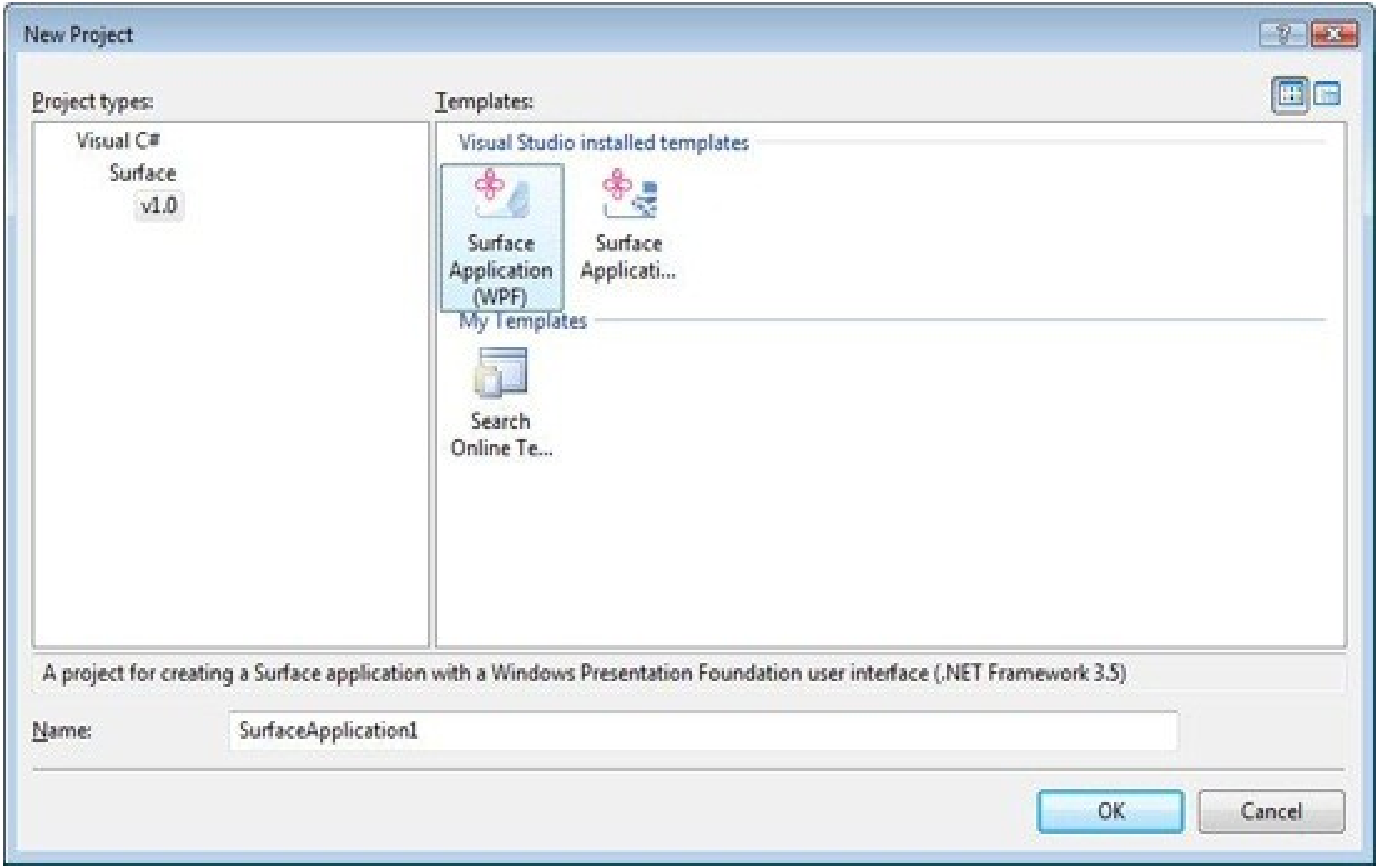
Two Layers

- Core Layer
  - Use under any GUI toolkit
    - 2D
    - 3D (high-performance graphics)
    - XNA, Game Development (for Xbox, Zune, Windows)
  - Based on HWND “window handles”
- Presentation Layer
  - Built on top of WPF

Visual Studio

- To start the Visual Studio project
  - Install the Microsoft Surface SDK
  - Open Visual C# 2008 (or Microsoft Visual Studio 2008)
  - On the File menu, click New
  - Click Project, expand Visual C#, expand Surface, and then click v1.0
  - In the Templates pane, under Visual Studio installed templates, click Surface Application (WPF)

Hello World



A project for creating a Surface application with a Windows Presentation Foundation user interface (.NET Framework 3.5)

Name: SurfaceApplication1

Editing the Program

- XAML
  - Extensible Application Markup Language
  - “user interface control language”
  - Elements are CLR objects
  - Attributes are properties and events
- C# code

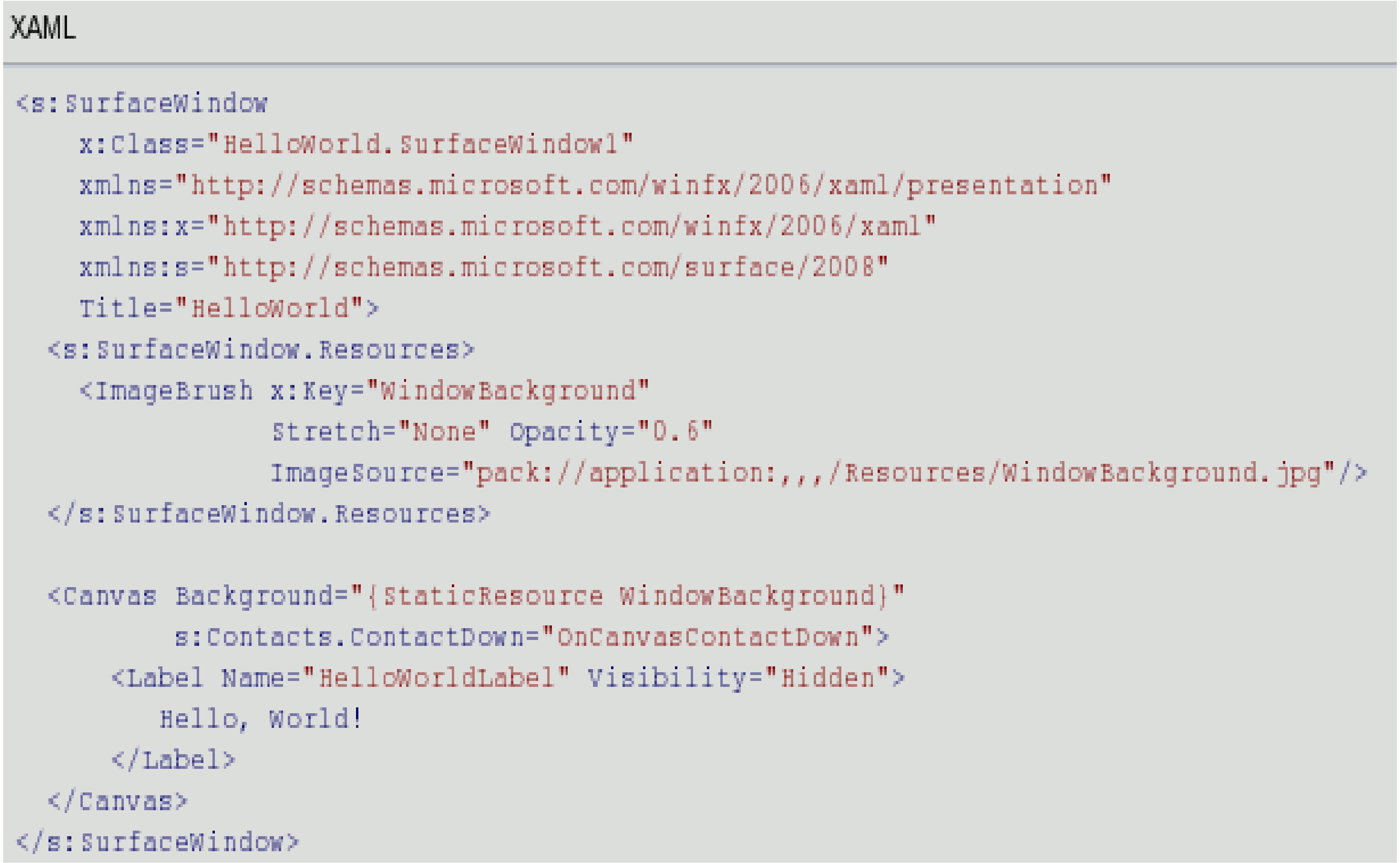
Edit the XAML file

- Open the SurfaceWindow1.xaml file (the XAML file is located in the Solution Explorer window, within the HelloWorld node)
- Edit SurfaceWindow1.xaml to add controls or elements to the application
- Change the Grid to a Canvas. The SurfaceWindow element can have only one primary child element, and the Canvas element enables you to position child elements exactly.
- Add a ContactDown attribute to the Canvas element in the XAML file.

Edit the XAML file, cont

- Type s:Contacts.ContactDown="" and press ENTER to accept the default event name "Canvas_ContactDown" and the event method shell is added to the C# file for you.
- Add a Label element as a child of the Canvas element.
- Add Name="helloWorldLabel" as an attribute on the Label element so you can access the Label element by name in the C# code.
- Add Visibility="Hidden" as an attribute in the Label element to specify that the Label element is initially hidden when the SurfaceWindow is started.
- The Label element will appear for the first time when the first Surface contact is made.
- Add Hello, World! between the Label start and end tags to initialize the Label element with text.

Edit the XAML file, cont



XAML

```
<s:SurfaceWindow
    x:Class="HelloWorld.SurfaceWindow1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:s="http://schemas.microsoft.com/surface/2008"
    Title="HelloWorld">
  <s:SurfaceWindow.Resources>
    <ImageBrush x:Key="WindowBackground"
                Stretch="None" Opacity="0.6"
                ImageSource="pack://application:,,,/Resources/WindowBackground.jpg"/>
  </s:SurfaceWindow.Resources>

  <Canvas Background="{StaticResource WindowBackground}"
          s:Contacts.ContactDown="OnCanvasContactDown">
      <Label Name="HelloWorldLabel" Visibility="Hidden">
         Hello, World!
      </Label>
  </Canvas>
</s:SurfaceWindow>
```

Edit the C# code

- In Solution Explorer, expand the SurfaceWindow1.xaml node, and then open the SurfaceWindow1.xaml.cs file created by the WPF template.
- Add an OnCanvasContactDown event handler method to the Window1 class that is defined in the SurfaceWindow1.xaml.cs file.
- Add code to the OnCanvasContactDown event handler so that the label appears at the point of contact every time that a contact touches the Microsoft Surface screen.

Edit the C# code, cont

```
#region OnCanvasContactDown
private void OnCanvasContactDown(object sender, ContactEventArgs e)
{
    // Get the position of the current contact.
    Point contactPosition = e.Contact.GetPosition(this);

    // Set the X and Y position of HelloWorldLabel
    // in relation to the canvas.
    Canvas.SetLeft(HelloWorldLabel, contactPosition.X);
    Canvas.SetTop(HelloWorldLabel, contactPosition.Y);

    // Make the label visible.
    HelloWorldLabel.Visibility = Visibility.Visible;
}
#endregion
```

Running the Application

Double-click the **SurfaceInput** icon on the desktop. If SurfaceInput is not running, your application will not receive Contact events. If you are running on a separate computer, start Surface Simulator from the Start menu. Surface Simulator starts **SurfaceInput** when it is launched.

The last step is to build your project by pressing the F5 key. When you touch the Microsoft Surface screen, or click the mouse inside the Surface Simulator window, a Label appears and displays "Hello, World!"