Interoperability

Outline

- Hybrid applications
- Interop limitations
- Combinations

Mixing UI Technologies

- **Few projects are 'green field'**

- **Mixing WPF with other UI useful:**
  - Using legacy controls
  - Supporting legacy plug-ins
  - Introducing WPF 'islands'
  - Migration path to WPF

WPF and HWNDs

- **Classic Win32: one HWND per control**
- **WPF: one big HWND**
  - Popups get their own

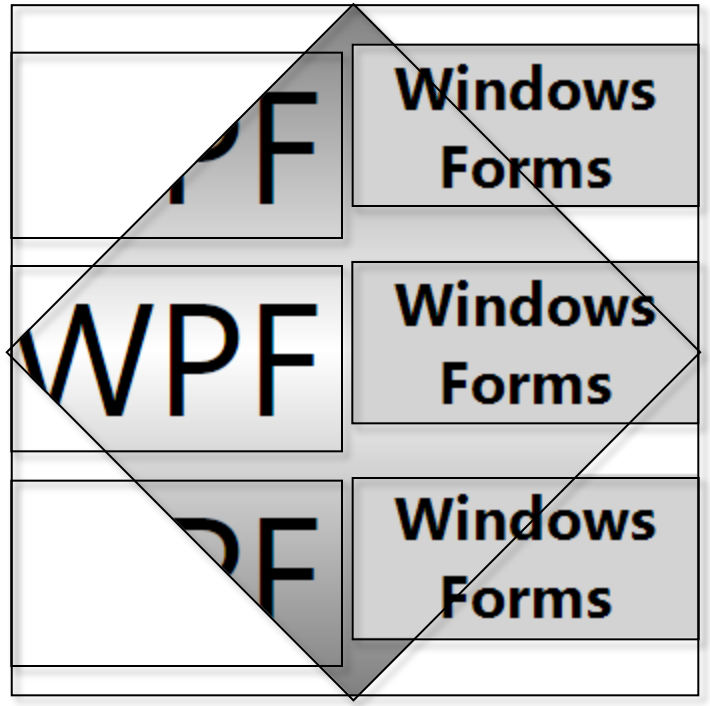
Interop Not Free

- **Development costs:**
  - Lower in short term – no rewrite
  - Beware of long term costs
- **Runtime costs**
  - Working set overhead of multiple UI frameworks
  - Some inefficiency at boundaries
- **Fit and finish**

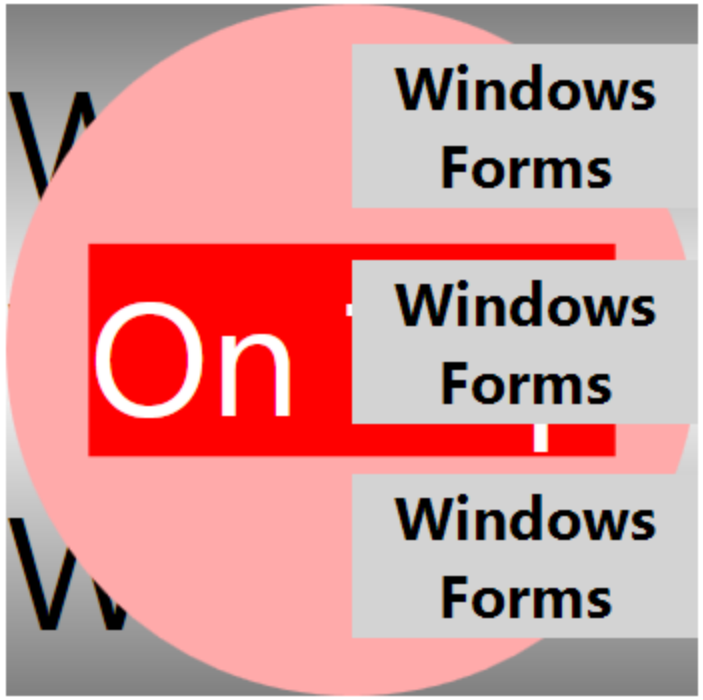
Limitations of Interop

- **'Airspace' – 1 technology per pixel**
  - WPF
  - Win32
  - DirectX
- **Transforms**
- **Events**
- **Multi-level hybrid nesting not supported**

Clipping

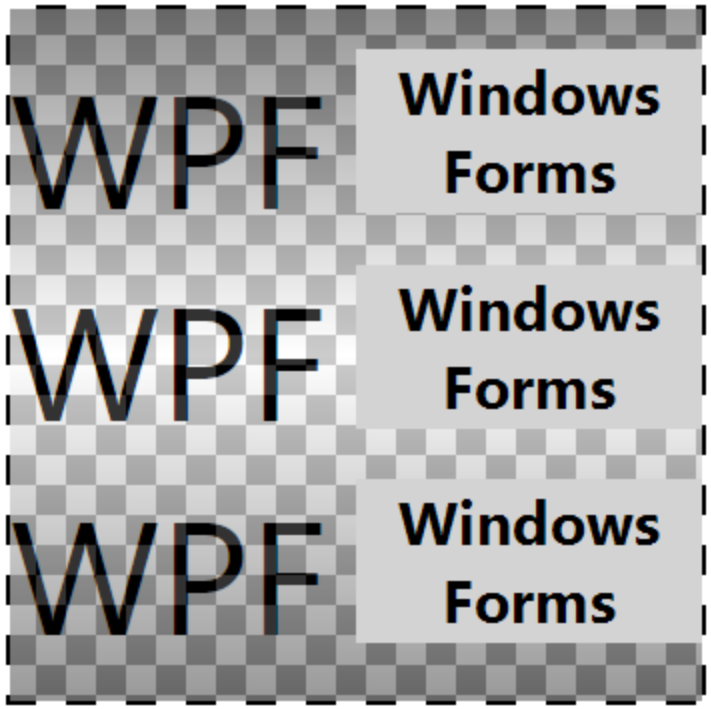


Z Order



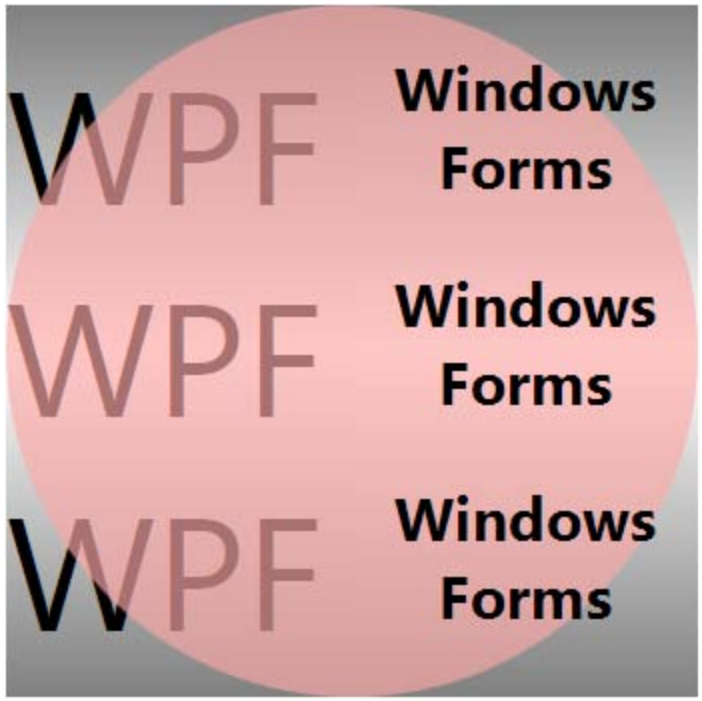
Animation

Opacity Property



Pseudo Transparency

```
<wfi:WindowsFormsHost Background="Transparent" >...
```



Multiple Top-Level Windows

- **Airspace restrictions are per-top-level window**
- **Multiple windows solve some problems**
  - Popups
  - Transparency – Window.AllowsTransparency

Input Differences

- Mouse events and IsMouseOver
- Keyboard and Focus events, and IsFocusWithin

Interop Combinations

Technology	WPF Inside	WPF Outside
Windows Forms	Easy	Easy
Win32	Fairly easy	Fairly easy
ActiveX	Hard	Easy
HTML	Easy (with limitations)	Easy

Win32 Outside

- **HwndSource**
  - Underpins Window
  - All on-screen WPF content relies on HwndSource

```
HwndSource src = (HwndSource)
     PresentationSource.FromVisual(myWindow);

IntPtr hWnd = src.Handle;
```

```
HwndSource src = new HwndSource(
  myWindowClass, myWindowStyle, myWindowExStyle,
  xPos, yPos, width, height,
  "My HWND Title", parentHwnd);

src.RootVisual = myWindowContent;
```

Win32 Inside

- HwndHost

```
public abstract class HwndHost : ...
{
  ...

  protected abstract HandleRef BuildWindowCore(HandleRef hwndParent);

  protected abstract void DestroyWindowCore(HandleRef hwnd);

  protected virtual IntPtr WndProc(...);

  ...
}
```

HwndHost Keyboard Handling

- TranslateAccelerator
- TabInto
- OnMnemonic

Message Pumps

- **Dispatcher provides message loop**
- **Win32 application may need custom loop**
  - ComponentDispatcher enables WPF integration

ComponentDispatcher Methods	
RaiseThreadMessage	Call for each message
PushModal	Call when entering modal loop
PopModal	Call when exiting modal loop
RaiseIdle	Call when suitable time for idle processing

Windows Forms Outside

```
ElementHost wpfHost = new ElementHost();

wpfHost.Child = myWpfElement;

wpfHost.Dock = DockStyle.Fill;
myWindowsFormsContainer.Controls.Add(wpfHost);
```

Windows Forms Inside

```
WindowsFormsHost wfHost = new WindowsFormsHost();

wfHost.Child = myWindowsFormsControl;

DockPanel.SetDock(wfHost, Dock.Top);
myWpfPanel.Children.Add(wfHost);
```

Windows Forms and Layout

- **Normal Windows Forms layout works as expected**
  - Docking
  - Anchoring
  - Auto-sizing

- **WPF LayoutTransform – scaling only**
  - Uses Windows Forms auto-scaling inside WindowsFormsHost
  - Other transforms raise LayoutError event

Windows Forms Ambient Properties

- **WindowsFormsHost and ElementHost have a PropertyMap**
  - Associates callbacks with named WPF properties
  - Default mappings supplied for most common properties

```
elemHost.PropertyMap.Add("Text",
    delegate(object host, string prop, object val)
    {
        ContentControl c =
                (ContentControl) elemHost.Child;
        c.Content = val;
    });
```

Summary

- Hybrid applications
- Interop limitations
- Combinations