Introduction to Windows Presentation Foundation

DEVELOPMENTOR

DEVELOPING PEOPLE WHO DEVELOP SOFTWARE

What is Windows Presentation Foundation (WPF)?

- WPF is the successor to Windows Forms GUI technology from Microsoft
  - designed to build dynamic and interactive user interfaces

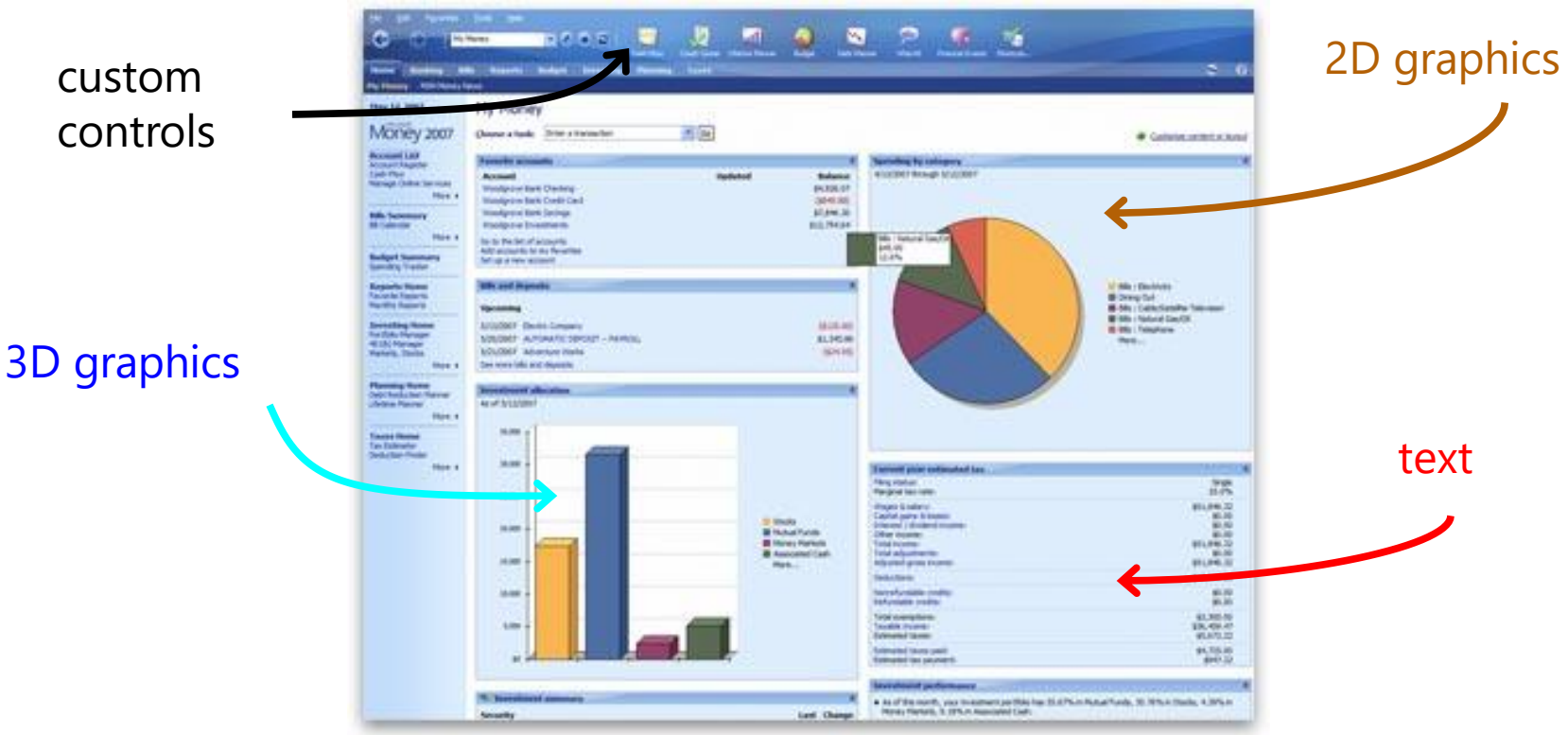
"In 2001 a new team was formed in Microsoft with a simple sounding mission – build a unified presentation platform that could eventually replace User32/GDI32, VB6, MSHTML, and Windows Forms.

The goal being to produce a best of breed platform that could really be a quantum leap forward."

-- Chris Anderson, WPF Architect

Motivation [multiple skillsets]

- Modern applications use a mix of technologies
  - each requires extensive time to master

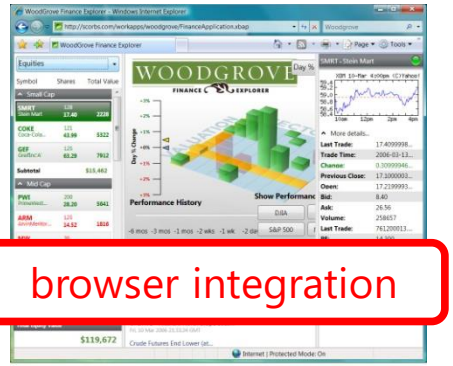


Introducing Windows Presentation Foundation

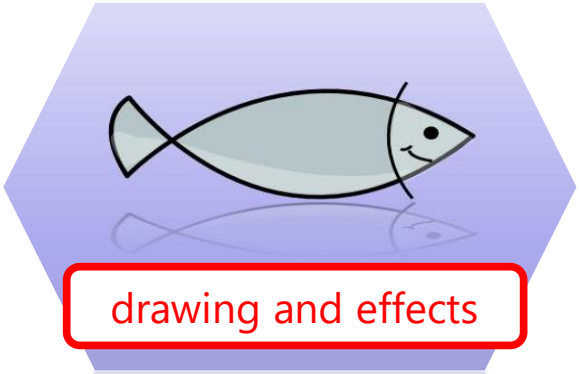
- WPF provides broad integration of technologies
  - single, consistent framework exposes all features

rich document support

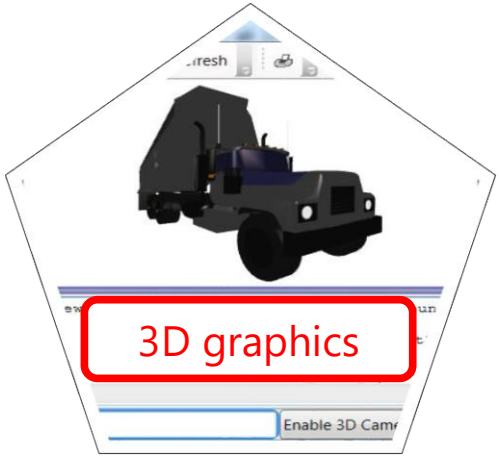
custom controls



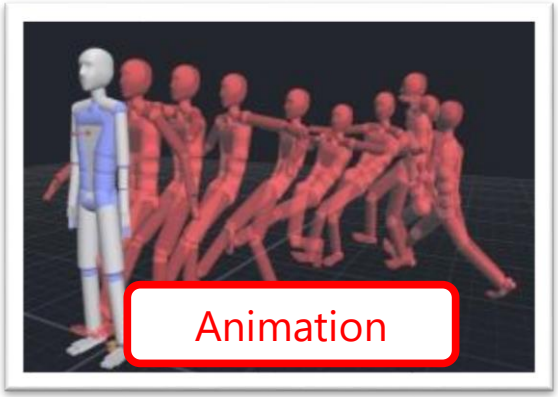
browser integration



drawing and effects



3D graphics



Animation

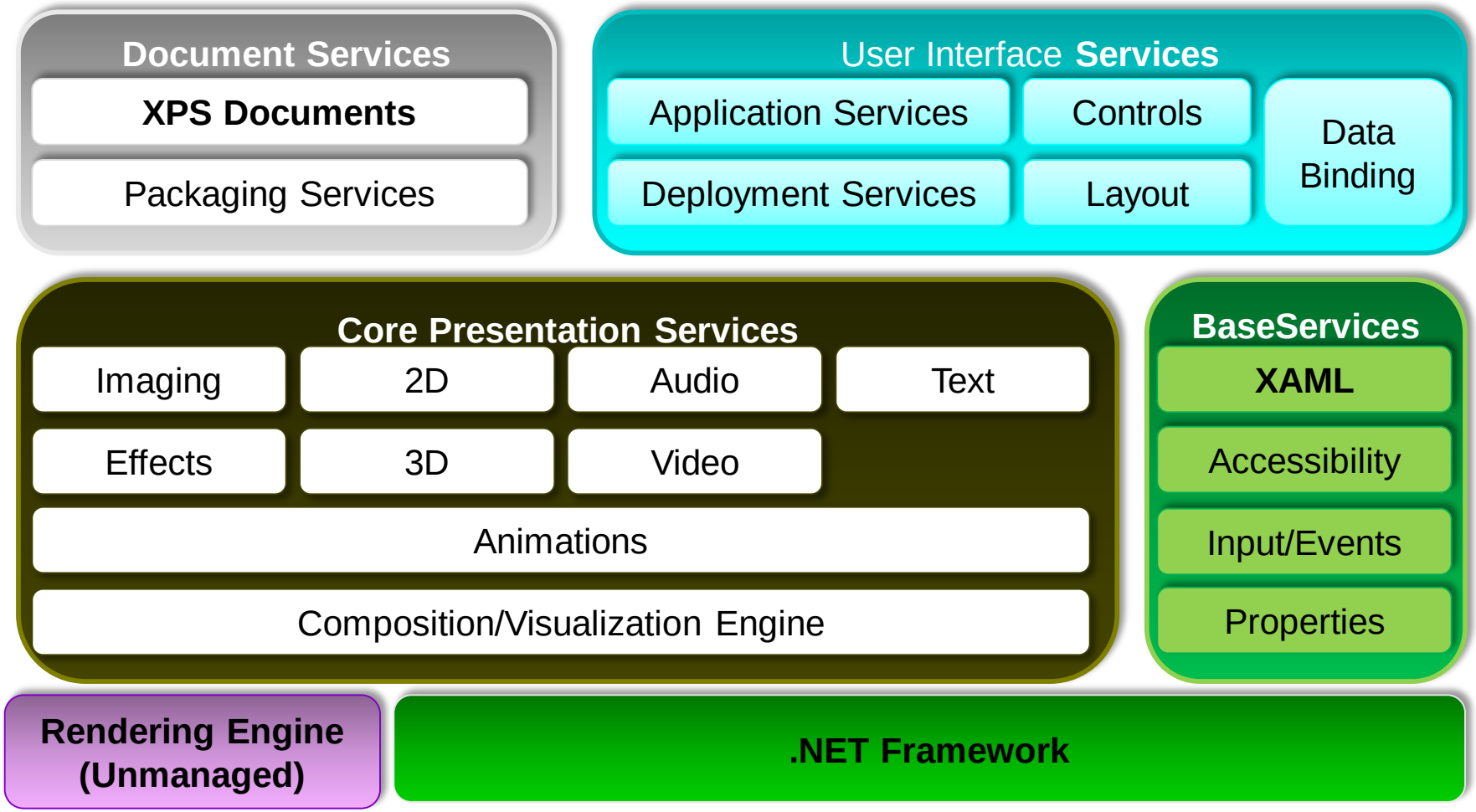
Pick your favorite flavor

- WPF/XAML technology is common to many Microsoft UI initiatives today

Benefits

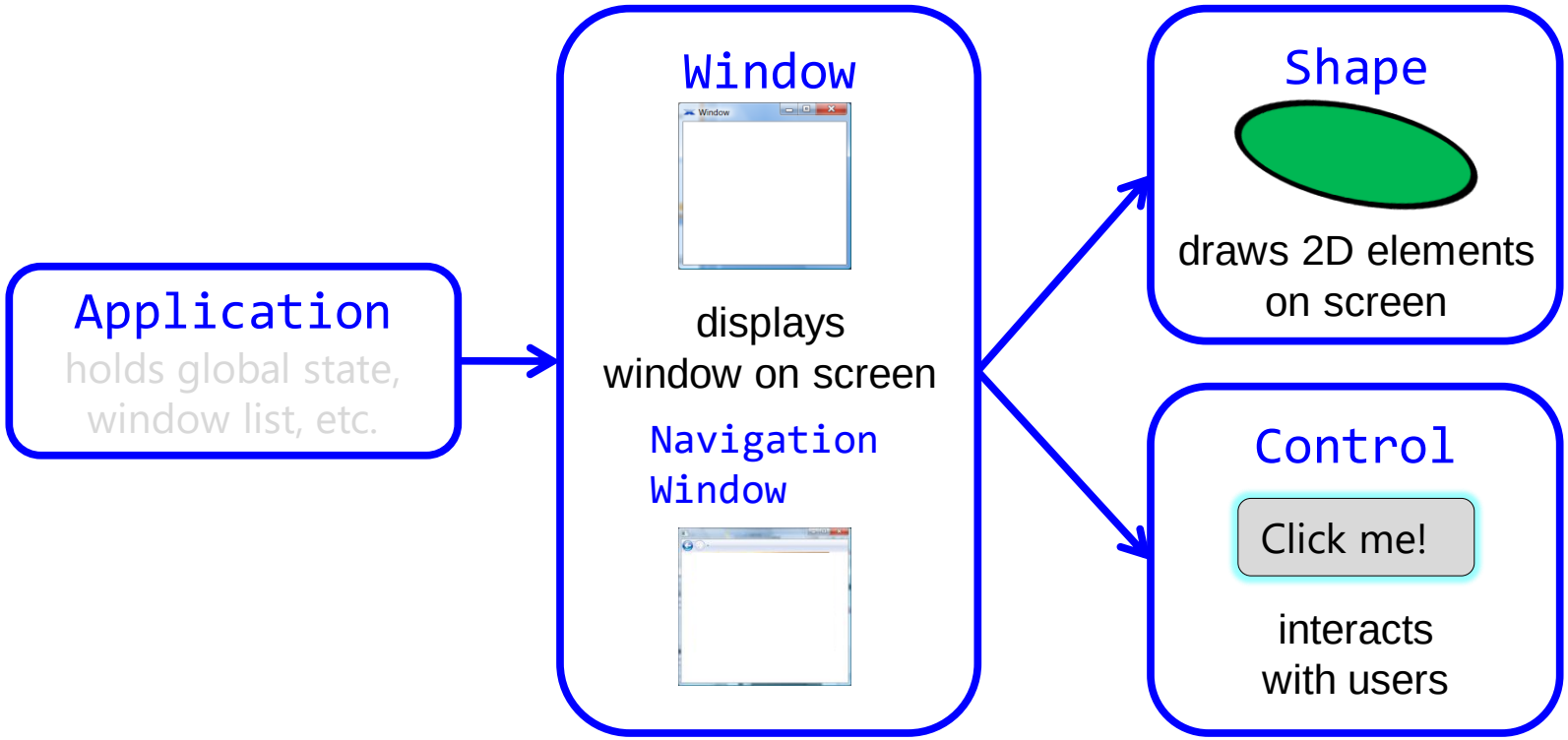
- Vector-based composing rendering engine
  - automatic scaling, overlay support and full transparency
  - retained mode rendering cuts down on repainting work
- Comprehensive and consistent architecture
  - 2D + 3D + animations + media + documents, etc.
  - highly extensible
  - designed for designer + developer workflow
- Enables new classes of applications never before possible

WPF Services – "The Big Picture"



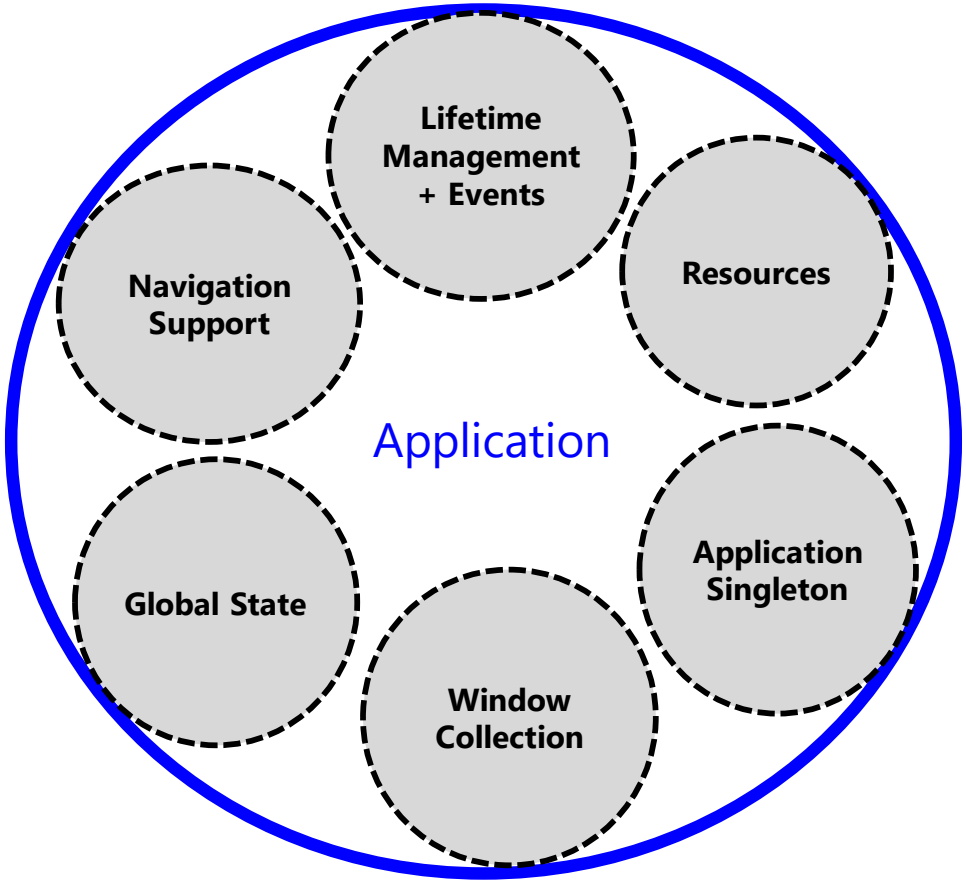
Building Applications

- Several classes work together to build a WPF application



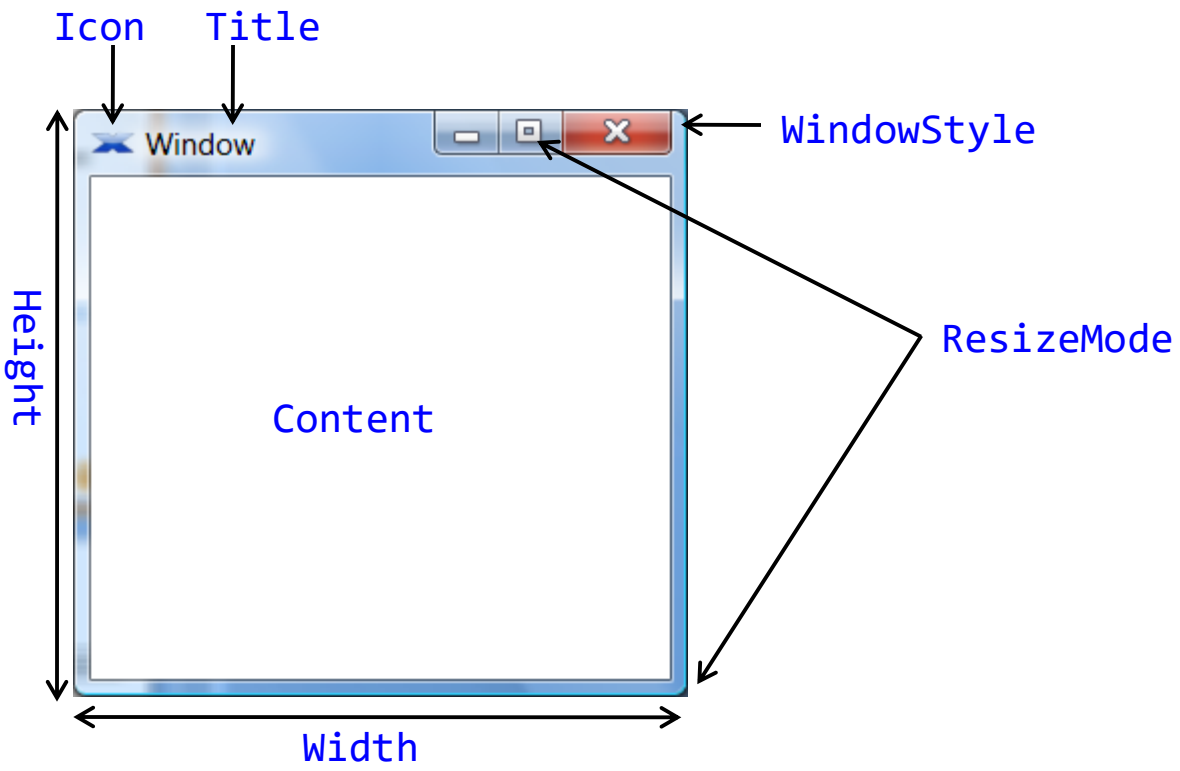
System.Windows.Application

- The `Application` class provides access to application-level resources



System.Windows.Window

- The `Window` class creates a top-level window on the screen
  - has properties to control characteristics
  - has events to monitor state and lifetime changes



Hello, WPF

- `Window` and `Application` work together to create a program

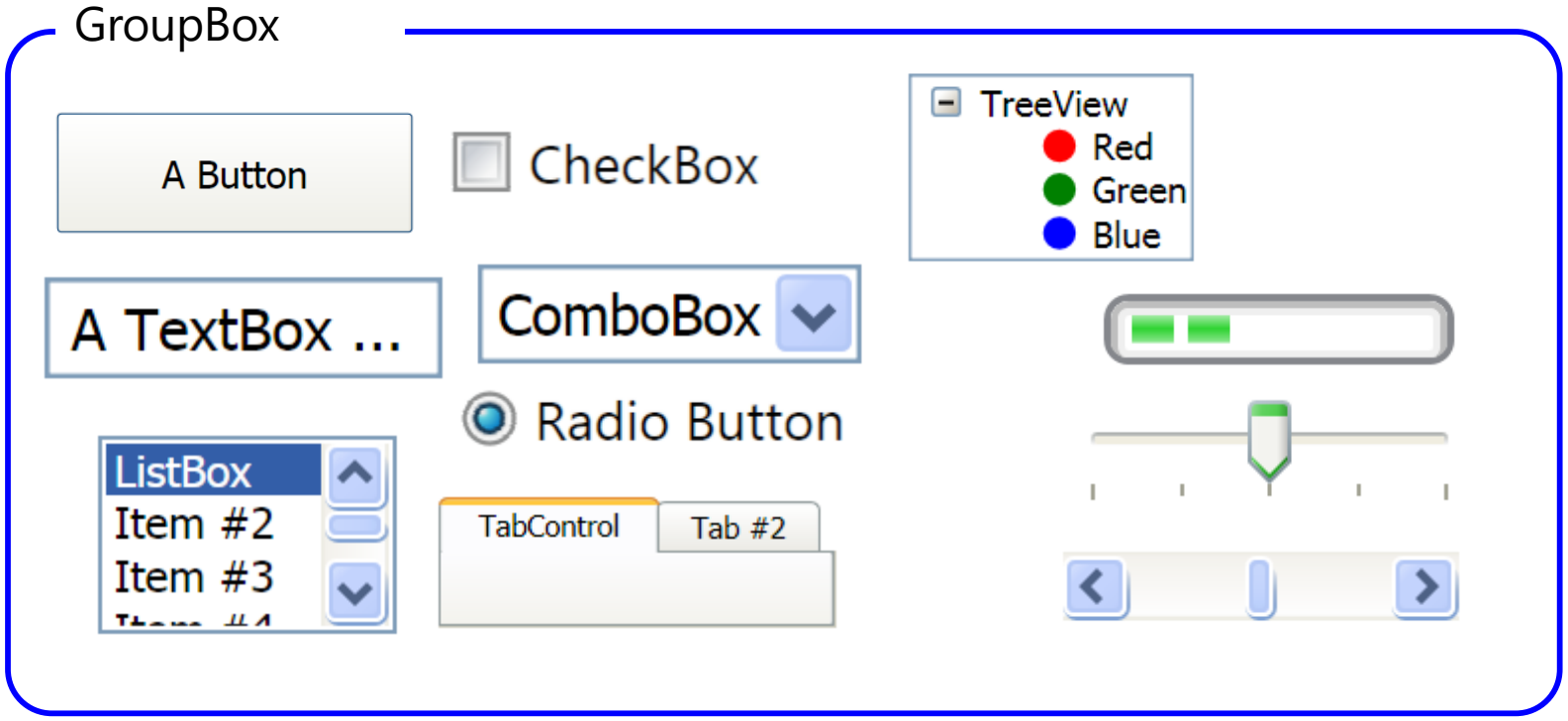
```
using System;
using System.Windows;

class SayHello
{
    [STAThread]
    static void Main()
    {
        Window mainWindow = new Window();
        mainWindow.Title = "Hello from WPF!";
        mainWindow.Width = mainWindow.Height = 300;

        Application app = new Application();
        app.Run(mainWindow);
    }
}
```

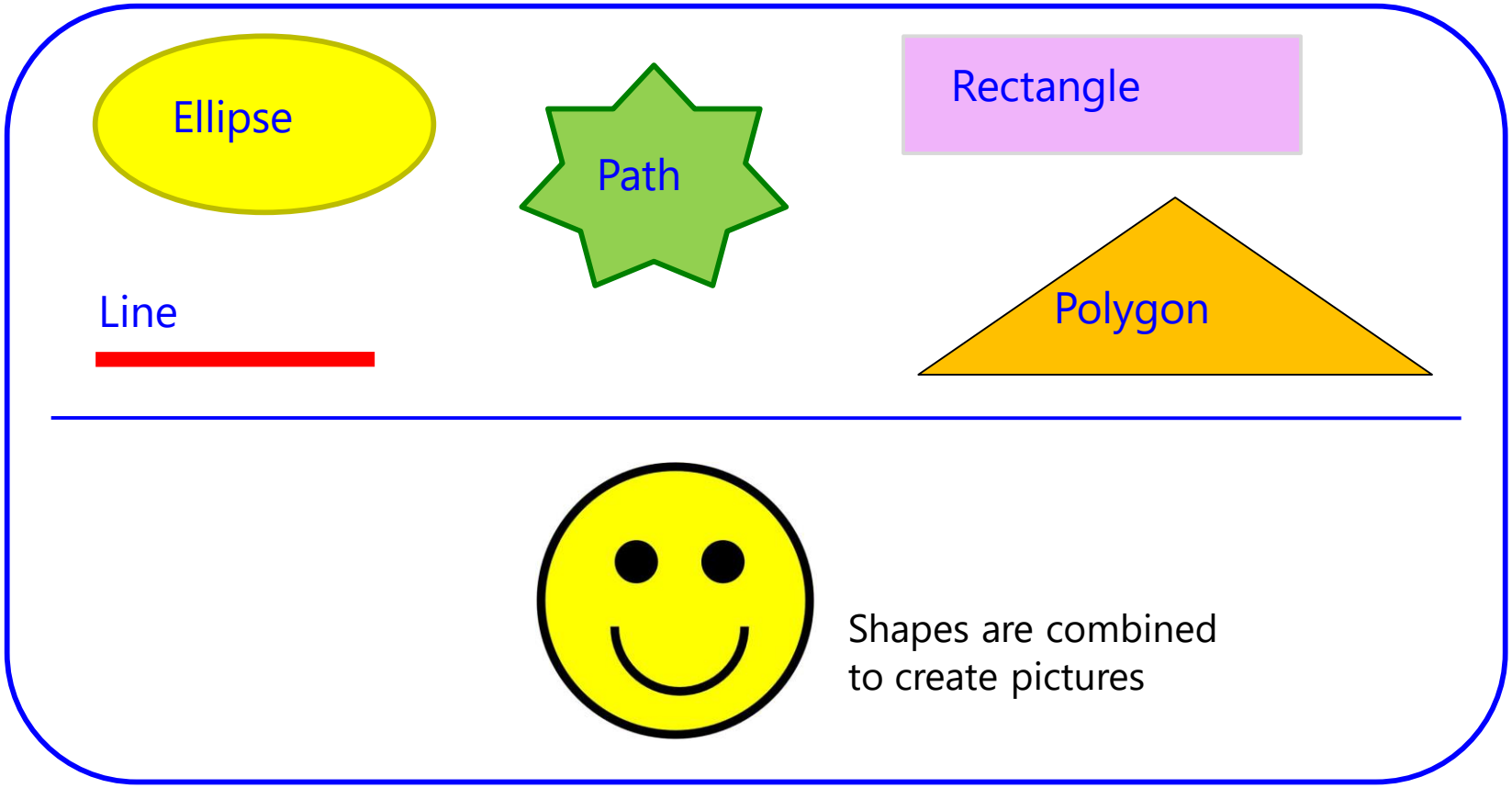
System.Windows.Controls

- Controls allow interaction with the user
  - display feedback
  - receive input and focus



System.Windows.Shapes

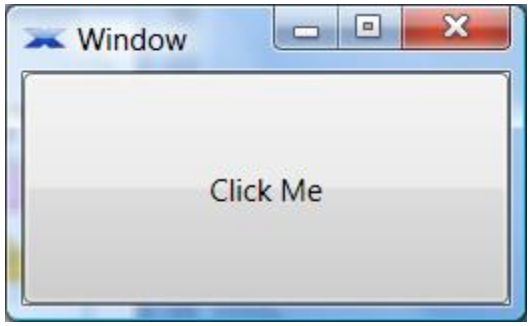
- Shape objects can be used to add 2D elements to window
  - can react to input but cannot have focus or children



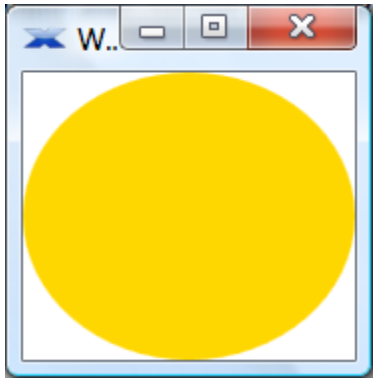
Adding Controls and Shapes to Windows

- `Window.Content` property is used to show a visual object
  - set it to display a **Shape** or **Control**

```
class MyButtonWindow : Window
{
    public MyButtonWindow()
    {
      Button btn = new Button();
      btn.Content = "Click Me";
      this.Content = btn;
    }
}
```




```
class MyShapeWindow : Window
{
    public MyShapeWindow()
    {
        Ellipse el = new Ellipse();
        el.Fill = Brushes.Gold;
        this.Content = el;
    }
}
```



Content limitations

- Most elements are limited to a single piece of Content

```
class MyButtonWindow : Window
{
    public MyButtonWindow()
    {
      Button btn = new Button();
      btn.Content = "Click Me";
      Ellipse el = new Ellipse();
      el.Fill = Brushes.Gold;

      this.Content = ???;
    }
}
```

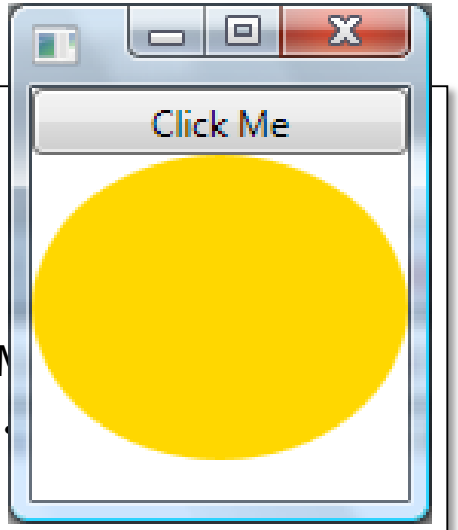
… anything more complex requires layout

Managing Layout

- Panels are used to organize children in a predictable way
  - dynamically sizes and positions child controls and shapes
  - allows complete flexibility in visual organization

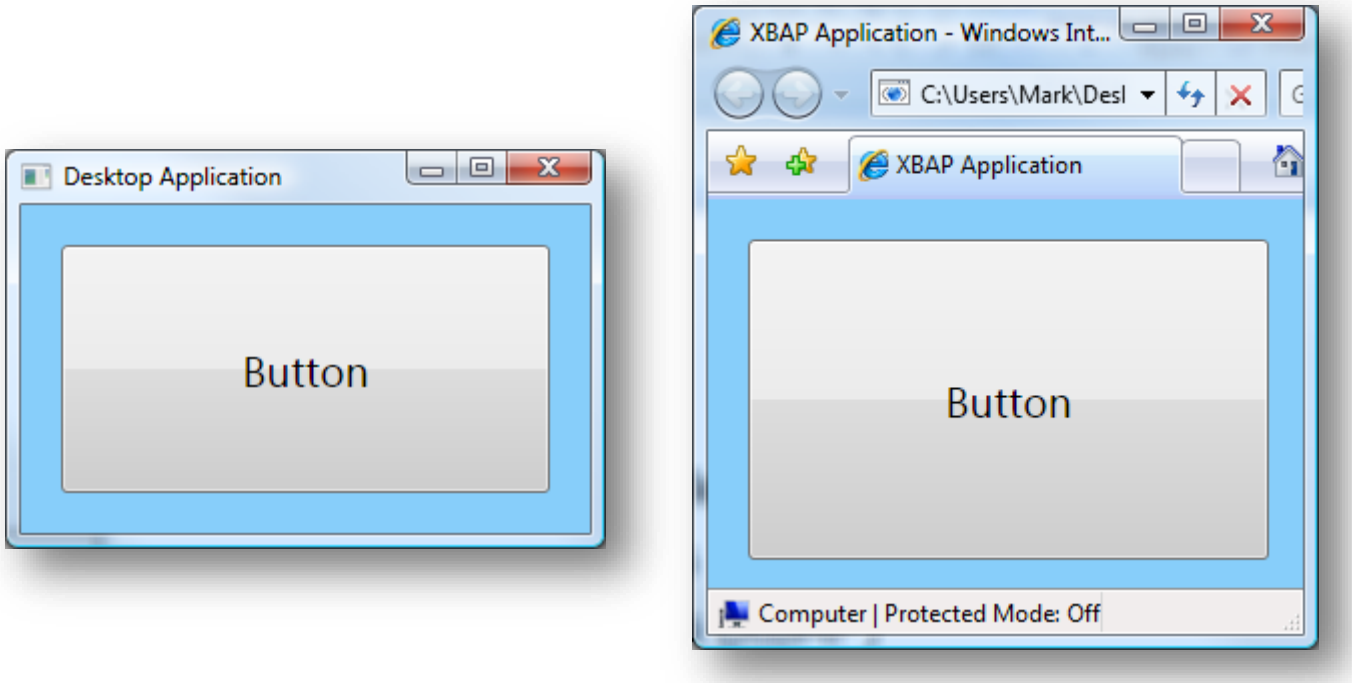
```
class MyButtonWindow : Window
{
   public MyButtonWindow()
   {
     Button btn = new Button { Content="Click M
     Ellipse el = new Ellipse { Fill = Brushes.

     StackPanel panel = new StackPanel();
     panel.Children.Add(btn); panel.Children.Add(el);
     this.Content= panel;
   }
}
```



Application Styles

- Two primary types of application styles available in WPF
  - desktop-based applications
  - browser-based applications (XBAP) – NOT Recommended anymore (Graduated to become Silverlight, also “deprecated”)



Summary

- Architecture was redesigned from the ground up
  - do not assume it works in the traditional Win32 fashion
- Flexible design provides for almost any style of application
  - learn a single technology
- Controls and Shapes provide simple building blocks for UI
  - anything not present is easily composed
- Layout is performed with Panels (more on this later)
  - goal is to provide flexibility + simplicity
  - enables automatic resize and UI scaling
- Can host WPF content directly in browser through XBAP
  - Not recommended anymore
  - Instead use Silverlight
  - Silverlight has it’s own limitations so probably use HTML 5 ;)