

Controls

Krzysztof Mossakowski Faculty of Mathematics and Information Science

http://www.mini.pw.edu.pl/~mossakow

Content Models

- There are 4 content models used by controls:
 - ContentControl contains a single item
 - used by: Button, ButtonBase, CheckBox, ComboBoxItem, ContentControl, Frame, GridViewColumnHeader, GroupItem, Label, ListBoxItem, ListViewItem, NavigationWindow, RadioButton, RepeatButton, ScrollViewer, StatusBarItem, ToggleButton, ToolTip, UserControl, Window
 - □ **HeaderedContentControl** contains a header and a single item
 - Expander, GroupBox, TabItem
 - ItemsControl contains a collection of items
 - ComboBox, ContextMenu, ListBox, ListView, Menu, StatusBar, TabControl, TreeView
 - HeaderedItemsControl contains a header and a collection of items
 - MenuItem, ToolBar, TreeViewItem

Krzysztof Mossakowski Faculty of Mathematics and Information Science

Content

- The **ContentControl** class represents a control with a single piece of content
- The ContentPresenter class is responsible for displaying the content of a ContentControl; it uses the following algorithm:
 - 1. If **Content** is of type **UIElement**, then add it to the display tree
 - 2. If **ContentTemplate** is set, use that to create a **UIElement** instance and add it to the display tree
 - 3. If **ContentTemplateSelector** is set, use that to find a template, use the template to create a **UIElement** instance, and add it to the display tree
 - 4. If the data type of **Content** has a data template associated with it, use that to create a **UIElement** instance
 - If the data type of **Content** has a **TypeConverter** instance associated with it that can convert to type **UIElement**, convert **Content** and add it to the display tree
 - 6. If the data type of **Content** has a **TypeConverter** instance associated with it that can convert to a string, wrap **Content** in **TextBlock** and add it to the display tree
- Finally, call **ToString** on **Content**, wrap it in **TextBlock**, and add it to the display tree

Faculty of Mathematics and Information Science

Content Example

```
<Window x:Class="WpfApplication1.Window2"</pre>
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   Title="Window2" Height="300" Width="300">
    <Grid>
        <ListBox Name="listBox1" VerticalAlignment="Stretch">
            <ListBoxItem>
                The first item
            </ListBoxItem>
            <ListBoxItem>
                <Rectangle Fill="Yellow" Height="20" Width="20"></Rectangle>
            </ListBoxItem>
            <ListBoxItem>
                <Image Source="Images/Fiona 67x77.gif"></Image>
            </ListBoxItem>
            <ListBoxItem>
                <StackPanel Orientation="Horizontal">
                    <TextBlock>The fourth item</TextBlock>
                    <Rectangle Fill="Yellow" Height="20" Width="20"></Rectangle>
                    <Image Source="Images/Fiona 67x77.gif"></Image>
                </StackPanel>
            </ListBoxItem>
        </ListBox>
    </Grid>
</Window>
```

Faculty of Mathematics and Information Science



Templates

- Templates are a factory for creating a new display tree
- There are 2 types of templates:
 - □ **ControlTemplate** creates a display tree for a control
 - **DataTemplate** creates a display tree for a piece of data
- The structure and appearance of a control can be modified by defining a new **ControlTemplate** for the control
 - □ If no own **ControlTemplate** is created for a control, the default template that matches the system theme will be used
 - There is no way to replace only part of the visual tree of a control a complete ControlTemplate must be provided
- A template's trigger sets properties or starts actions such as animation when a property value changes or when an event is raised

```
<Window x:Class="WpfApplication1.WindowWithButton"</pre>
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                                                                                  WindowWi... 💶 🔳
  Title="WindowWithButton" Height="80" Width="150">
  <Button>
                                                                                    This is a button
    <Button.Template>
      <ControlTemplate TargetType='{x:Type Button}'>
        <Border Name="MyBorder" CornerRadius='4' BorderThickness='1'>
          <Border.BorderBrush>
            <SolidColorBrush Color="Black"/>
          </Border.BorderBrush>
                                                                                  WindowWi...
          <Border.Background>
            <LinearGradientBrush EndPoint='0,1'>
                                                                                   This is a button
              <LinearGradientBrush.GradientStops>
                 <GradientStop Offset='0' Color='#FF777777' />
                                                                                  WindowWi... 🗖 🔳
                 <GradientStop Offset='1' Color='#FFFFFFF' />
              </LinearGradientBrush.GradientStops>
                                                                                    This is a button
            </LinearGradientBrush>
          </Border.Background>
          <ContentPresenter HorizontalAlignment='Center' VerticalAlignment='Center' />
        </Border>
        <ControlTemplate.Triggers>
          <Trigger Property="Button.IsPressed" Value="True">
            <Setter TargetName="MyBorder" Property="Border.Background"</pre>
             Value="{DynamicResource {x:Static SystemColors.ControlDarkBrushKey}}" />
          </Trigger>
        </ControlTemplate.Triggers>
      </ControlTemplate>
    </Button.Template>
    This is a button
  </Button>
</Window>
```

Template Binding

Template binding allows to add parameters to templates

Parameters allow to customize a template for a control by using properties

```
<ControlTemplate x:Key="MyButtonTemplate" TargetType='{x:Type Button}'>
  <Border Name="MyBorder" CornerRadius='4'</pre>
                                                                                WindowWi... 🗕 🔳
     BorderThickness='{TemplateBinding Property=BorderThickness}'
                                                                               This is a button
     BorderBrush='{TemplateBinding Property=BorderBrush}'
                                                                               This is another button
     Background='{TemplateBinding Property=Background}'>
    <ContentPresenter />
                                                                                WindowWi...
  </Border>
                                                                               This is a button
  <ControlTemplate.Triggers>
                                                                               This is another button
    <Trigger Property="Button.IsPressed" Value="True">
      <Setter TargetName="MyBorder" Property="Border.Background"</pre>
          Value="{DynamicResource {x:Static SystemColors.ControlDarkBrushKey}}" />
    </Trigger>
                                                                                WindowWi... 🗖
  </ControlTemplate.Triggers>
                                                                               This is a button
</ControlTemplate>
                                                                               This is another button.
<Button Template="{StaticResource MyButtonTemplate}" Background="Yellow" >
  This is a button
```

</Button>

```
<Button Template="{StaticResource MyButtonTemplate}" BorderBrush="Cyan">
This is another button
</Button>
```

Styles

```
<Window.Resources>
    <Style x:Key="NavyButton" TargetType="{x:Type Button}">
        <Setter Property="Background" Value="Navy"/>
        <Setter Property="Foreground" Value="White"/>
        <Setter Property="Margin" Value="2"/>
    </Style>
    <Style x:Key="ShadowedNavyButton"
           BasedOn="{StaticResource NavyButton}"
           TargetType="{x:Type Button}">
        <Setter Property="BitmapEffect">
             Setter.Value>
                 <DropShadowBitmapEffect Opacity ="0.5"/>
            </setter.Value>
                                                             StylesExampleWindow
                                                                                   - 🗆 ×
        </setter>
    </Style>
                                                               The first button
                                                                             The second button
</Window.Resources>
<Button Grid.Column="1"
                                                                             The fourth button
                                                               The third button
       Style="{StaticResource NavyButton}">
    The second button
</Button>
<Button Grid.Row="1" Margin="10"
       Style="{StaticResource ShadowedNavyButton}">
    The third button
</Button>
```

Buttons

- ButtonBase
 - Button
 - ToggleButton
 - CheckBox
 - RadioButton

<StackPanel Orientation='Horizontal'> <Button Margin='5' VerticalAlignment='Top'>Click</Button> <StackPanel Margin='5' VerticalAlignment='Top'> <CheckBox IsChecked='True'>Click</CheckBox> <CheckBox IsChecked='False'>Click</CheckBox> <CheckBox IsChecked='{x:Null}'>Click</CheckBox> </stackPanel> <StackPanel Margin='5' VerticalAlignment='Top'> Ctrl_Buttons <RadioButton>Click</RadioButton> Click O Click Click <RadioButton IsChecked='True'>Click</RadioButton> Click O Click <RadioButton>Click</RadioButton> Click O Click </stackPanel> </StackPanel>

Krzysztof Mossakowski Faculty of Mathematics and Information Science

Lists

ListBox and ComboBox

- As the source for a list, any type that implements IEnumerable can be used for the ItemsSource property
- In .NET 3.0 a new ObservableCollection<T> collection was provided
- □ The **ItemsPanel** property can provide a template for creating the layout panel that will be used to display the items in the list
 - ItemsPanel takes ItemsPanelTemplate, which requires that the template build a panel
 - It allows to use custom layout for a **ListBox** or **ComboBox**

<StackPanel>

<ComboBox>

<ComboBox.ItemsPanel>

<ItemsPanelTemplate>

<UniformGrid Columns='2'/>

</ItemsPanelTemplate>

</ComboBox.ItemsPanel>

<ComboBoxItem>First</ComboBoxItem>

<ComboBoxItem>Second</ComboBoxItem>

<ComboBoxItem>Third</ComboBoxItem>

<ComboBoxItem>Fourth</ComboBoxItem>

</ComboBox>

<Rectangle Height="100"></Rectangle> <ListBox>

<ListBox.ItemsPanel>

<ItemsPanelTemplate>

<UniformGrid Columns='2'/>

</ItemsPanelTemplate>

</ListBox.ItemsPanel>

<ListBoxItem>First</ListBoxItem>

<ListBoxItem>Second</ListBoxItem>

<ListBoxItem>Third</ListBoxItem>

<ListBoxItem>Fourth</ListBoxItem>

</ListBox>

<ListBox>

<ListBoxItem>First</ListBoxItem> <ListBoxItem>Second</ListBoxItem> <ListBoxItem>Third</ListBoxItem> </ListBox>

</StackPanel>

Ctrl_ListBox		
Fourth		•
First	Second	
Third	Fourth	
First	Second	
Third	Fourth	
First		
Second		
Third		

e 8 - 12

Lists cont.

- The ListView control derives from the ListBox class
 - It adds the ability to separate view properties from control properties
- To specify a view mode for the content of a ListView control, the View property must be set
 - One view mode that WPF provides is GridView, which displays a collection of data items in a table that has customizable columns



Krzysztof Mossakowski Faculty of Mathematics and Information Science

Lists cont.

- The TreeView control provides a way to display information in a hierarchical structure by using collapsible nodes
- The TreeView control contains a hierarchy of TreeViewItem controls
 - A TreeViewItem control is a
 HeaderedItemsControl that has a Header and an Items collection







Windows Programming

```
<Window x:Class="WpfApplication1.Ctrl TemplateList"</pre>
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:sys="clr-namespace:System;assembly=mscorlib"
   Title="Ctrl TemplateList" Height="300" Width="300">
  <Grid>
    <ListBox>
      <ListBox.ItemContainerStyle>
        <Style TargetType='{x:Type ListBoxItem}'>
          <Setter Property='Template'>
            Setter.Value>
              <ControlTemplate TargetType='{x:Type ListBoxItem}'>
                <RadioButton
                 IsChecked='{Binding
                 Path=IsSelected,RelativeSource={RelativeSource TemplatedParent}}'
                 Content='{TemplateBinding Property=Content}' />
              </ControlTemplate>
            </setter.Value>
          </setter>
        </Style>
      </ListBox.ItemContainerStyle>
      <sys:String>First</sys:String>
      <sys:String>Second</sys:String>
      <sys:String>Third</sys:String>
                                                            Ctrl_TemplateList
                                                                                  First
      <sys:String>Fourth</sys:String>
                                                             Second
    </ListBox>
                                                             Third
                                                            Fourth
  </Grid>
</Window>
```

Menus and Toolbars

- A menu is logically nothing more than a **TreeView** control with a very special template
 - TreeViewItem and MenuItem derive from the same base type: HeaderedItemsControl
- Both a menu and a toolbar provide effectively the same functionality: the ability to execute one or more commands
 - The primary differences between the two are visual space and interaction model
 - Menus and toolbars are typically paired with commands

Menus

- Menus consist of a series of MenuItem controls hosted within either Menu or ContextMenu
- In WPF, a menu is no longer relegated to the top of the window

```
<Menu DockPanel.Dock="Bottom">
  <MenuItem Header=' File'>
    <MenuItem Header=' New'/>
    <MenuItem Header=' Open' InputGestureText="Ctrl+0" />
    <Separator/>
    <MenuItem Header='E xit' Click="ExitMenuItem Click">
      <MenuItem.Icon>
        <Image Source="Images/Fiona 67x77.gif" Width="16" Height="16"></Image>
      </MenuItem.Icon>
    </MenuItem>
                                                      Ctrl Menu
                                                                            </MenuItem>
  <MenuItem Header=' Edit'>
    <MenuItem Header=' Cut' />
                                                     File Edit
    <MenuItem Header='C opy' />
                                                       New
    <MenuItem Header=' Paste' />
                                                       Open Ctrl+O
  </MenuItem>
                                                       Exit
</Menu>
```

Toolbars

- Toolbars provide only one level of nesting
- There is a special host type for toolbars ToolBarTray
- Each toolbar has a set of items and a header
 - The content model allows to add anything as an item in the toolbar

 Image: Prev Next Image: http://www.google.com



Containers

- TabControl provides the traditional tab-style UI
 - □ It derives from **Selector** (and therefore **ItemsControl**)
- Expander offers the Windows XP-style expansion functionality known from the file explorer
- GroupBox provides a simple visual containment for separating parts of the UI
- TabItem, Expander, and GroupBox derive from HeaderedContentControl
 - The HeaderedContentControl can contain a single header and single content model
 - It can hold multiple items because a single layout container, like StackPanel or Grid, can be the root of that content

Containers Example

```
<TabControl>
  <Tabltem Header='Tab 1'>
    <StackPanel>
      <Expander Header='Expander 1' IsExpanded='True'>
        <GroupBox Header='GroupBox 1'>
          <StackPanel Orientation="Horizontal">
             <Label>Something 1</Label>
             <Image Source="Images/Fiona 67x77.gif"></Image>
          </stackPanel>
        </GroupBox>
      </Expander>
      <Expander Header='Expander 2'>
                                                         Ctrl_Containers
        <StackPanel>
           <GroupBox Header='GroupBox 2'>
                                                              Tab 2
                                                         Tab 1
             <Label>Something 2</Label>

    Expander 1

          </GroupBox>
                                                         GroupBox 1
          <GroupBox Header="GroupBox 3">
                                                         Something 1
             <Label>Something 3</Label>
          </GroupBox>
        </StackPanel>

    Expander 2

      </Expander>
    </StackPanel>
  </TabItem>
  <Tabltem Header='Tab 2' />
</TabControl>
```



Editors

- PasswordBox provides the all-too-familiar function of a text box, except that it replaces what the user types with a series of dots or asterisks
 - The password is secured, there is no way to disclose it by browsing the text tree
- TextBox supports only plain text
- RichTextBox enables to display or edit flow content including paragraphs, images, tables, and more
 - RichTextBox hosts a FlowDocument object which in turn contains the editable content
- InkCanvas is an element that can be used to receive and display ink input

Editors – Text Data

- There are two main concepts: block elements and inline elements
 - Block elements occupy a rectangle of space; they start on a new line and are contiguous
 - examples of block elements are Paragraph or Table
 - Inline elements can span lines
 - examples of inline elements are Span, Run,

Krzysztof Mossakowski Faculty of Mathematics and Information Science



Windows Programming

Documents Examples

	Gizmos	Thingamajigs	Doohickies
Blue	1	2	3
Red	1	2	3
Green	1	2	3
Totals	3	6	9

A UIElement element may be embedded directly in flow content by enclosing it Lorem ipsum dolor sit amet, consectetuer in a BlockUIContainer element. Click me! The BlockUIContainer element may host no more than one top-level UIElement.

However, other UIElements may be nested within the UIElement contained by an BlockUIContainer element. For example, a StackPanel can be used to host multiple UIElement elements within a BlockUIContainer element.

Choose a value:

a	~
Choose a value:	
Dx	
Oy D	
Enter a value:	
A text editor embedded in flow content.	



more work into the day can mean working longer hours, but it also requir become more efficient in what we do and look for ways to better manage

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nulla ligula tortor, dapibus et, facilisis a, aliquet et, diam. Cras convallis. Fusce at urna. Quisque interdum, turpis sed dictum fringilla, magna arcu gravida libero, elementum tincidunt dolor mauris sed erat. Curabitur egestas aliquet nibh. Etiam quis sem in lorem auctor consequat. Suspendisse vitae lorem. Proin eleifend elementum ligula. Donec mattis consectetuer erat. Donec fermentum consectetuer neque. Suspendisse tempus faucibus magna. Pellentesque sodales velit eget nibh. Phasellus velit magna, malesuada vitae, rhoncus eget, dignissim ut, metus. Praesent pulvinar suscipit diam.

adipiscing elit. Nulla ligula tortor, dapibus et, facilisis a, aliquet et, diam, Cras convallis, Fusce at urna. Quisque interdum, turpis sed dictum fringilla, magna arcu gravida libero, elementum tincidunt dolor mauris sed erat. Curabitur egestas aliquet nibh. Etiam quis sem in lorem auctor consequat. Suspendisse vitae lorem. Proin eleifend elementum ligula. Donec mattis consectetuer erat. Donec fermentum consectetuer neque. Suspendisse tempus faucibus magna. Pellentesque sodales velit eget nibh. Phasellus velit magna, malesuada vitae, rhoncus eget, dignissim ut, metus. Praesent pulvinar suscipit diam.



Morbi ut tellus at tellus semper consectetuer. Nullam fringilla nonummy justo. Proin rutrum, purus id adipiscing malesuada, enim velit accumsan nisi, pellentesque rhoncus nibh nisi ut magna. Nunc massa nulla, volutpat sit amet, pulvinar ac, scelerisque ac, magna. Nulla facilisi. Integer pharetra felis vitae risus. Nunc aliquet lacus pretium urna varius hendrerit. Duis odio nisl, venenatis at, sollicitudin eu, viverra sed, nibh, Nullam consequat, Duis felis felis, rutrum viverra, auctor eget, iaculis ut, mi. Integer sagittis pharetra ipsum. Donec lacinia scelerisque nisl. Nullam aliquet. In et sapien. Fusce blandit odio et mi.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nulla ligula tortor, dapibus et, facilisis a, aliquet et, diam. Cras convallis. Fusce at urna. Ouisque interdum, turpis sed dictum fringilla, magna arcu gravida libero, elementum tincidunt dolor mauris sed erat. Curabitur egestas aliquet nibh. Etiam quis sem in lorem auctor consequat. Suspendisse vitae lorem. Proin eleifend elementum ligula. Donec mattis consectetuer erat. Donec fermentum consectetuer neque. Suspendisse tempus faucibus magna. Pellentesque sodales velit eget nibh. Phasellus velit magna, malesuada vitae, rhoncus eget, dignissim ut, metus. Praesent pulvinar suscipit diam.

<1 of 4 🕽 🗐 🔯 🖶 —등 🗄

Morbi ut tellus at tellus semper consectetuer. Nullam fringilla nonummy justo. Proin rutrum, purus id adipiscing malesuada, enim velit accumsan nisi, pellentesque rhoncus nibh nisi ut magna. Nunc massa nulla, volutpat sit amet, pulvinar ac, scelerisque ac, magna. Nulla facilisi. Integer pharetra felis vitae risus. Nunc aliquet lacus pretium urna varius hendrerit. Duis odio nisl, venenatis at, sollicitudin eu, viverra sed, nibh. Nullam consequat. Duis felis felis, rutrum viverra, auctor eget, iaculis ut, mi. Integer sagittis pharetra ipsum. Donec lacinia scelerisque nisl. Nullam aliquet. In et sapien. Fusce blandit odio et mi.

<1 of 2 ▶

pellentesque diam.

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Suspendisse laoreet condimentum purus. Etiam vel enim. Suspendisse at dolor. Pellentesque gravida. Aenean convallis. Vivamus diam. Aenean lorem libero, suscipit vel, tempor eu, fermentum aliquam, metus. Quisque volutpat urna at augue. Nam placerat. In viverra congue tellus. Proin auctor. Fusce nisl lorem, dapibus vitae, porta sed, malesuada a, lacus. Phasellus mollis elementum enim. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, Etiam vitae urna vel velit volutpat tempor. Ouisque ac dolor. Suspendisse potenti. Nunc nec tortor. Curabitur pharetra

4 1 of 1 ≥

- - 11 11

Windows Progra

InkCanva

Windows Programming	Ctrl_InkCanvas2				
InkCanvas Example	Select Ink Select EraseByPoint EraseByStroke				
anvas>					
<canvas.resources></canvas.resources>					
Define an array containing the InkEditin</th <th>ngMode Values></th>	ngMode Values>				
<pre><x:array x:key="MyEditingModes" x:type="{x:Type InkCanvasEditingMode}"></x:array></pre>					
<x:static member="InkCanvasEditingMode.Ink"></x:static>					
<pre><x:static member="InkCanvasEditingMode.Select"></x:static></pre>					
<pre><x:static member="InkCanvasEditingMode.Era</pre></th><th>aseByPoint"></x:static></pre>					
<pre><x:static member="InkCanvasEditingMode.Era</pre></th><th>aseByStroke"></x:static></pre>					
<stackpanel></stackpanel>					
<combobox <="" name="cmbEditingMode" th=""><td></td></combobox>					
ItemsSource="{StaticResource MyEditin	ngModes}" />				
<inkcanvas< th=""><th></th></inkcanvas<>					
EditingMode="{Binding ElementName=cmb	<pre>DEditingMode, Path=SelectedItem}"></pre>				

</Canvas>

<Canvas>

Document Viewers

- RichTextBox editable scrolling view
- FlowDocumentScrollViewer continuous scrolling view
- FlowDocumentPageViewer paginated view, one page at a time
- FlowDocumentReader single-page view, multiple-page view, or scrolling view
 - The optimal control for the user, where he can choose an appropriate view

Neptune (planet), major planet in the solar system, eighth planet from the Sun tune maintains

Neptune has 72 times 4,490 million Earth's volume... the Sun. Nep-Uranus and for elliptical path

and fourth largest in diameter. Nepan almost constant distance, about km (about 2,790 million mi), from tune revolves outside the orbit of most of its orbit moves inside the of the outermost planet Pluto (see

Solar System). Every 248 years, Pluto's elliptical orbit brings the planet inside Neptune's nearly circular orbit for about 20 years, temporarily making Neptune the farthest planet from the Sun. The last time Pluto's orbit brought it inside Neptune's orbit was in 1979. In 1999 Pluto's orbit carried it back outside Neptune's orbit.

Neptune Stats		Astro Nepti
Mean Distance from S	un 4,504,000,000 km	round
Mean Diameter	49,532 km	of w
Approximate Mass	1.0247e26 kg	rocky
Information from the Encarta	web site.	ocear

nomers believe ne has an inner core that is sured by a vast ocean ter mixed with material. From nner core, this extends upward until it meets a gaseous atmosphere of hydrogen, helium, and trace

amounts of methane. Neptune has four rings and 11 known moons. Even though Neptune's volume is 72 times Earth's volume, its mass is only 17.15 times Earth's mass. Because of its size, scientists classify Neptune-along with Jupiter, Saturn, and Uranus-as one of the giant or Jovian planets (so-called because they resemble Jupiter).



Controls

Controls

FlowDocumentScrollViewer

Content

Area-

FlowDocumentReader

Neptune (planet), major planet in the solar system, eighth planet from the Sun and fourth largest

constant dismillion mi), orbit of Uranus elliptical path

Neptune has 72 times Earth's volume...

in diameter. Neptune maintains an almost tance, about 4,490 million km (about 2,790 from the Sun. Neptune revolves outside the and for most of its orbit moves inside the of the outermost planet Pluto (see Solar 248 years, Pluto's elliptical orbit brings the

System). Every planet inside Neptune's nearly circular orbit for about 20 years, temporarily making Neptune the farthest planet from the Sun. The last time Pluto's orbit brought it inside Neptune's orbit was in 1979. In 1999 Pluto's orbit carried it back outside Neptune's orbit.

Neptune Stats

Mean Distance from Sun	4,504,000,000 km
Mean Diameter	49,532 km
Approximate Mass	1.0247e26 kg
Information from the Encarta web	site.

Astronomers believe Neptune has an inner rocky core that is surrounded by a vast ocean of water mixed with rocky material. From the inner core, this ocean extends upward until it meets a gaseous atmosphere of hydrogen, helium, and trace amounts of methane. Neptune has four rings and 11 known moons. Even though Neptune's volume is 72

times Earth's volume, its mass is only 17.15 times Earth's mass. Because of its size, scientists classify Neptune—along with Jupiter, Saturn, and Uranus—as one of the giant or Jovian planets (so-called because they resemble Jupiter).

Mathematical theories of asof Neptune. To account for Neptune has an orbital planet Uranus, British astroperiod of ~20 years... and French astronomer independently calculated the

tronomy led to the discovery wobbles in the orbit of the nomer John Couch Adams Urbain Jean Joseph Leverrier existence and position of a

new planet in 1845 and 1846, respectively. They theorized that the gravitational attraction of this planet for Uranus was causing the wobbles in Uranus's orbit. Using information from Leverrier, German astronomer Johann Gottfried Galle first observed the planet in 1846.



http://www.mini.pw.edu.pl/~mossakow

ToolTip

- Usually, tool tip functionality is accessed using the **ToolTip** property on elements
- To adjust more tool tip properties ToolTipService can be used

A custom template for the **ToolTip** control can be applied

```
<Window x:Class="WpfApplication1.Ctrl ToolTip"</pre>
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   Title="Ctrl ToolTip" Height="100" Width="300">
  <Window.Resources>
    <Style TargetType="{x:Type ToolTip}">
      [...]
    </Style>
  </Window.Resources>
                                                                 The 1st button
  <StackPanel>
                                                                Cti /IooIIIp
                                                                                      <Button ToolTip="The 1st button">Button 1</Button>
                                                                          Button 1
    <Button ToolTip="The 2nd button">Button 2</Button>
                                                                          Button 2
    <Button ToolTip="The 3rd button">Button 3</Button>
                                                                          Button 3
  </stackPanel>
                                                           http://thewpfblog.com/?p=61
</Window>
```

Thumb

Thumb provides a region that can be moved

It provides the ability to get events when a user drags this region (the thumb) around



Border

Border is a rectangle that can contain a child

Most render elements (Rectangle, Ellipse, etc.) don't support children, so the Border can be used as their parent

```
<Canvas>
  <Border Canvas.Left='15' Canvas.Top='15' BorderThickness='3'</pre>
          CornerRadius='0' BorderBrush='Black' Padding='5'>
                                                                         Ctrl Border
                                                                                _ 🗆 ×
    <TextBlock>Hello 1</TextBlock>
  </Border>
                                                                                Hello 2
                                                                          Hello 1
  <Border Canvas.Left='85' Canvas.Top='15' BorderThickness='3'</pre>
                                                                         Hello 3 Hello 4
          CornerRadius='3' BorderBrush='Black' Padding='5'>
    <TextBlock>Hello 2</TextBlock>
  </Border>
  <Border Canvas.Left='15' Canvas.Top='50' BorderThickness='10,1,10,1'</pre>
          CornerRadius='10' BorderBrush='Black' Padding='5'>
    <TextBlock>Hello 3</TextBlock>
  </Border>
  <Border Canvas.Left='85' Canvas.Top='50' BorderThickness='4,1,4,1'</pre>
          CornerRadius='0,15,0,15' BorderBrush='Black' Padding='5'>
    <TextBlock>Hello 4</TextBlock>
  </Border>
</Canvas>
```

Popup

- The **Popup** control provides a way to display content in a separate window that floats over the current application window relative to a designated element or screen coordinate
- The following controls implement the Popup control for specific uses: ToolTip, ContextMenu, ComboBox,

```
<Button HorizontalAlignment="Left" Click="DisplayPopup"
                                                                       Ctrl Popup
                                                                                    - - ×
     Width="150" Margin="20,10,0,0">
  <StackPanel>
                                                                         Display Your Popup Text
    <TextBlock>Display Your Popup Text</TextBlock>
    <Popup Name="myPopup">
                                                                   Foreground="Blue">
      <TextBlock Name="myPopupText" Background="LightBlue"
      Popup Text
                                                                       Ctrl_Popup
                                                                                    - 🗆 ×
      </TextBlock>
    </Popup>
                                                                         Display Your Popup Text
  </stackPanel>
                                                                          Popup Text
</Button>
private void DisplayPopup(object sender, RoutedEventArgs e) {
    myPopup.IsOpen = !myPopup.IsOpen;
```

ScrollViewer

- The ScrollViewer control provides a convenient way to enable scrolling of content
 - It encapsulates horizontal and vertical ScrollBar elements and a content container
- A ScrollViewer can only have one child, typically a Panel element that can host a Children collection of UIElements

```
<Window x:Class="WpfApplication1.Ctrl ScrollViewer"</pre>
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   Title="Ctrl ScrollViewer" Height="300" Width="300">
                                                                 Ctrl ScrollViewer
                                                                                        <Window.Resources>
    [...]
  </Window.Resources>
  <ScrollViewer HorizontalScrollBarVisibility="Visible">
                                                                                  My second button
    CockPanel
      <Button Width="400" Height="50">My button</Button>
      <Button>My second button</Button>
    </DockPanel>
  </ScrollViewer>
                                                                http://sachabarber.net/?p=122
</Window>
                                                    http://www.mini.pw.edu.pl/~mossakow
 Faculty of Mathematics and Information Science
```

}

Viewbox

Viewbox takes a single child and stretches it to fit by applying rendering transforms (stretching the content)

```
<StackPanel Orientation="Vertical">
  <StackPanel Margin="0,0,0,10" HorizontalAlignment="Center"</pre>
              Orientation="Horizontal" DockPanel.Dock="Top">
    <Button Name="btn1" Click="stretchNone">Stretch="None"</Button>
    <Button Name="btn2" Click="stretchFill">Stretch="Fill"</Button>
    <Button Name="btn3" Click="stretchUni">Stretch="Uniform"</Button>
    <Button Name="btn4" Click="stretchUniFill">Stretch="UniformToFill"</Button>
  </stackPanel>
  <StackPanel Margin="0,0,0,10" HorizontalAlignment="Center"</pre>
              Orientation="Horizontal" DockPanel.Dock="Top">
    <Button Name="btn5" Click="sdUpOnly">StretchDirection="UpOnly"</Button>
    <Button Name="btn6" Click="sdDownOnly">StretchDirection="DownOnly"</Button>
    <Button Name="btn7" Click="sdBoth">StretchDirection="Both"</Button>
  </stackPanel>
  <TextBlock DockPanel.Dock="Top" Name="txt1" />
  <TextBlock DockPanel.Dock="Top" Name="txt2" />
  <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
    <Viewbox MaxWidth="500" MaxHeight="500" Name="vb1">
      <Image Source="Images/Fiona 67x77.gif"/>
    </Viewbox>
                      public void stretchNone(object sender, RoutedEventArgs e) {
  </StackPanel>
                          vb1.Stretch = System.Windows.Media.Stretch.None;
</stackPanel>
                          txt1.Text = "Stretch is now set to None.";
```



Layout

Krzysztof Mossakowski Faculty of Mathematics and Information Science

http://www.mini.pw.edu.pl/~mossakow

FrameworkElement

- FrameworkElement is the connecting point between WPF framework-level element classes and the WPF core-level set of UIElement presentation services
- FrameworkElement extends UIElement and adds the following capabilities:
 - □ Layout system definition
 - □ The logical tree
 - Object lifetime events
 - □ Support for data binding and dynamic resource references
 - □ Styles
 - □ More animation support

FrameworkElement Properties

- MinWidth, MaxWidth, MinHeight, MaxHeight
- Width, Height usually not set directly, to allow to resize automatically
- ActualHeight, ActualWidth report the final size of the element
- Margin lets a child put a buffer of space around itself inside of the slot, but outside of the content of the control
 Negative values can also be used
- HorizontalAlignment, VerticalAlignment determine how the child occupies the remaining space in the slot

Clipping

- Clipping is not the default
 - □ WPF uses a composition engine that employs the painter's algorithm (painting from back to front)
 - Clipping is expensive for each pixel there is a need to check if it should be visible

Layout01		
	Cente	r
Left		_
		Right
	Stretc	h
Тор		
Control I	Charles	
Center	Stretch	
Bottom		

```
<UniformGrid Columns="1">
```

<StackPanel Orientation='Vertical' Background="AntiqueWhite">

- <Button HorizontalAlignment='Center'>Center</Button>
 - <Button HorizontalAlignment='Left'>Left</Button>
 - <Button HorizontalAlignment='Right'>Right</Button>
 - <Button HorizontalAlignment='Stretch'>Stretch</Button>

```
</StackPanel>
```

- <StackPanel Orientation='Horizontal' Background="Chocolate">
 - <Button VerticalAlignment='Center'>Center</Button>
 - <Button VerticalAlignment='Top'>Top</Button>
 - <Button VerticalAlignment='Bottom' Margin="-20,0,0,0">Bottom</Button>
 - <Button VerticalAlignment='Stretch' Margin="0,-20,0,-20">Stretch</Button>
- </StackPanel>
- </UniformGrid>

Transforms

- To adjust the final position of a control, two properties are available for making an arbitrary transformation:
 - RenderTransform is applied immediately before rendering, (therefore affecting only rendering)
 - □ LayoutTransform is applied before layout

```
<TabControl>
                                                              Transform
                                                                                     - 🗆 ×
  <Tabltem Header="LayoutTransform">
                                                                         RenderTransform
                                                              LayoutTransform
    <StackPanel Orientation="Horizontal">
      <Button Width="100">15
         <Button.LayoutTransform>
           <RotateTransform Angle="15"/>
         </Button.LayoutTransform>
      </Button>
                                                                                     - 🗆 ×
                                                              Transform
      <Button Width="100">45
                                                                         RenderTransform
                                                              LayoutTransform
         <Button.LayoutTransform>
           <RotateTransform Angle="45"/>
         </Button.LayoutTransform>
      </Button>
                                                                      80
    </stackPanel>
  </TabItem>
  <!-- Analogously, but using the RenderTransform -->
</TabControl>
                                                                              nossakow
```

Z-Index

By default, all controls have a z-index of 0

```
Zindex
                                                                                Default
<StackPanel>
                                                                   Putton One If Putton Two IO1
                                                                   Button Three [( Button Four [0]
  <Label>Default</Label>
  <WrapPanel>
    <Button Margin='-5'>Button One [0]</Button>
    <Button Margin='-5'>Button Two [0]</Button>
                                                                   Modified ZIndex
                                                                  Button One [2] utton Two [0]
    <Button Margin='-5'>Button Three [0]</Button>
                                                                   Sutton Three 11 utton Four [0]
    <Button Margin='-5'>Button Four [0]</Button>
  </WrapPanel>
  <Rectangle Height="50"/>
  <Label>Modified ZIndex</Label>
  <WrapPanel>
    <Button Panel.ZIndex='2' Margin='-5'>Button One [2]
    <Button Margin='-5'>Button Two [0]</Button>
    <Button Panel.ZIndex='1' Margin='-5'>Button Three [1]/Button>
    <Button Margin='-5'>Button Four [0]</Button>
  </WrapPanel>
</StackPanel>
```

Canvas

- Canvas is the simplest layout included in WPF
- It offers four properties: Top, Left, Right, and Bottom
 - Only two properties can be used: one horizontal coordinate and one vertical coordinate
 - It lets position its child elements at any offset from one corner of the panel

```
<Canvas Width='200' Height='100' Background="AntiqueWhite" >

<Button Canvas.Right='10' Canvas.Top='10'> Top, Right </Button>

<Button Canvas.Left='10' Canvas.Top='10'> Bottom, Right </Button>

<Button Canvas.Left='10' Canvas.Bottom='10'> Bottom, Left </Button>

<Button Canvas.Left='30' Canvas.Bottom='10'> Bottom, Left </Button>

<Button Canvas.Left='30' Canvas.Bottom='30' Canvas.Right='30' Canvas.Top='30'>

All Four

</Canvas>
```

Bottom, Left

Bottom, Right

StackPanel

- StackPanel stacks its child elements in a column (by default) or in a row (when the Orientation property is set to Horizontal)
 - The slot for each child in StackPanel is given the entire width or height of the control (depending on its orientation)

```
<Border BorderBrush='Black' BorderThickness='2'</pre>
       HorizontalAlignment='Center' VerticalAlignment='Center'>
  <StackPanel Orientation='Vertical'>
    <StackPanel Margin='10' Background='Green' Orientation='Horizontal'>
      <Button Margin='4'>One</Button>
                                                                            StackPanel
                                                                                       <Button Margin='4'>Two</Button>
      <Button Margin='4'>Three</Button>
                                                                              One Two Three
    </stackPanel>
    <StackPanel Margin='5' Background='Blue' Orientation='Vertical'>
                                                                                  One
      <Button Margin='4'>One</Button>
      <Button Margin='4'>Two</Button>
                                                                                  Two
      <Button Margin='4'>Three</Button>
                                                                                  Three
    </stackPanel>
  </StackPanel>
</Border>
Krzysztor mossakows
                                                   http://www.mini.pw.edu.pl/~mossakow
Faculty of Mathematics and Information Science
```

DockPanel

- DockPanel allows mixing of stacking from different edges within the same layout container
- The LastChildFill property allows the last child to consume all the remaining space
- The **Dock** property allows to specify the edge to which a control is docked

CkPanel> CButton DockPanel.Dock='Top'>Menu Area CButton DockPanel.Dock='Top'>Toolbar Area CButton DockPanel.Dock='Left'>Folders CButton>Content DockPanel>		Dock	Menu Area
		Folders	Toolbar Area Content
<dockpanel></dockpanel>	ו	Dock	Panel
<pre><button dockpanel.dock="Top">Menu Area</button> <button dockpanel.dock="Left">Folders</button> <button dockpanel.dock="Top">Toolbar Area</button> <button>Content</button> </pre>		Folders	Toolbar Area Content

Lecture 8 - 43

WrapPanel

- WrapPanel is a stack panel with wrapping support
 - It uses the available space and fits elements to it; when it runs out of room, it wraps to the next line
- By default, WrapPanel simply sizes all the children to fit their content
 - The width and height of the children by using the ItemWidth and ItemHeight properties



UniformGrid

- UniformGrid provides a very basic grid layout: each cell is the same size, and the locations of the items are determined simply by their order in the children collection
- The number of columns and optionally the number of rows can be set using the **Columns** and **Rows** properties resp.

```
<StackPanel>

<UniformGrid Columns="2">

<Button>1</Button>

<Button>2</Button>

<Button>3</Button>

<Button>4</Button>

<Button>5</Button>

</UniformGrid>

<Rectangle Height="25"/>

<UniformGrid Columns="2" Rows="2"> [...] </UniformGrid>

<Rectangle Height="25" Opacity="0.5" Fill="Cyan" />

<UniformGrid Rows="2"> [...] </UniformGrid>

</StackPanel>
```

UniformGr	id		
1			2
3			4
5			
1			2
3			4
5			
1	1	2	3
4	5		
			-

Lecture 8 - 45

Grid

Krz

- Grid is the most power, flexible, and complex of the UI layouts
- The simplest use of Grid is to set the RowDefinitions and ColumnDefinitions properties, add some children, and use the Grid.Row and Grid.Column attached properties to specify which child goes in which slot



Grid – Percentage Sizing Example



Krzysztof Mossakowski Faculty of Mathematics and Information Science Windows Programming

Grid – Shared Size Example

<Grid> [...] [1,0]<Grid.ColumnDefinitions> <ColumnDefinition/> <ColumnDefinition/> Grid01 </Grid.ColumnDefinitions> Default Auto [...] </Grid> [0,0] [0,1][...] <Grid> [1,0] [1,1] (longer than others) [...] <Grid.ColumnDefinitions> <ColumnDefinition Width="Auto"/> <ColumnDefinition Width="Auto"/> </Grid.ColumnDefinitions> [...] </Grid> [...] <Grid Grid.IsSharedSizeScope="True"> [...] <Grid.ColumnDefinitions> <ColumnDefinition Width="Auto" SharedSizeGroup="ssq01"/> <ColumnDefinition Width="Auto" SharedSizeGroup="ssq01"/> </Grid.ColumnDefinitions> [...] </Grid>

Auto Shared [0,0] [0,1] [1,1] (longer than others) - 🗆 × Shared

Grid01

Default

E Grid01		
Default Auto	Shared	
[0,0]	[0,1]	
[1,0]	[1,1] (longer than others)	

Locturo 8 - 47

lu.pl/~mossakow

GridSplitter

 GridSplitter is a normal control (supporting templates, etc.) that allows the user to edit the layout of a row or column at runtime

