Sizing, Positioning and Tranforming Elements

WPF Layout

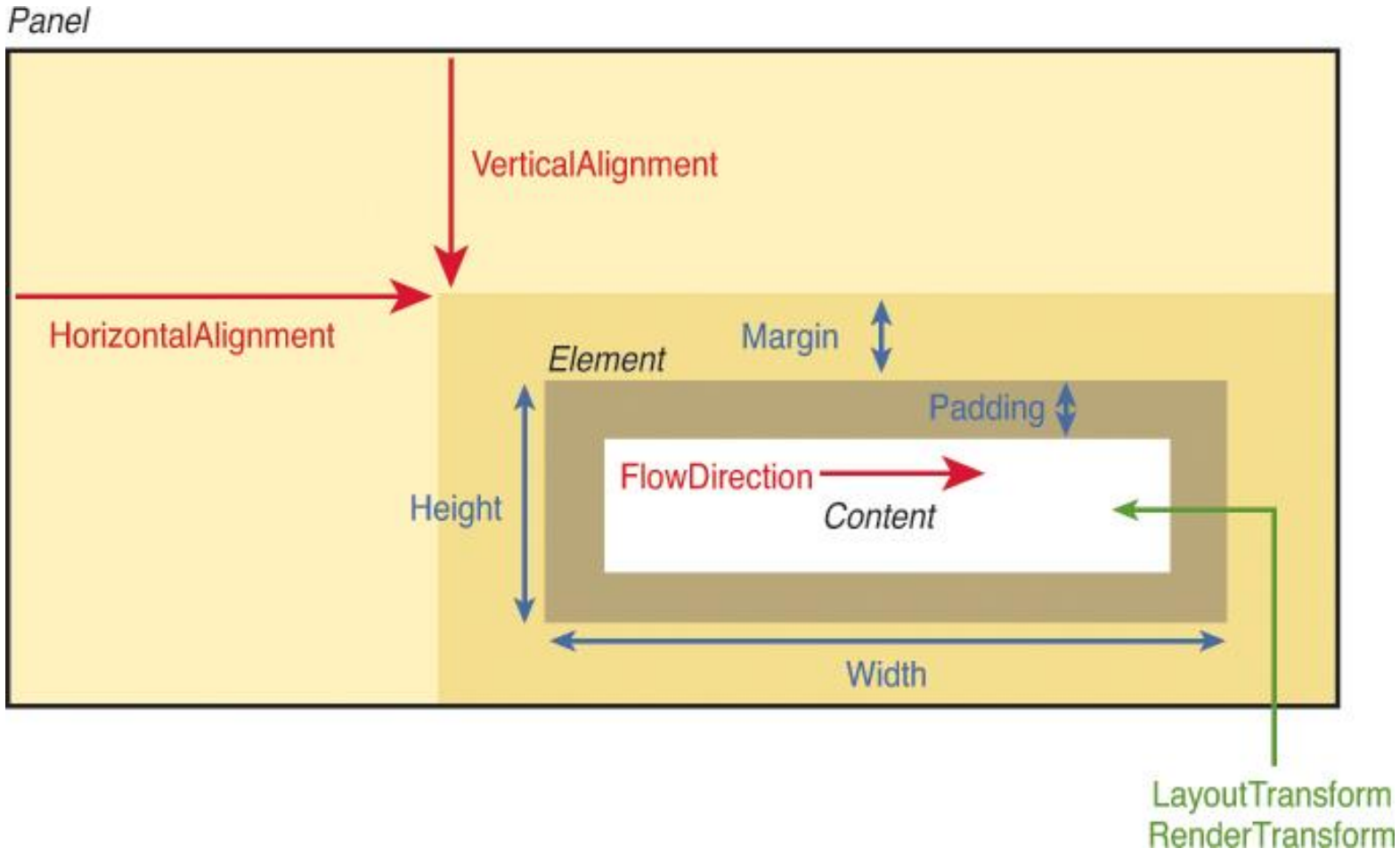
- Layout – the sizing and positioning of controls (and other elements) is called *layout*. WPF contains a lot of infrastructure to provide a feature-rich layout system.

- Layout in WPF boils down to interactions between parent elements and their child elements.

- Parent elements that support the arrangement of multiple children are known as *panels*, and they derive from the abstract `System.Windows.Controls.Panel` class.

- All the elements involved in the layout process (both parents and children) derive from `System.Windows.UIElement`.

Main Child Layout Properties



Controlling Size – Height and Width

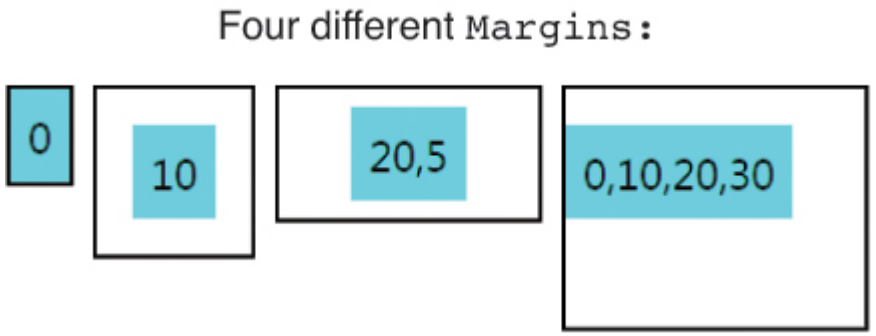
- All FrameworkElements have simple *Height* and *Width* properties (type *double*), and *MinHeight*, *MaxHeight*, *MinWidth*, and *MaxWidth* properties that can be used to specify a range of acceptable values. When used together, Height and Width take precedence as long as they are in the range from Min to Max.

- ****Avoid setting explicit sizes!** Giving controls explicit sizes, especially ContentControls such as Button and Label, opens up the risk of cutting off text when users change system font settings Therefore, you should avoid setting explicit sizes unless absolutely necessary.

- The default value of MinHeight and MinWidth is 0, and the default value of MaxHeight and MaxWidth is Double.PositiveInfinity.

Controlling Size – Margin

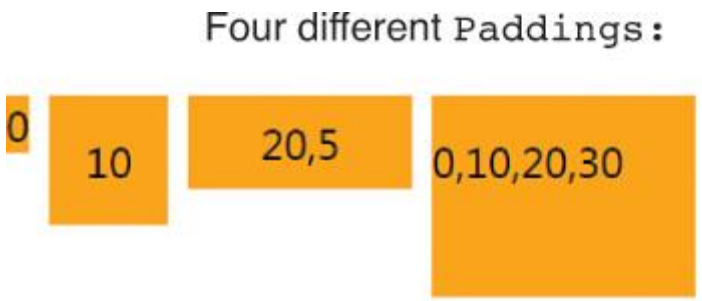
- Margin controls how much extra space gets placed around the outside edges of the element




```
<Border BorderBrush="Black" BorderThickness="1">
  <!-- No margin: -->
  <Label Background="Aqua">0</Label>
</Border>

<Border BorderBrush="Black" BorderThickness="1">
  <!-- 1 value: The same margin on all four sides: -->
  <Label Margin="10" Background="Aqua">10</Label>
</Border>

<Border BorderBrush="Black" BorderThickness="1">
  <!-- 2 values: Left & Right get the 1st value,
                 Top & Bottom get the 2nd value: -->
  <Label Margin="20,5" Background="Aqua">20,5</Label>
</Border>

<Border BorderBrush="Black" BorderThickness="1">
  <!-- 4 values: Left,Top,Right,Bottom: -->
  <Label Margin="0,10,20,30" Background="Aqua">0,10,20,30</Label>
</Border>
```

Controlling Size – Padding

- Padding controls how much extra space gets placed around the inside edges of the element.

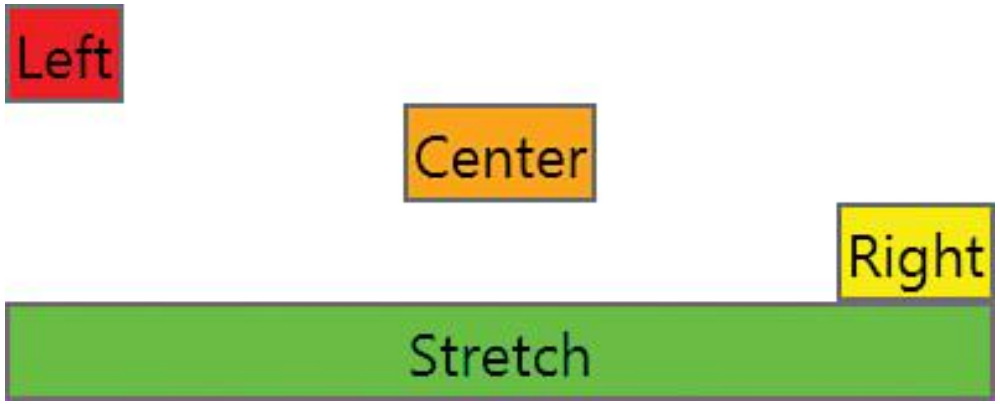



```
<!-- PADDING: -->

<!-- 1 value: The same padding on all four sides: -->
<Label Padding="0" Background="Orange">0</Label>
<Label Padding="10" Background="Orange">10</Label>

<!-- 2 values: Left & Right get the 1st value,
               Top & Bottom get the 2nd value: -->
<Label Padding="20,5" Background="Orange">20,5</Label>

<!-- 4 values: Left,Top,Right,Bottom: -->
<Label Padding="0,10,20,30" Background="Orange">0,10,20,30</Label>
```

Controlling Position - Alignment

- The HorizontalAlignment and VerticalAlignment properties enable an element to control what it does with any extra space that its parent panel gives it. Each property has a corresponding enumeration as :
  - **HorizontalAlignment** — Left, Center, Right, and Stretch
  - **VerticalAlignment** — Top, Center, Bottom, and Stretch

Controlling Position - Alignment




```
<StackPanel>
  <Button HorizontalAlignment="Left" Background="Red">Left</Button>
  <Button HorizontalAlignment="Center" Background="Orange">Center</Button>
  <Button HorizontalAlignment="Right" Background="Yellow">Right</Button>
  <Button HorizontalAlignment="Stretch" Background="Lime">Stretch</Button>
</StackPanel>
```

Controlling Position – Content Alignment

The HorizontalContentAlignment and VerticalContentAlignment properties determine how a control's content fills the space *within* the control.


```
<StackPanel>
  <Button HorizontalContentAlignment="Left" Background="Red">Left</Button>
  <Button HorizontalContentAlignment="Center" Background="Orange">Center</Button>
  <Button HorizontalContentAlignment="Right" Background="Yellow">Right</Button>
  <Button HorizontalContentAlignment="Stretch" Background="Lime">Stretch</Button>
</StackPanel>
```

Applying Transforms

- Built-in 2D transforms, all in the System.Windows.Media namespace:
  - RotateTransform
  - ScaleTransform
  - SkewTransform
  - TranslateTransform
  - MatrixTransform

Applying Transforms - RotateTransform

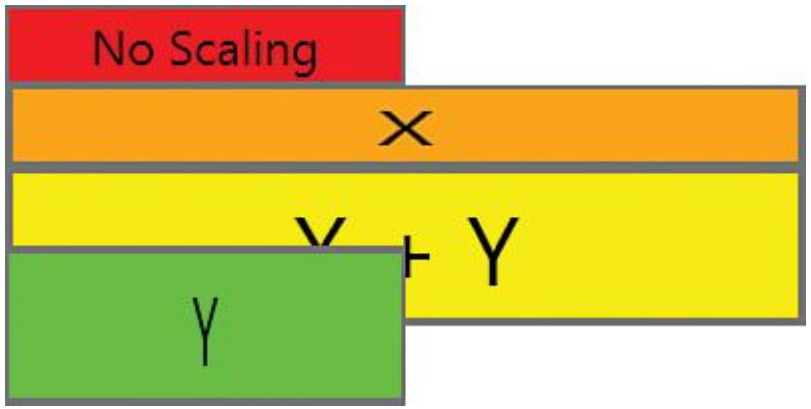
- RotateTransform, demonstrated in the preceding section, rotates an element according to the values of three double properties:
  - Angle—Angle of rotation, specified in degrees (default value = 0)
  - CenterX—Horizontal center of rotation (default value = 0)
  - CenterY—Vertical center of rotation (default value = 0)

- The default (CenterX,CenterY) point of (0,0) represents the top-left corner.

Applying Transforms - ScaleTransform

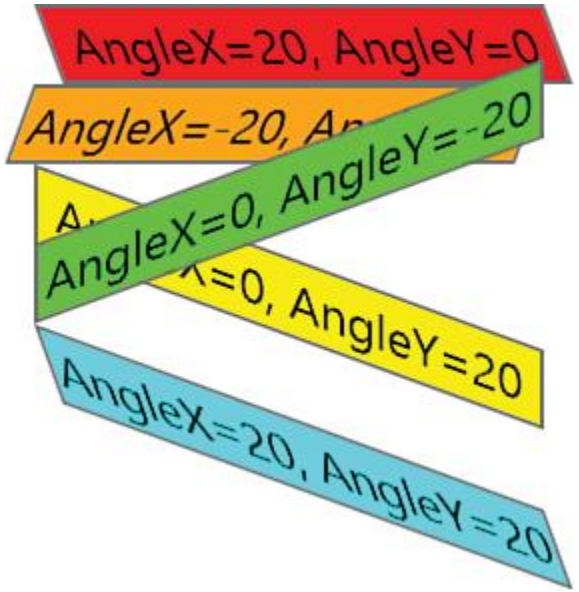
- ScaleTransform enlarges or shrinks an element horizontally, vertically, or in both directions. This transform has four straightforward double properties:
  - ScaleX—Multiplier for the element's width (default value = 1)
  - ScaleY—Multiplier for the element's height (default value = 1)
  - CenterX—Origin for horizontal scaling (default value = 0)
  - CenterY—Origin for vertical scaling (default value = 0)

```
<StackPanel Width="100">
  <Button Background="Red">No Scaling</Button>
  <Button Background="Orange">
  <Button.RenderTransform>
    <ScaleTransform ScaleX="2"/>
  </Button.RenderTransform>
    X</Button>
  <Button Background="Yellow">
  <Button.RenderTransform>
    <ScaleTransform ScaleX="2" ScaleY="2"/>
  </Button.RenderTransform>
    X + Y</Button>
  <Button Background="Lime">
  <Button.RenderTransform>
    <ScaleTransform ScaleY="2"/>
  </Button.RenderTransform>
    Y</Button>
</StackPanel>
```



Applying Transforms - SkewTransform

- SkewTransform slants an element according to the values of four double properties:
  - AngleX—Amount of horizontal skew (default value = 0)
  - AngleY—Amount of vertical skew (default value = 0)
  - CenterX—Origin for horizontal skew (default value = 0)
  - CenterY—Origin for vertical skew (default value = 0)



Applying Transforms - TranslateTransform

- TranslateTransform simply moves an element according to two double properties:
  - X—Amount to move horizontally (default value = 0)
  - Y—Amount to move vertically (default value = 0)

Applying Transforms - TransformGroup

- TransformGroup is to combine child Transform objects

```
<Button>
<Button.RenderTransform>
  <TransformGroup>
    <RotateTransform Angle="45"/>
    <ScaleTransform ScaleX="5" ScaleY="1"/>
    <SkewTransform AngleX="30"/>
  </TransformGroup>
</Button.RenderTransform>
  OK
</Button>
```