Arranging FrameworkElements

Layout (FrameworkElement)

Layout is the process whereby each `FrameworkElement` is sized, positioned and rendered onto the screen. A parent `Panel` allocates a rectangular subset of its total available space to each child. The size and location of this rectangle defines a **layout slot** or **bounding box** onto which the child can be rendered. As a panel's children can also be panels, layout is implemented as a two-pass traversal of the element tree. In the first pass, known as **measure**, parents ask their children how much space they would like.

Inputs into process

- Available screen space
- Size of constraints
- Layout specific properties (margin and padding)
- Logic and behaviour of the parent panel

Layout occurs when an application starts and every time a window is resized.

Overview of Measure/Arrange

A specialized piece of code can show how measure arrange between parent and child works in full generality. In this example the parent is a Grid panel of size (500,500) which contains four identically sized cells of (250,250). The child is hence passed (250,250) to its measure which it ignores and asks for (109,109). This value of (109,109) is then stored in the DesiredSize property. The parent then invokes arrange on the child and passes in (109,109) which the child ignores and asks for (150,150) which forms the basis of the RenderSize

Listing 1Measure Arrange
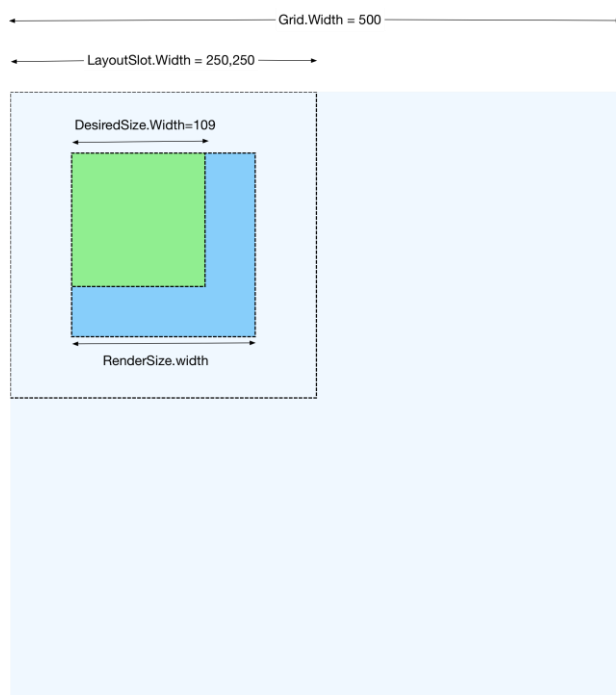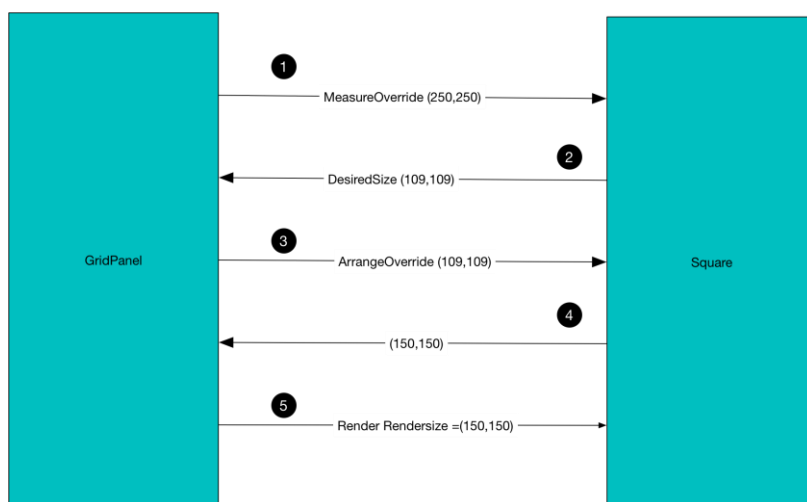
Measure

A parent will invoke Measure on its child FrameworkElement. Measure is defined as a non-virtual method in the UIElement base class. Measure in turn invokes the virtual method MeasureOverride enabling the subclass to specify its content size. The Measure method then derives and populates the element's DesiredSize property from the various layout properties set on the element and the value returned from its MeasureOverride.

Listing 2Measure
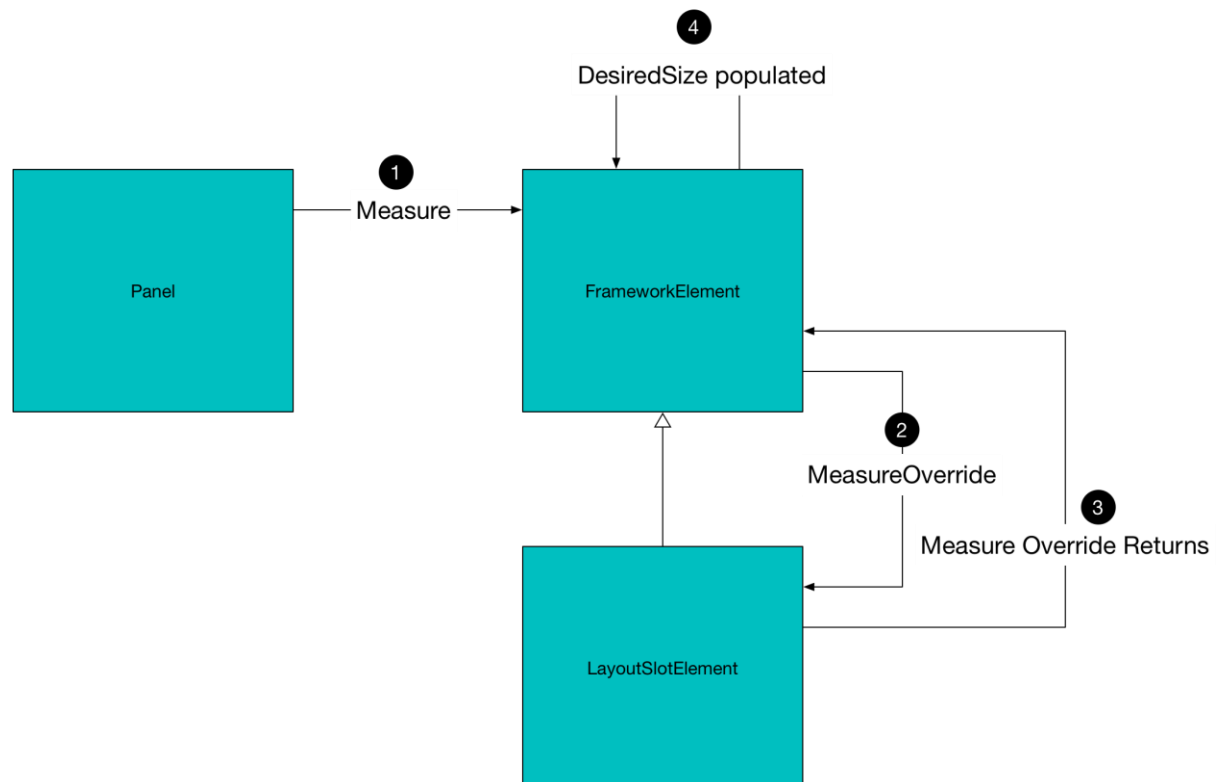
Child Layout Properties

We can split child element layout properties into three categories

- ♦ Size related properties
- ♦ Position related properties
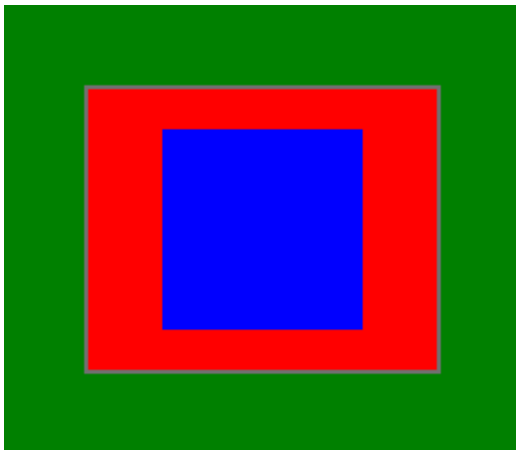- ♦ Transforms

Size Related Properties

MARGIN

Margin specifies how much space to leave around the outside of an element during layout.
Padding put space between inside the control and around the controls content.

Listing 3Margin and Padding

```xml
<Window x:Class="ChildProperties.Size.MarginAndPadding"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        WindowStyle="None"
        Background="Green"
        SizeToContent="WidthAndHeight">
    <Border>
        <Button Background="Red" Margin="20" Padding="10">
            <Rectangle Width="50" Height="50" Fill="Blue"></Rectangle>
        </Button>
    </Border>
</Window>
```

Listing 4 Margin and Padding

MaxHeight/MaxWidth

The maxwidth and maxheight control how much components stretch to fill the available space when the available space is larger than the amount they need. In the below example our button starts at a size of 100x100 and as we resize the window larger vertically and horizontally it stretched to fit the available space until it hits the limit of 150x150
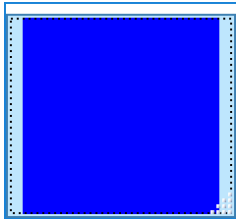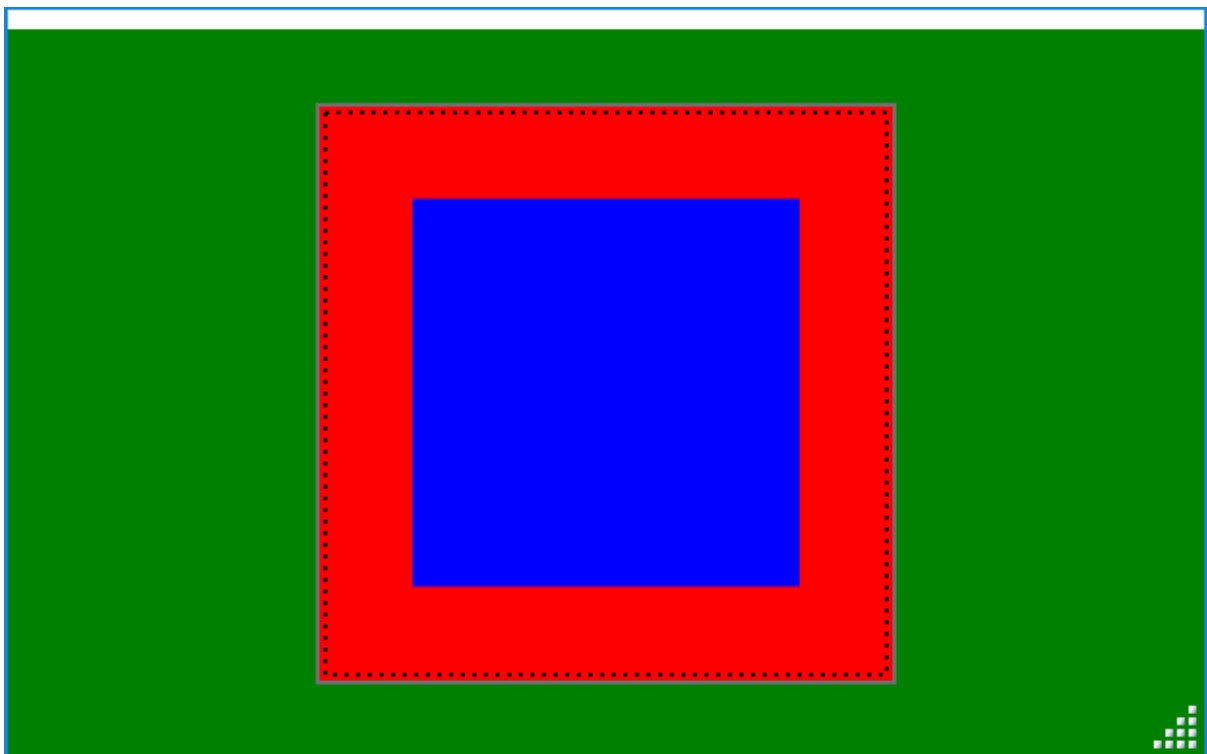
Figure 1Before Resize


Figure 2After Resize

MINHEIGHT/MINWIDTH

The minwidth and minheight control the minimum size of a component as the available space shrinks. If the available space is less than the min then the component is cropped

Figure 3Min Width/Height

```
<Window x:Class="ChildProperties.Size.MinWidthHeightxaml"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        WindowStyle="None"
        Background="Green"
        ResizeMode="CanResizeWithGrip"
        SizeToContent="WidthAndHeight">
    <Border>
        <Button  MinWidth="150" MinHeight="150">
            Click Me
        </Button>
    </Border>
</Window>
```

Figure 4Min Width/Height
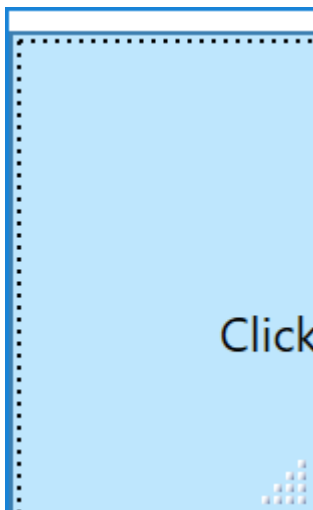
Position related properties

ALIGNMENT

In some cases a parent will allocate a child more space than it needs. The HorizontalAlignment and VerticalAligment properties determine how the control makes use of this extra space. The default value for these properties if not explicitly set is 'stretch'. **???**Furthermore these properties will have no effect if the parent container allocated less than the amount of space the child needs.

```
Horizontal=Stretch,Vertical=Stretch

Horizontal=Left,Vertical=Stretch

                                        Horizontal=Right,Vertical=Stretch

                    Horizontal=Stretch,Vertical=Top

                    Horizontal=Center,Vertical=Center
```
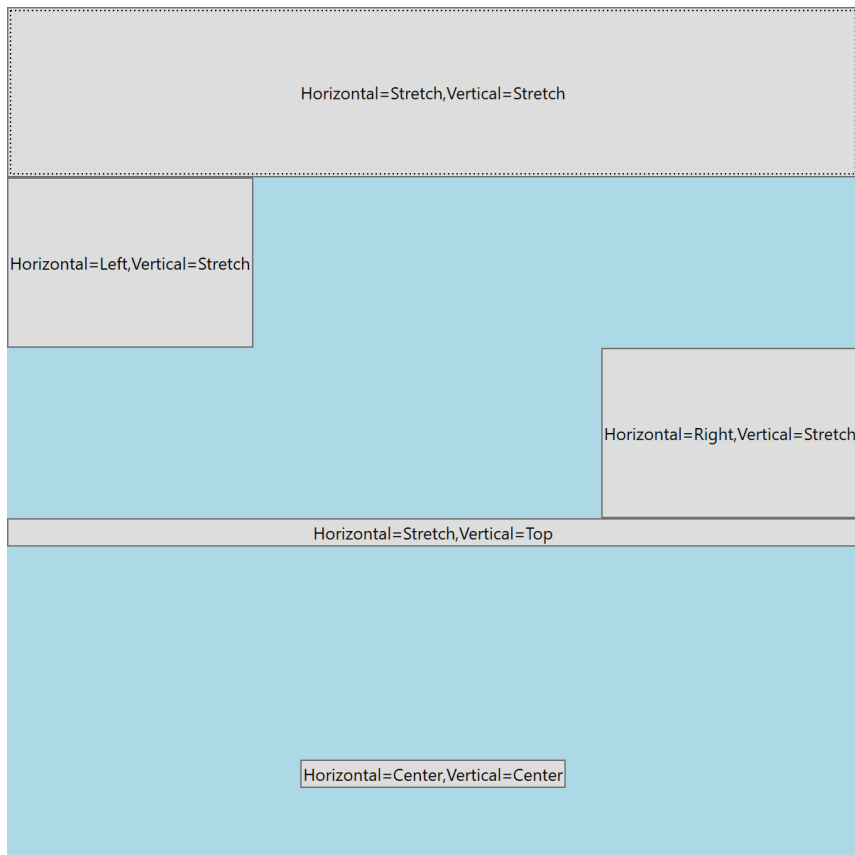
Scenarios

The following scenarios make use of a custom FrameworkElement for exposition

SCENARIO A BASIC EXAMPLE

```xml
<Window x:Class="Layout.CodeSamples.A_MeasureArrange.Scenarios.Scenario_A"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:helperCode="clr-namespace:Layout.CodeSamples.HelperCode"
        WindowStyle="None"
        ResizeMode="NoResize"
      Title="DesiredSize" Height="300" Width="300">
    <Grid>
        <helperCode:LayoutSlotElement
            Fill="LightBlue"
            MeasureOverrideSize="170,170"
            HorizontalAlignment="Center"
            VerticalAlignment="Center" >
        </helperCode:LayoutSlotElement>
    </Grid>
</Window>
```

When the parent grid invokes Measure on the LayoutSlotElement it passes in all the windows available space (300,300) [Because we have no explicitly specified rows/columns the grid

has a single cell with * width and height. The child in this single cell is given all available space].

The LayoutSlotElement's MeasureOverride property is set to 170,170 so in the absence of any other layout properties the desired size becomes (170,170). The parent then allocates a layout slot of the whole available space (300,300) but because the LayoutSlotElement's alignment properties are set to center it does not use the extra space and renders at 170,170
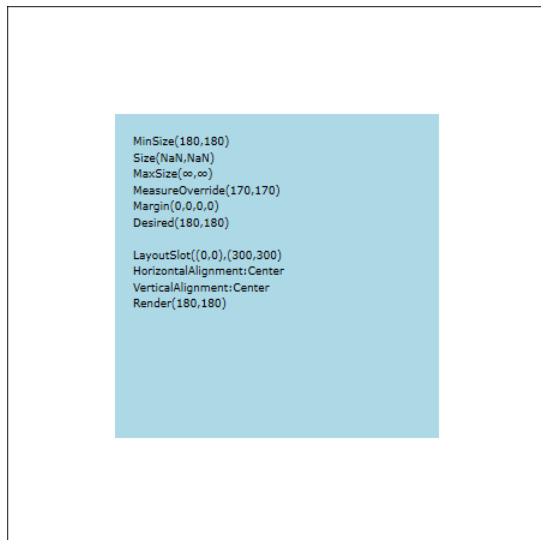
```
MinSize(0,0)
Size(NaN,NaN)
MaxSize(∞,∞)
MeasureOverride(170,170)
Margin(0,0,0,0)
Desired(170,170)

LayoutSlot((0,0),(300,300)
HorizontalAlignment:Center
VerticalAlignment:Center
Render(170,170)
```

SCENARIO B – MINSIZE GREATER THAN MEASUREOVERRIDE

```xml
<Window x:Class="Layout.CodeSamples.A_MeasureArrange.Scenarios.ScenarioB_MinSize"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
          xmlns:helperCode="clr-namespace:Layout.CodeSamples.HelperCode"
        WindowStyle="None"
        ResizeMode="NoResize"
      Title="DesiredSize" Height="300" Width="300">
    <Grid>
        <helperCode:LayoutSlotElement
          Fill="LightBlue"
          MeasureOverrideSize="170,170"
          MinWidth="180" MinHeight="180"
          HorizontalAlignment="Center"
          VerticalAlignment="Center" >
        </hlperCode:LayoutSlotElement>
    </Grid>
</Window>
```

If we set the MinWidth and MinHeight to be greater than the value returned from MeasureOverride the layout system in Measure takes this into account and widens the DesiredSize to MinWidth, MinHeight.

> One difference between this logic and the code in the next scenario is that when MeasureOverride is invoked by Measure it still passes in all the available space of 300,300. [In the next example where Width and Height are set the layout system invokes MeasureOverride with 180,180 or the values set in Width and Height]

MinSize(180,180)
Size(NaN,NaN)
MaxSize(∞,∞)
MeasureOverride(170,170)
Margin(0,0,0,0)
Desired(180,180)

LayoutSlot((0,0),(300,300))
HorizontalAlignment:Center
VerticalAlignment:Center
Render(180,180)

SCENARIO C SIZE GREATER THAN MEASUREOVERRIDE

```xml
<Window x:Class="_0003SizeA.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:views="clr-namespace:RiskAndPricingSolutions.WPF.SharedResources.Views;as-
sembly=SharedResources"
        WindowStyle="None"
        ResizeMode="NoResize"
        Title="DesiredSize" Height="300" Width="300">
    <Grid>
        <views:LayoutSlotElement
            Fill="LightBlue"
            Margin="10"
            HorizontalAlignment="Center"
            VerticalAlignment="Center"
            MeasureOverrideSize="170,170"
            Width="180" Height="180"/>
    </Grid>
</Window>
```

The main difference between this and the previous example is that the layout system passes in Width/Height of 180,180 to MeasureOverride rather that the whole available 300x300. This will be more apparent when we use stretch alignment properties

MinSize(0,0)
Size(180,180)
MaxSize(∞,∞)
MeasureOverride(170,170)
Margin(10,10,10,10)
Desired(200,200)

LayoutSlot((0,0),(300,300)
HorizontalAlignment:Center
VerticalAlignment:Center
Render(180,180)

SCENARIO D SIZE AND MINSIZE SPECIFIED – MINSIZE GREATER

```xml
<Window
x:Class="Layout.CodeSamples.A_MeasureArrange.Scenarios.ScenarioD_SizePlusMinSize"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
         xmlns:helperCode="clr-namespace:Layout.CodeSamples.HelperCode"
      WindowStyle="None"
        ResizeMode="NoResize"
       Title="DesiredSize" Height="300" Width="300">
    <Grid>
        <helperCode:LayoutSlotElement
            Fill="LightBlue"
            MeasureOverrideSize="170,170"
            HorizontalAlignment="Center"
            VerticalAlignment="Center"
            Width="180" Height="180"
            MinWidth="190" MinHeight="190"
            />
    </Grid>
</Window>
```

Here the MinSize is wider than Size and MeasureOverride so it drives the DesiredSize. Interestingly the presence of the Size properties causes the parent to not pass in the full 300,300 but instead an available size of 190,190
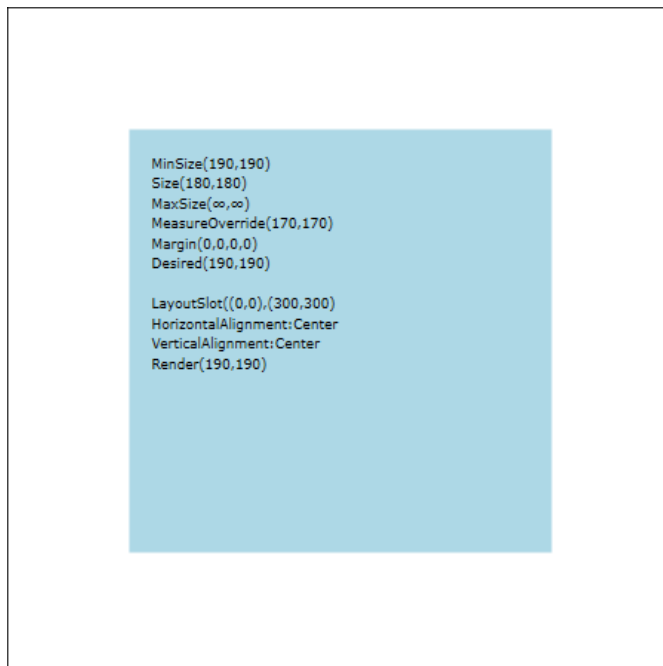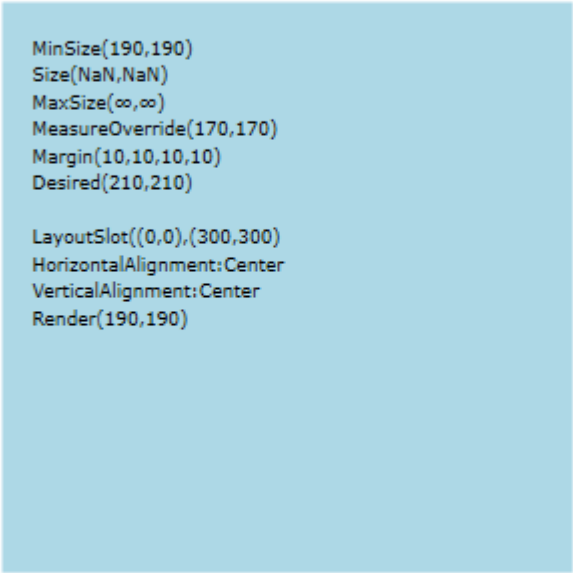
SCENARIO E – MARGIN

The margin gets added in when calculating DesiredSize

```xml
<Window x:Class="Layout.CodeSamples.A_MeasureArrange.Scenarios.ScenarioE_Margin"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:helperCode="clr-namespace:Layout.CodeSamples.HelperCode"
        WindowStyle="None"
        ResizeMode="NoResize"
      Title="DesiredSize" Height="300" Width="300">
    <Grid>
        <helperCode:LayoutSlotElement
            Fill="LightBlue"
            MeasureOverrideSize="170,170"
            HorizontalAlignment="Center"
            VerticalAlignment="Center"
            MinWidth="190" MinHeight="190"
            Margin="10"
            />
    </Grid>
</Window>
```

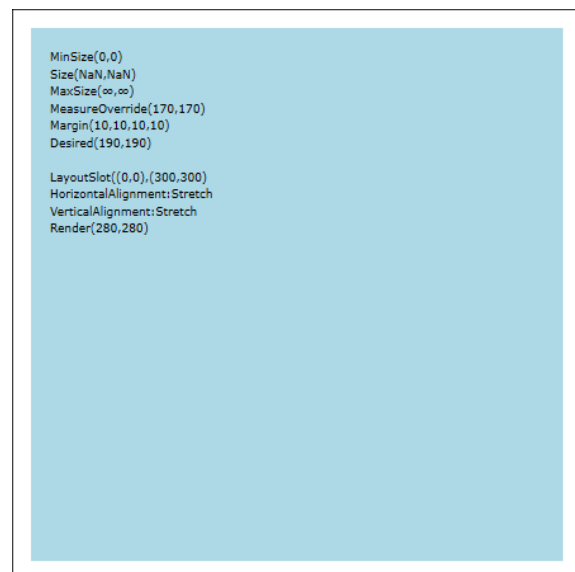The margin gets added to the MinSize to form the DesiredSize. Notice RenderSize does not include the Margin

MinSize(190,190)
Size(NaN,NaN)
MaxSize(∞,∞)
MeasureOverride(170,170)
Margin(10,10,10,10)
Desired(210,210)

LayoutSlot((0,0),(300,300))
HorizontalAlignment:Center
VerticalAlignment:Center
Render(190,190)

SCENARIO F ALIGNMENT

```xml
<Window x:Class="Layout.CodeSamples.A_MeasureArrange.Scenarios.ScenarioF_Alignment"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:helperCode="clr-namespace:Layout.CodeSamples.HelperCode"
        mc:Ignorable="d"
    WindowStyle="None"
        ResizeMode="NoResize"
        Title="DesiredSize" Height="300" Width="300">
    <Grid>
        <helperCode:LayoutSlotElement
            Fill="LightBlue"
            MeasureOverrideSize="170,170"
            HorizontalAlignment="Stretch"
            VerticalAlignment="Stretch"
            Margin="10"
            />
    </Grid>
</Window>
```

In this case as the Grid is taking all 300,300 and allocating it to the LayoutSlotElement's layout slot changing the Alignment to stretch has the following effect. The render size is the whole 300,300 less margins giving 280,280. Much larger than the desiredsize.


```
MinSize(0,0)
Size(NaN,NaN)
MaxSize(∞,∞)
MeasureOverride(170,170)
Margin(10,10,10,10)
Desired(190,190)

LayoutSlot((0,0),(300,300))
HorizontalAlignment:Stretch
VerticalAlignment:Stretch
Render(280,280)
```

Scenario G Max size

```xml
<Window x:Class="Layout.CodeSamples.A_MeasureArrange.Scenarios.ScenarioG_MaxSize"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:helperCode="clr-namespace:Layout.CodeSamples.HelperCode"
        mc:Ignorable="d"
    WindowStyle="None"
        ResizeMode="NoResize"
      Title="DesiredSize" Height="300" Width="300">
    <Grid>
        <helperCode:LayoutSlotElement
            Fill="LightBlue"
            MeasureOverrideSize="170,170"
            HorizontalAlignment="Stretch"
            VerticalAlignment="Stretch"
            MaxWidth="250" MaxHeight="250"
            Margin="10"
            />
    </Grid>
</Window>
```

We can limit the amount of stretch by using the max size

Panels

A panel is an interface element whose job it is to arrange the elements it contains. Each panel type offers a straightforward layout mechanism. Panels can them be composed to generate more complex arrangements. An elements position is always determined by its containing container. Most panels also manage the size of their children.

Window Layout Style

FIXED WIDTH VERSUS SIZETOCONTENT

A window will pass one of two layout styles to its child panel when asked to layout its content. `SizeToContent` and `FixedWidth`. Size to content will occur when a window whose SizeToContent attribute is set to WidthAndHeight first opens. In this situation the panel can determine what size it needs in order to layout its contents. The window will then occupy as much space as is needed. Fixed with will occur when a window is resized by the user. In this situation the window will pass to its panel the exact with and height. It is then up to the panel to make do with what it is given.

TRAVERSING THE HIERARCHY

When a window asks its top level panel to layout its contents according to given layout style it does not necessarily pass the same style to its children. Consider a window whose SizeToContent property is set to WidthAndHeight whose content consists of a single vertical stack panel. Let the stackpanel contain four buttons. Notice that the height of each button is determined by its content,. The stack panel has asked each button what size it needs for its content and then allocated it what it needs. On the other hand the width of each button widens and shrinks as the window is resized by the user. The stackpanel is allocating the entire width of to each of its contained buttons causeing them to widen and shrink with the window

LAYOUT STYLE

Fixed Size – If the user resizes the window then its size is imposed on the layout system, i.e it is fixed. The job of the layout system is to arrange the content as best it can

Size to content – Occurs when a window whose SizeToContent is set to WidthAndHeight then the windows initial size is obtained by measuring its content

Impact of layout style on children – A panel subject to a fixed layout does not necessarily pass it on its children

Clipping

UIElements can paint outside the bounds of their containing containing Panel if the panels `ClipToBounds` is set to false

Listing 5No Clipping

```xml
<Canvas Width="150" Height="150" Background="LightGray"
ClipToBounds="False">
    <Rectangle Width="100" Height="100" Fill="Gray" Canvas.Top="-10"
Canvas.Left="-10"/>
</Canvas>
```
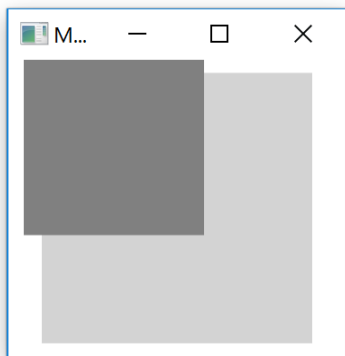

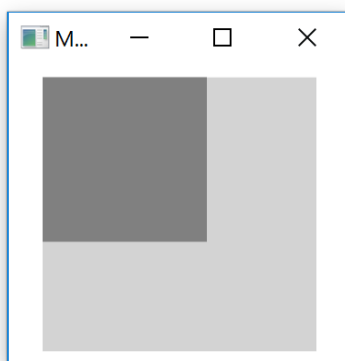*Listing 6Clipping*

```xml
<Canvas Width="150" Height="150" Background="LightGray" ClipToBounds="True">
    <Rectangle Width="100" Height="100" Fill="Gray" Canvas.Top="-10"
Canvas.Left="-10"/>
</Canvas>
```

Layout Questions

MEASURE ARRANGE

What is layout?

The process whereby each Framework element is sized, positioned and rendered

What happens during layout?

Parent panels allocate rectangular subsets of their total available space to each child.

How is the size and location for each child specified?

As a rectangle known as the layout slot onto which the child is rendered

How is layout performed?

As a recursive two-phase pass known as Measure-Arrange

What do the passes do?

During Measure parents ask children how much space they need

During Arrange the children are told how much space they have been allocated

When does layout occur?

When an application starts

every time a window is resized.

What are inputs into this process?

Available screen space

Size constraints

Layout specific properties such as margin and padding

Logic and behaviour of parent panels

Where is the result of measure stored?

DesiredSize

How is DesiredSize determined?

The various layout properties of the element and the value returned from MeasureOverride.

Where is the result of arrange stored?

RenderSize

CHILD LAYOUT PROPERTIES

What are the size related properties?

Margin – How much space to leave around outside of element during layout

Padding – How much space around a controls content and its border or margin

MaxHeight/MaxWidth – bounds on how much an element can stretch when its available space is larger than needed

MinHeight/MinWidth – Minimum size as available space shrinks

What happens if available size is less than min size and min height?

Cropping

What are position related properties?

Horizontal Alignment and Vertical Alignment – used if parents allocated more space than needed

When can a UIElment paint outside its bounds?

If its ClipToBounds are set to false

What is a Panel?

An interface element that arranges its children

What are the WPF Panels?

GridPanel

StackPanel

DockPanel

Canvas

WrapPanel