Rendering

Painting WPF interfaces onto the screen

Visual

From a high level, any UI technology is about processing user input from the mouse or keyboard and rendering the resultant state to the screen. Rendering in WPF is supported via the abstract base class **`Visual`**. Visual is a lightweight implementation that supports

- Specifying and rendering drawing content
- Hit Testing
- Bounding box calculations
- Transformations
- Clipping

The Visual Tree

A Visual can have zero or more child visuals arranged into a hierarchical structure known as the visual tree. The hierarchical aspect of Visual is supported by one property and one method

- VisualChildCount
- GetVisualChild

UIElement extends Visual and FrameworkElement extends UIElement meaning all WPF Framework Elements can be rendered to the screen.

Visual Tree Examples

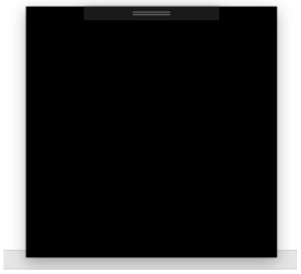
The following sections contain code that highlight how visual trees are constructed and rendered.

SINGLE NODE VISUAL TREE

Because **`FrameworkElement`** extends **`Visual`** all framework elements can be rendered. The root element in any visual tree is Window. We can create a one node visual tree by sub classing Window and overriding the **`VisualChildCount`** property to return zero. This informs the rendering engine not to look for children of this node.

> Window is a complex subclass of ContentControl which supports layout and other more advanced topics which I don't want to cover here. By overriding `OnRender` with a no-op we effectively prevent the window from doing any rendering of itself as part of layout.

As our single node has no rendering instructions our window shows just as a black box on the screen.



Listing Single node tree

```
public class SingleNodeVisualTree : Window
{
    public SingleNodeVisualTree()
    {
        Width = Height = _diameter;
        WindowStyle = WindowStyle.None;
        ResizeMode = ResizeMode.NoResize;
        MouseLeftButtonDown += (sender, args) => DragMove();
        AllowsTransparency = false;
    }

    // First we set overide VisualChildrenCount to indicate
    // this Window has no Visual Children
    protected override int VisualChildrenCount => 0;

    private readonly double _diameter = 200;
}
```

Adding a node

A window with no visual representation is fairly useless. If we add a single child visual, we can render our window as a single red circle with the same diameter as the window itself. We do this by overriding `VisualChildrenCount` to be one. We then create a DrawingVisual which we return from the `GetVisualChild` method. Our window then renders as follows.

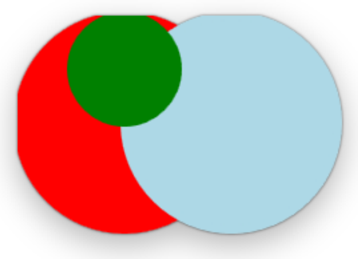
And the source code is as follows.

> **Visuals must have at least one FrameworkElement ancestor**
>
> Visuals are lightweight objects. As we have mentioned they can be arranged into arbitrarily deep hierarchies in order to efficiently render complex content. One proviso is that a hierarchy of visuals and sub visuals must at some stage be added to at least one Framework Element in order to render. As Window is a FrameworkElement this requirement is met by all the samples in this document.

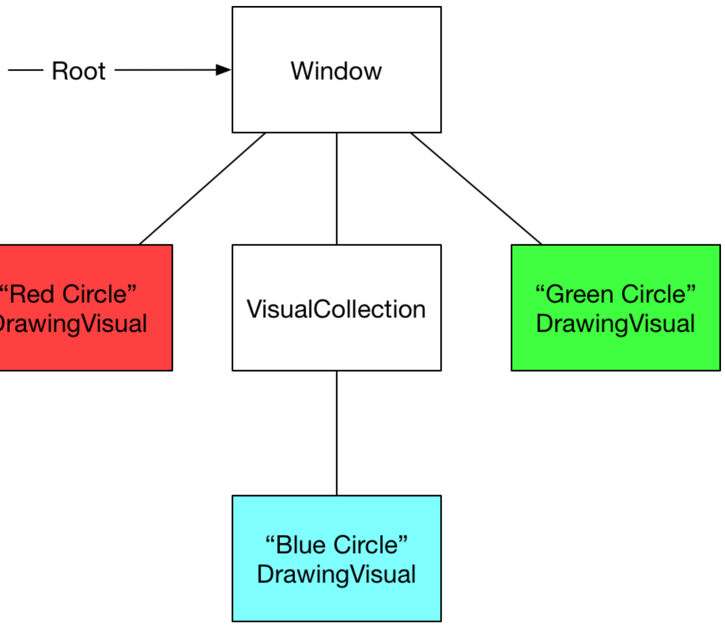
```csharp
public class TwoNodeVisualTree : Window
{
    public TwoNodeVisualTree()
    {
        SetupWindow();
        AddVisualChild(VisualChildVisual);
    }

    // (1) Overide VisualChildrenCount to indicate
    // this Window has a single Visual Child
    protected override int VisualChildrenCount => 1;

    // (2) Create the visual child
    protected Visual VisualChildVisual = GetDrawingVisual();

    // (3) Return the single child
    protected override Visual GetVisualChild(int index) => VisualChildVisual;

    private static DrawingVisual GetDrawingVisual()
    {
        // Return a new DrawingVisual with instructions to render
        // a red rectangle
        var dv = new DrawingVisual();
        using (var dc = dv.RenderOpen())
        {
            var radiusX = _diameter / 2.0;
            var center = new Point(radiusX, radiusX);
            var geom = new EllipseGeometry(center, radiusX, radiusX);
            dc.DrawGeometry(Brushes.Red, null, geom);
        }
        return dv;
    }

    protected override void OnRender(DrawingContext drawingContext) { }

    private void SetupWindow()
    {
        Width = Height = _diameter;
        WindowStyle = WindowStyle.None;
        ResizeMode = ResizeMode.NoResize;
        MouseLeftButtonDown += (sender, args) => DragMove();
        AllowsTransparency = true;
    }

    private static readonly double _diameter = 200;
}
```

Visual Trees render depth first

The nodes in a tree are rendering by walking down the tree in a depth first traversal. The z-order is specified by the order of rendering, so visuals rendered later are in the pass are rendered in front of visuals rendered earlier in the pass. The screenshot below shows how the given code listing renders



From a visual tree as follows. Note the window and the internal node do not render anything.


Root
Window
"Red Circle" DrawingVisual
VisualCollection
"Green Circle" DrawingVisual
"Blue Circle" DrawingVisual

```
public class DepthFirstRendering : Window
{
    public DepthFirstRendering()
    {
        SetBasicWindowProperties();

        _children = new VisualCollectionVisual();

        // 1.1 Big Red Circle
        _children.Add(Createcircle(new Point(40, 40), Brushes.Red, 40));
        this.AddVisualChild(_children);

        // 1.2 internal Node doesnt render
        VisualCollectionVisual onePointTwo = new VisualCollectionVisual();
        _children.Add(onePointTwo);

        // 1.3 Medium Green Circle
        _children.Add(Createcircle(new Point(40, 20), Brushes.Green, 20));

        // 2.1 Big White Circle
         onePointTwo.Add(Createcircle(new Point(80, 40), Brushes.LightBlue, 40));
    }

    public DrawingVisual Createcircle(Point location, Brush brush, int radius)
    {
        DrawingVisual dv = new DrawingVisual();
        using (DrawingContext dc = dv.RenderOpen())
        {
            dc.DrawEllipse(brush,new Pen(brush,3),location,radius,radius);
        }

        return dv;
    }

    protected override int VisualChildrenCount => _children.Count;

    protected override Visual GetVisualChild(int index)
    {
        return _children._visualCollection[index];
    }

    private void SetBasicWindowProperties()
    {
        Width = Height = 150;
        WindowStyle = WindowStyle.None;
        ResizeMode = ResizeMode.NoResize;
        MouseLeftButtonDown += (sender, args) => DragMove();
        AllowsTransparency = true;
    }
    private VisualCollectionVisual _children;
}
```

Hit Testing

```
public class HitTestingWindow : Window
 {
     public HitTestingWindow()
     {
         InitializeWindow();

         _visualChildren.Add(CreateVisual(new RectangleGeometry(new Rect(new Point(10,10),new
             Size(100,100))),Brushes.Red));
         _visualChildren.Add(CreateVisual(new RectangleGeometry(new Rect(new Point(20,20),new
             Size(100,100))),Brushes.Green));
         _visualChildren.Add(CreateVisual(new RectangleGeometry(new Rect(new Point(30,30),new
             Size(100,100))),Brushes.Blue));
     }

     protected override void OnRender(DrawingContext drawingContext)
     {
         drawingContext.DrawRectangle(Brushes.Transparent,null,
             new Rect(new Point(0,0),new Size(this.RenderSize.Width,
             this.RenderSize.Height)));
     }

     protected override void OnMouseRightButtonDown(MouseButtonEventArgs e)
     {
         base.OnMouseRightButtonDown(e);
         var dp = GetHitObjects(this, e.GetPosition(this));
     }

     private List<DependencyObject> GetHitObjects(Visual searchRootNode, Point point)
     {
         var hits = new List<DependencyObject>();

         VisualTreeHelper.HitTest(searchRootNode,null, result =>
         {
             hits.Add(result.VisualHit);
             return HitTestResultBehavior.Continue;
         },new PointHitTestParameters(point));

         return hits;
     }

     private Visual CreateVisual(Geometry geometry, Brush fill)
     {
         var dv = new DrawingVisual();
         using (var dc = dv.RenderOpen())
         {
             if (fill != null) dc.DrawGeometry(fill,null,geometry);
         }

         // Note that for painting it is enough to include visuals
         // in the visual tree methods GetVisualChild and
         // VisualChildrenCount. However for hit testing the
         // children must also be added via the AddVisualChild method
         AddVisualChild(dv);
         return dv;
     }

     private void InitializeWindow()
     {
         WindowStyle = WindowStyle.None;
         ResizeMode = ResizeMode.NoResize;
         MouseLeftButtonDown += (sender, args) => DragMove();
         AllowsTransparency = true;
         Foreground = Brushes.White;
         Height = 220;
         Width = 440;
     }

     private readonly List<Visual> _visualChildren = new List<Visual>();
     protected override Visual GetVisualChild(int index) => _visualChildren[index];
     protected override int VisualChildrenCount => _visualChildren.Count;
 }
```

Questions - Rendering

Basic Rendering

What is the visual tree?

The tree of rendered Visual objects

Which two methods control the children of a visual in the visual tree?

1. *Visual.VisualChildrenCount*
2. *Visual.GetVisualChild(int)*

UIElements are lightweight elements that can be arranged in deep hierarchies. What condition must be met for a hierarchy of Visual to be rendered?

They must have at least one FrameworkElement in the visual tree.

In what order are nodes in a tree rendered?

Depth first traversal of the visual tree

Given a visual object how would you render vector graphic geometries onto it?

Get its DrawingContext and invoke DrawGeometry on the context

How does rendering in WPF differ from rendering in Win32

Retained mode rendering system.

Controls

What defines a control's visual tree?

Its Template property of type ControlTemplate

What are three ways you can affect a controls appearance

1. *Directly set properties*
2. *Styles*
3. *Control Templates*

If you create a new ControlTemplate can you just put anything you want in there or are there any restrictions? How does a control indicate what a control template needs to provide in order to be valid?

TemplatePart attributes on the Control class defintion

If you were creating you own custom control how would you create state that could be bound to a ViewModel

By creating DependencyProperties

Layout

Describe how layout is performed in WPF?

Two pass traversal known as the Meaure Arrange pass

Describe the measure arrage interaction between a parent and it child?

1. *Parent asks child how much space it wants by invoking Measure*
2. *Parent calculates how much space to allocate to the child and passes this to the childs arrange method*
3. *Child can use a subset of the space passed to it if necessary*

How is margin dealt with in the measure arrange pass?

When a parent invokes Measure on a child and passes in the amount of available space. The internals of Measure subtract the childs margins before invoking its MeasureOverride method

How does a parent specify the size of the layout slot for the child?

By passing a rectangle to the childs measure method

Which classes expose padding?

Block, Border, Control, TextBlock

What ate the main panels in WPF

StackPanel

GridPanel

Canvas

WrapPanel

DockPanel

Which properties control how a control deals with the situation where it is given more space than it needs to show its content?

Horizontal and Vertical alignment