A Descriptive Analysis on Windows Presentation Foundation (WPF)

Mr. Varusai Mohamed

Lecturer in Faculty of Computing Sciences, Gulf College, Muscat, Sultanate of Oman

Abstract – Windows Presentation Foundation (WPF) is a next generation presentation system which is used to create windows applications with visually stunning user experiences. It uses "System.Windows" namespace which is a subset of .NET Framework. It has more powerful capabilities to simplify the programming experience and it includes additional programming constructs that enhance properties and events. It is one of the new "foundations" which is introduced by Microsoft in .NET 3.0. It is an important component for the future of application development which is used to develop rich, interactive, media-enhanced user interfaces. It improves the productivity enhancement that configures large-scale systems in minutes instead of hours or days. It gives a powerful experience to users, developers and designers. I analyzed some capabilities of WPF and I have shared my experience in this journal. It consist the details of WPF Architecture, XAML and WPF features and some examples using C# programming language.

Keyword: WPF, Windows Presentation Foundation, WPF architecture, WPF Features.

What is WPF?

Windows Presentation Foundation (WPF) is the new graphics subsystem in Windows Vista that will facilitate programmers in developing applications that offer advanced user experiences. It is the invention of Microsoft Corporation to be accessed as a development tool for Web applications and rich client applications. [1]

Why WPF?

This is the world for Windows applications and Web applications. While Windows applications suggest massively rich client functionality, installing Windows applications need significant resources and put together maintenance an unvarying challenge. On the other side, Web applications offers simple way of deployment and maintenance. But it has difficulty in the development process as the Web is so dynamic as well as less than ultimate platform incorporation.

Objective: Providing a development platform that proposes the best of Windows & Web worlds, agreeing administrator to install and handle applications securely. [2]

What is XAML?

XAML stands for eXtensible **A**pplication **M**arkup **Language**. It is an all-purpose XML-based language developed by Microsoft. It can be called as the language behind as it reflects the Visual Presentation of an application. It is used by WPF application developers to declare the layout of a user interface (UI), and the resources used in that UI. [3]

It is not compulsory to use XAML when programming in WPF. No matter which that can be done in XAML can also be done in code. Using XAML makes many UI development circumstances a lot easier and faster, for example constructing the layout of a UI and configuring styles, templates, and other WPF-specific units.

Code-behind files

XAML is compiled down into a class. If we don't state the class name it should use, the compiler will create a class name automatically. Still, when you apply the x:Class element to the root XML element in a XAML file, you can generate a partial class in C# or VB.NET which will be combined with the XAML partial. Thus we can connect the behavior with the layout and visuals stated in XAML. [4]

How WPF works?

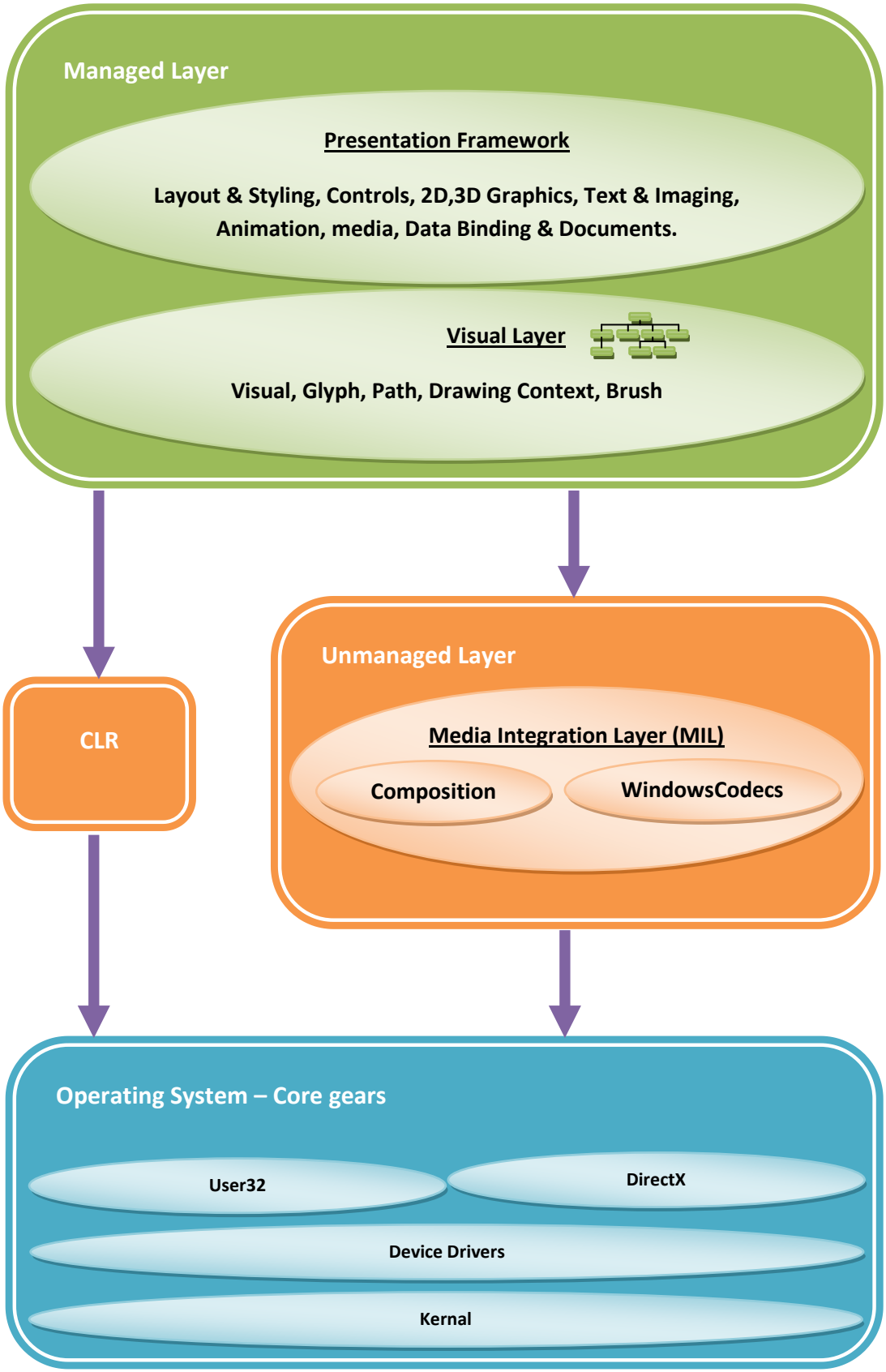
The working method of WPF mainly covers managed code and native code components. Open APIs are supplied by the Managed code.

PresentationFramework, PresentationCore, and milcore are the 3 major code areas.

1. Presentation features of an application are managed by Presentation Framework (presentationframework.dll) which takes responsibility for the end users. The major WPF subsystems are Object, Threading. Dispatcher Object, Windows. Dependancy Object, Windows.Media.Visual, Windows.UIElement, Windows. Framework Element, Windows.Controls.Control.
2. PresentationCore (presentationcore.dll) outfit the main services for WPF.
3. MIL (Media Integration Layer) is the composition engine of WPF. It is a native component and resides in

the milcore.dll. This library is written in unmanaged code. To support 2D and 3D surfaces, it is linked with DirectX at this stage. WPF displays are done through the DirectX engine. With WPF, developers can use XAML, the Extensible Application Markup Language, to create custom controls, graphics, 3D images and animations that are not available in traditional HTML implementations. [6]

Figure 1: WPF Architecture



Declarative Programming

The class for the form will be automatically generated. The file contains auto generated code as below. We need not modify the code.

```
public class Sample
{
        public Sample ()
        {
        }
}
```

XAML instantiate the class using the parameterless constructor and it can't be a nested class. It can have other constructors also but they are not need by XAML. Structs also can be used by XAML as they have a default parameterless constructor provided automatically by the system.

The next step is to link between the XAML document and the class definition which is done by importing the class's namespace into XAML.

We can import the namespace of any assembly and use namespaces to specify just which class we are referring to. For this, we need to import the namespace of the assembly that the XAML file is part of, i.e. the current project.

Accordingly moving to the XAML editor we have to add a single line to the <Window> tag:

```
<Window x:Class="SampleApp.Window1"
  xmlns="http://schemas.microsoft.com/
  winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/
  winfx/2006/xaml"
  xmlns:m="clr-namespace:SampleApp"
  Title="Window1" Height="300"
  Width="300" Loaded="Window_Loaded">
```

To create the instance of our class all we have to state as:

<m: Sample > </m: Sample > between the <Window> and </Window> tag.

Screen resolution independent

WPF window will show on any monitor at the same size despite of the resolution the monitor is using.

Control inside controls

WPF brought in a very good feature, now we can add a control inside another control which is not possible in Windows application. [7]
Ex: add a label inside a textbox

Control Templates

Changing the shape of a control is big challenge for developers using Windows application. WPF defines a control template for this purpose.
Ex: change the shape of a button to elliptical or circle.

Control Transformations

WPF has lots of 2D transforms using which it change the size, position, rotation angle of controls. It allows skewing too. [8]

Transformations can be done in two ways:

- Layout Transform - Transform is applied before the control is rendered on the form
- RenderTransform - Transform is applied after the rendered is laid on the form

Types of Transforms:

1. Translate - moves the control based on x and y values
2. Scale - enlarges or shrinks the controls in x and y axis
3. Rotate - rotates the controls to a specified angle
4. Skew - slant the control
5. Matrix - merges all the above transforms

Different Layouts

WPF offer us a wide and dominant set of layout controls to present them neat on the window by grouping them on the interface. They are:

- Stack Panel:
  It positions its child controls either horizontally or vertically based on the specified Orientation.
- Wrap Panel:
  It positions the child controls from left to right and as the name implies it goes to the new line once it fills up the container width.
- Dock Panel:
  It docks the child control either to Top, Bottom, and Right or to Left based on the Dock type specified.
- Grid:
  It allows us to create a table like structure on the WPF window, it has rows and columns. It helps

the user to place the controls in the desired cell of the grid layout.

- Canvas:
  It allows us to place the controls as we wish. In this layout, the developer has the overall designing control.

2D, 3D Graphics (DirectX & Vector based rendering), Animations & Media

WPF comes with complete rich component all in one. We can use controls as 3D layout or 2D layout. In this, you can also use animation, media file and graphics. It supports playing video or audio file is supported by Media player. [10]

Effective Databinding

The effectiveness includes that we need not worry about the synchronization of data between the data source and the UI element. The data binding framework of WPF will take care of the same where as in ASP.NET or Windows application we have to concentrate on submitting the updates done in the UI data to the data source. [11]

This is achieved by specifying the target UI element property as a dependency property as given in the below XAML.

```
1. <TextBlock Text='{Binding Path=Address}' />
```

This confirms that the property 'Address' and Text property of the TextBlock control will be in Sync. The very special advantage of WPF is that it supports a variety of data sources like XML, ADO.NET classes, LINQ queries, types of IEnumerable and even other UI elements. [12]

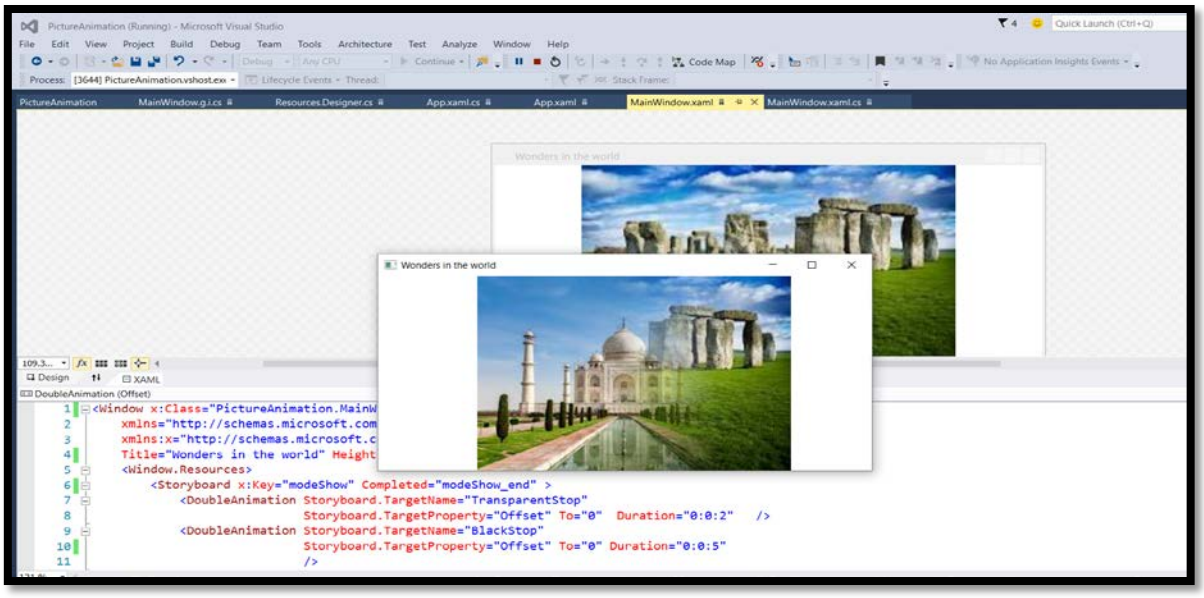
Project : PictureAnimation.cs

This is a very simple project to explain one of the classes "StoryBoard" and perform animations using it. [13]

We might have used portions of software that animates images or manipulates video content. A Storyboard is a place where animation information is stored. A storyboard decides at what point in time a frame will occupy a specific position and will have specific properties.

In WPF, a storyboard is absolutely the same concept. We can work with the storyboard in three ways – by using Expression Blend, which will help us some time since we won't be writing code manually, by using XAML and by using the code-behind capabilities (C# or VB.NET).

A Storyboard is a type of container timeline that provides targeting information for the timelines it contains. A Storyboard can contain any type of Timeline, including other container timelines and animations. Storyboard objects enable you to combine timelines that affect a variety of objects and properties into a single timeline tree, making it easy to organize and control complex timing behaviours like changing the size, color and appearance of the control during the user interaction.



MainWindow.Xaml

```
<Window x:Class="PictureAnimation.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Wonders in the world" Height="300" Width="600" Loaded="Window_Loaded">
    <Window.Resources>
        <Storyboard x:Key="modeShow" Completed="modeShow_end" >
            <DoubleAnimation Storyboard.TargetName="TransparentStop"
                             Storyboard.TargetProperty="Offset" To="0"  Duration="0:0:2"   />
            <DoubleAnimation Storyboard.TargetName="BlackStop"
                             Storyboard.TargetProperty="Offset" To="0" Duration="0:0:2"
                             />
        </Storyboard>
        <Storyboard x:Key="modeHide" Completed="modeHide_end">
            <DoubleAnimation Storyboard.TargetName="TransparentStop"
                             Storyboard.TargetProperty="Offset" To="1"  Duration="0:0:2"   />
            <DoubleAnimation Storyboard.TargetName="BlackStop"
                             Storyboard.TargetProperty="Offset" To="1" Duration="0:0:2"   />
        </Storyboard>
    </Window.Resources>
    <Window.Triggers>
        <EventTrigger RoutedEvent="Window.Loaded">
            <EventTrigger.Actions>
                <BeginStoryboard Storyboard="{StaticResource  modeShow}"/>
            </EventTrigger.Actions>
        </EventTrigger>
    </Window.Triggers>
    <Grid Name="grid">

        <Image x:Name="picWonderRgt" Source="MyResources/Wonder2.jpg" />
        <Image x:Name="picWonderLeft" Source="MyResources/Wonder1.jpg">
            <Image.OpacityMask>
                <LinearGradientBrush StartPoint="0,0" EndPoint="1,0">
                    <GradientStop Offset="1" Color="Black" x:Name="BlackStop"/>
                    <GradientStop Offset="1" Color="Transparent" x:Name="TransparentStop"/>
                </LinearGradientBrush>
            </Image.OpacityMask>
        </Image>

    </Grid>

</Window>
```

Using Image control's OpacityMask property, an Image is swapped with another one virtually. It is happening only by the LinearGradienBrush Class. In this project the StoryBoard is explained with the feature of changing the appearance of the control by fading the color from Black to Transparent and vice versa. It makes the user to feel only the styled variance of image display and not the logic behind it.

MainWindow.xaml.cs

```
/********************************* Module Header ***********************************\
* Module Name:  MainWindow.xaml.cs
* Project:      PictureAnimation
\***********************************************************************************/
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Media.Imaging;
using System.Windows.Media.Animation;
```

```
namespace PictureAnimation
{
   public partial class MainWindow : Window
   {
      int imgQueueLen;
      List<BitmapImage> bmpWonders = new List<BitmapImage>();

      public MainWindow()
      {
         InitializeComponent();
      }
      private void Window_Loaded(object sender, RoutedEventArgs e)
      {
         bmpWonders.Add(new BitmapImage(new Uri("MyResources/Wonder1.jpg", UriKind.Relative)));
         bmpWonders.Add(new BitmapImage(new Uri("MyResources/Wonder2.jpg", UriKind.Relative)));
         bmpWonders.Add(new BitmapImage(new Uri("MyResources/Wonder3.jpg", UriKind.Relative)));
         bmpWonders.Add(new BitmapImage(new Uri("MyResources/Wonder4.jpg", UriKind.Relative)));

         imgQueueLen = 2;
      }

      private void modeShow_end(object sender, EventArgs e)
      {
          this.picWonderLeft.Source = bmpWonders[imgQueueLen++];
         if (imgQueueLen == bmpWonders.Count)
         {
            imgQueueLen = 0;
         }
         Storyboard sb = this.FindResource("modeHide") as Storyboard;
         sb.Begin(this);
      }

private void modeHide_end(object sender, EventArgs e)
      {
         this.picWonderRgt.Source = bmpWonders[imgQueueLen++];
         if (imgQueueLen == bmpWonders.Count)
         {
            imgQueueLen = 0;
         }
         Storyboard sb = this.FindResource("modeShow") as Storyboard;
         sb.Begin(this);
      }

   }
}
```

CONCLUSION

As WPF is latest, it is up to date with current standards. Microsoft is using it to develop and upgrade applications like Visual Studio. It is very flexible, so that we achieve the best things without having to write or buy new controls. Developers love WPF as it overcomes the need for 3rd party control. I demonstrated the important features of WPF, WPF architecture and the relationship between WPF architecture and XAML file using the images and a sample project. [14]

This article helps to developers to understand the architecture clearly and supports to learners to study on WPF. It uses graphics cards to render the output on the screen by using direct 3D rendering feature. Because of this, it performs smooth drawing and it supports to use operating system controls to build application and to customize them to modify their behavior in the application.

It act as a good platform if the application uses various media types such as video, documents, 3D content or animated transitions between a sequence of images or a combination of all. [15] It also creates a skinned user interface which binds to XML data and it dynamically load portions of a user interface from a web service and it makes a desktop application into a web application because of its navigation style.

REFERENCES

[1] Fran Jarnjak "Flexible GUI in Robotics Applications Using Windows Presentation Foundation Framework and Model View ViewModel Pattern".

[2] Microsoft Corporation, "Introducing Windows Presentation Foundation". Available at: http://msdn.microsoft.com/enus library/aa663364.aspx.

[3] Microsoft Corporation, "XAML Overview" (undated). Available http://msdn.microsoft.com/en-us/library/ms752059.aspx.

[4] https://www.tutorialspoint.com/wpf/wpf_xaml_overview.htm

[5] Architectural Styles and the Design of Network-based Software Architectures, Roy Thomas Fielding.

[6] An introduction to Software Architecture by David Garlan and Mary Shaw.

[a] http://msdn.microsoft.com/en-us/library/ms750441.aspx [b] http://www.zdnet.com/blog/stewart/wpf-and-proving-the-importance-of-experience/269 [c]http://www.socaldotnetarchitecture.org/Portals/0/Repository/WPFArchitecture.pdf

[7] https://en.wikibooks.org/wiki/Introduction_to_.NET_ Framework_3.0/Windows_Presentation_Foundation

[8] http://www.infragistics.com/community/blogs/devtoolsguy /archive/2015/04/17/windows-presentation-foundation-vs-winforms.aspx

[9] http://www.codeproject.com/Articles/140611/WPF-Tutorial-Beginning

[10] http://philkildea.co.uk/james/books/Pro_WPF_CSharp_2010_dotNET_4.pdf

[11] http://sd.blackball.lv/library/Programming_Windows_ Presentation_Foundation_(2005).pdf

[12] https://www.dotnetperls.com/wpf

[13] http://learnwpf.com/

[14] https://books.google.com.om/books?id=Se8OSNB95rcC&pg =PA22&lpg=PA22&dq=wpf&source=bl&ots=wexiMXDoO6&sig=ynG9xSc-OPDrsLf8Jq1sQIuQzc4&hl=en&sa= X&redir_ esc=y#v=onepage&q=wpf&f=false

[15] http://ieeexplore.ieee.org/document/6334585/?reload=true

AUTHOR'S PROFILE

Mr. Varusai Mohamed has received his Bachelor of Science degree in B.Sc. Computer Science from Manonmaniam Sundaranar University in the year 1996 and Master of Computer Application from Manonmaniam Sundaranar University in the year 1999 and M. Phil. Computer Science from Periyar University in the year 2007. At present he is working as a lecturer in the Faculty of Computing Science, Gulf College, Muscat, Oman.