ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 1, Issue 4, June 2012

Application Development Using WPF

Shirish Patil, Sameer Soni, Pranali Dhete and Dr B.B. Meshram
VJTI, Mumbai, India
Email: shirishpatil7@gmail.com
VJTI, Mumbai, India
Email: pranali.dhete@gmail.com
VJTI, Mumbai, India
Email: soni.sameer88@gmail.com

ABSTRACT
The paper focuses on the application development using WPF. It covers the overall architecture of WPF. The programming models of the WPF shows how code and XAML is separated and how XAML objects can be accessed. We already have display technologies User32, GDI,GDI+,DirectX but still many developers go for WPF because these display technologies have their own limitations, that are overcome in WPF. Paper also focuses on hardware acceleration using WPF.

Keywords-- **WPF, XAML, Display Technology, DirectX**

I.NTRODUCTION

Windows Presentation Foundation (WPF) is the new presentation API in WinFX. WPF represents a major step forward in User Interface technology. WPF is a two and three dimensional graphics engine. It has the all equivalent common user controls like buttons, check boxes sliders etc. It has fixed and flow format documents. WPF has all of the capabilities of HTML and Flash. It has 2D and 3D vector graphics, animation, multimedia as well data binding capabilities. WPF supports XAML, XAML is a declarative XML-based language by which user can define object and properties in XML. XAML document is loaded by a XAML parser. XAML parser instantiates objects and set their properties. XAML describes objects, properties and there relation in between them. Using XAML, user can create any kind of objects that means graphical or non-graphical. WPF parses the XAML document and instantiates the objects and creates the relation as defined by XAML. In other words XAML is a XML document which defines objects and properties and WPF loads this document in actual memory.

II. ARCHITECTURE OF WPF

All WPF applications start with two threads, one for managing the UI and another background thread for handling the rendering and repainting. Rendering and repainting is managed by WPF itself. Fig 1. Shows the architecture of WPF.

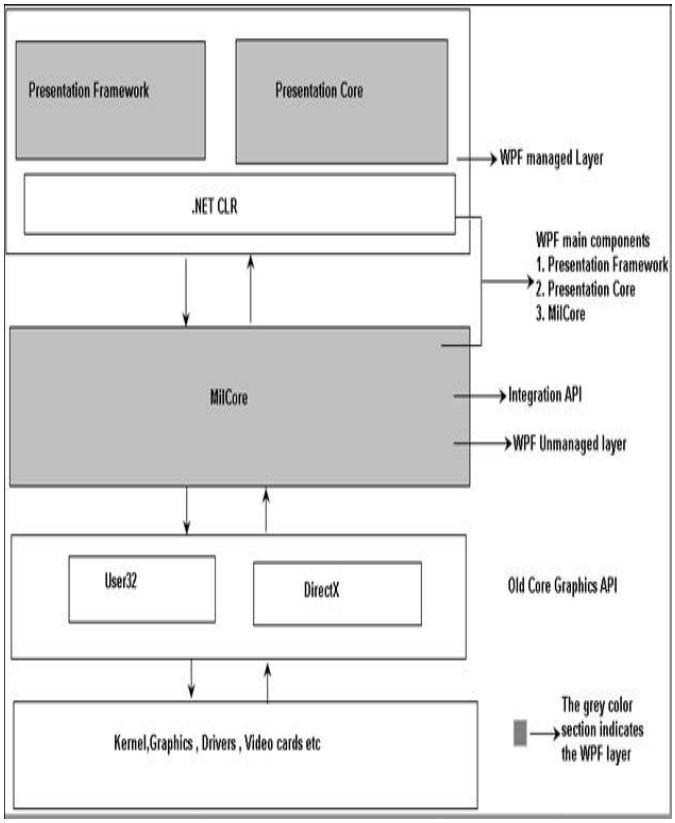

Fig. 1 Architecture of WPF

Above figure shows the overall architecture of WPF. It has three major sections Presentation core, Presentation framework and Media Integration layer (milcore). Milcore is written in unmanaged code in order to enable tight integration with DirectX. DirectX engine is responsible for all display in WPF, allowing for efficient hardware and software rendering.

Managed Layer: - The public API exposed is only via this layer. Majority of WPF is in managed code.

Presentation Framework:- It holds the top level WPF elements, including those that represents windows, controls, panels, styles etc. It also implements the end user presentation feature including time dependent, story based animations and data binding.

Presentation Core: Presentation Core provides a managed wrapper for MIL and implements the core services for WPF such as UI Element and visual from which all shapes and controls derived in Presentation Framework.

Windows Base:- Windows Base hold more basic elements which are capable to be reused outside the WPF environment like Dispatcher objects and Dependency object.

Unmanaged Layer:-This layer mainly consist of milCore and windows code.

milCore:- The composition engine is the native component of WPF application. It is called as Media Integration Layer. The purpose of the milCore is to interface directly with DirecX and provide basic support for 2D and 3D surface.It is interface between DirectX and CLR.

Windows Codes: Windows code is low level API which is used for imaging support in WPF applications like image processing, scaling etc. It comprises of a number of codes which encode/decodes images into vector graphics that would be rendered into WPF screen

Core Operating System Layer

User32:- It decides which goes where on the screen. It is the primary core API which every application uses. User32 actually manages memory and process separation.

DirectX: - As said previously WPF uses directX internally. DirectX talks with drivers and renders the content. DirectX is the low level API through which WPF renders all graphics.

III. WPF PROGRAMMING MODEL

XAML only:-In this model user can load raw XAML "pages" into Internet Explorer. No design time compilation required. Limited set of functionality is available. It supports static content, data binding, and animations. No code behind or script allowed in XAML only. Real applications require event handling, data access, calling services, accessing files, etc., which require programmatic code. Also user can embed C# or VB code in script blocks within XAML. It must compile into assembly. It provides Lose IntelliSense, colour coding, and compile time checking.

Code Only:-In this model WPF objects are all defined as .NET classes. Users can instantiate using a programming model similar to Windows Forms. It is instructive for understanding WPF, not practical for real world development. It is not productive. In this model design tools will all be based on XAML mark-up. Element hierarchy can be more compactly and cleanly represented in XAML

XAML and Code :- XAML very expressive for static layout of UI and initial configuration of properties. It is human readable. It is easier for tools to parse. Programmatic code needed for dynamic behaviour of applications.

XAML and Code Separation

The most important features of WPF, separating the XAML from the code to be handled. So designers can independently work on the presentation of the application and developers can actually write the code logic independent of how the presentation is.

MainWindow.xaml

```
<Window x:Class="DemoPro.MainWindow"  (1)
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="MainWindow" Height="350" Width="525">
  <Grid>
    <Button Content="Click Me" Margin="88,45,0,0"
Name="button1" VerticalAlignment="Top" Width="75"
        Click="button1_Click" />
  </Grid>
</Window>                    (2)
```

Fig 2 XMAL code

MainWindow.cs

```
public partial class MainWindow : Window (1)
{
  public MainWindow()
  {
    InitializeComponent();
  }
            (2)
  private void button1_Click(object sender, RoutedEventArgs e)
  {
    MessageBox.Show("Button is Clicked");
  }
}
```

Fig 3 Code behind File

Above is the code snippet, which shows a XAML file and the code completely detached from the XAML presentation.Number 1 indicates same class name i.e. MainWindow and number 2 indicates same method name i.e. button1_Click. In order to associate a class with XAML file user need to specify the x: Class attribute. Any event specified on the XAML object can be connected by defining a method with sender and event values. User can see from the above code snippet we have linked the MyClickEvent to an event in the behind code.

To access XAML objects in behind code user just need to define them with the same name as given in the XAML document. For instance in the below code snippet we named the object as objtext and the object is defined with the same name in the behind code.

```
<Canvas MouseEnter="ChangeValue" MouseLeave="ChangeValue1">

  <TextBlock Name="objText">Move your mouse on this </TextBlock>

</Canvas>
```

```
public partial class Window1 : Window
{
    public Window1()
    {
        InitializeComponent();
        objText = new TextBlock();
    }

    private void ChangeValue(object sender, EventArgs e)
    {
        objText.Text = "I am trying to learn XAML";
    }

    private void ChangeValue1(object sender, EventArgs e)
    {
        objText.Text = "XAML is easy to learn";
    }
}
```

Accessing the object by name

(1) (2) (3)

Fig 4 Accessing XAML object

IV. WPF and Other Display Technologies

We already have User32, GDI, GDI+, DirectX but still we go for WPF because these display technologies have their own limitations, that limitations are overcome in WPF.

User32:- This provides the windows look and feel for buttons and textboxes and other UI elements. User32 lacked drawing capabilities.

GDI (Graphics device interface):- GDI not only provided drawing capabilities but also provided a high level of abstraction on the hardware display.In other words it encapsulates all complexities of hardware in the GDI API.

GDI+:- GDI+ which basically extends GDI and provides extra functionalities like jpg and PNG support, gradient shading and anti-aliasing. The biggest issue with GDI API was it did not use hardware acceleration and did not have animation and 3D support.

DirectX: - The issue of hardware acceleration is solved in DirectX. DirectX exploited hardware acceleration, had support for 3D, full colour graphics, media streaming facility and lot more. This API no matured when it comes to gaming industry.

WPF: - WPF stands on the top of directX you can not only build simple UI elements but also go one step further and develop special UI elements like Grid, Flow Document, and Ellipse. In other words WPF is a wrapper which is built over DirectX. Still for game development DirectX is used. But for slight animation WPF is the best option.

V. Hardware acceleration with WPF

Hardware acceleration is a process in which we use hardware to perform some functions rather than performing those functions using the software which is running in the CPU. WPF exploits hardware acceleration in a two tier manner.

WPF API first detects the level of hardware acceleration using parameters like RAM of video card, per pixel value etc. Depending on that it either uses Tier 0, Tier 1 or Tier 2 Rendering mode.

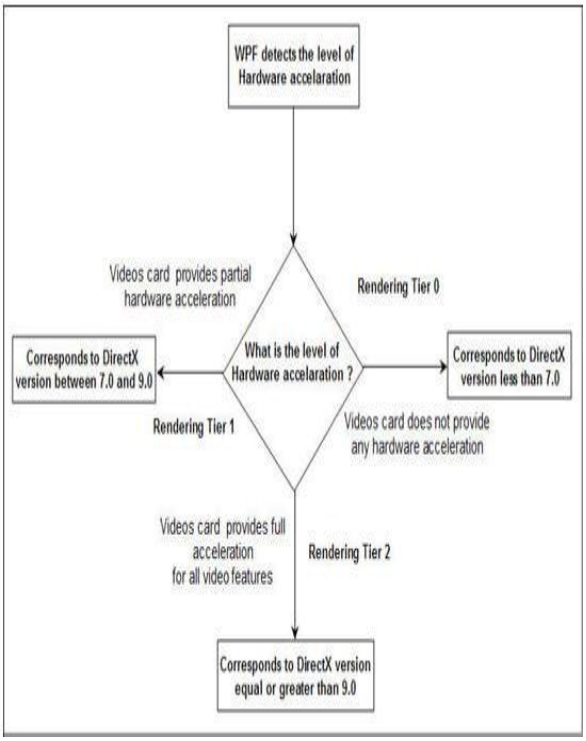

Fig. 5 Hardware acceleration in WPF

Tier 0:- If the video card does not support hardware acceleration then WPF uses Tier 0 rendering mode. In other words it uses software acceleration. This corresponds to working of DirectX version less than 7.0.

Tier 1:- If the video card supports partial hardware acceleration then WPF uses Tier 1 rendering mode. This corresponds to working of DirectX version between 7.0 And 9.0.

Tier 2:- If the video card supports hardware acceleration then WPF uses Tier 2 rendering mode. This corresponds to working of DirectX version equal or greater than 9.0.

VI.CONCLUSION

Paper describes the overall architecture of WPF. How code is separated from XAML. Programming model shows separation of code behind and XAML, which is very useful for the WPF developers. Paper shows how WPF is better than other display technologies.WPF has the ability to make very rich UIs. Using WPF to achieve animation and special effects are easier. Hardware acceleration is better as compared to other display technologies.

VII.REFERENCES

[1]. G Trygve M. H. Reenskaug, "MVC XEROX PARC1978-79"(undated). http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

[2]. Trygve Reenskaug - MODELS - VIEWS – CONTROLLERS. http://heim.ifi.uio.no/~trygver/1979/mvc-2/1979-12-MVC.pdf

[3]. Introduction to WPF .NET Framework 4. http://msdn.microsoft.com/en-us/library/aa970268.aspx

[4]. XAML in WPF. http://msdn.microsoft.com/en-us/library/ms747122.aspx

[5]. Introduction to Model/View/ViewModel pattern for building WPF apps – John Gossman. http://blogs.msdn.com/b/johngossman/archive/2005/10/08/478683.aspx

[6]. Steve Burbeck, "Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)", March 4, 1997. http://st-www.cs.illinois.edu/users/smarch/stdocs/mvc.html

[7] Mike Potel, "MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java". TaligentInc1996. http://www.wildcrest.com/Potel/Portfolio/mvp.pdf

[8].Microsoft Corporation. "Model View Presentter Pattern" (undated). http://msdn.microsoft.com/en-us/library/cc304760.aspx

[9]. Microsoft Corporation, "Introducing Windows Presentation Foundation" (undated). http://msdn.microsoft.com/enus/library/aa663364.aspx

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

[11]. Arlen Feldman, Maxx Daymon, "WPF in Action with Visual Studio 2008". Manning Publications Co, 2009, pp 283-285.

[12]. Flexible GUI in Robotics Applications Using Windows Presentation Foundation Framework and Model View ViewModel Pattern - Fran Jarnjak, IEEE 2010

[13]. Ergonomics Research and CNC machine tools in the interface design of the application - IEEE 2008