course: COMP6215 week: 09 lesson: 02 topic: introduction-to-uwp

DOWNLOAD PDF CLICK HERE

UWP - Universal Windows Platform

UWP was introduced with the release of Windows 10 and allows you to build the graphical interfaces for applications in the Windows Eco system.

These are application anywhere from Windows 10 IOT to Windows Surface Hub devices and anywhere in between. This includes Mix reality devices like the the Hololens

UWP uses the XAML syntax, but has some changes and additions to the language to suit the operating system since its use in the WPF framework.

Layout Strategies

UWP introduces a number of new layouts, but also the ability to change the layouts based on the application size.

There are a couple of terms we need to get familiar with:

- Liquid Design
- Adaptive Design
- Responsive Design

Liquid Design

An application that has liquid design build into it allows for elements to resize based on the applications frame. The layout does not change, but for example the size of an image might become bigger as the space allows for it.

Adaptive Design

An application that conforms to adaptive design allows for an application to change layout types when it hits certain markers - the UI elements do not change in the in between stages.

Responsive design

Responsive design is in effect the use of both liquid and adaptive design, meaning that it both resizes the UI elements and changes the layout of the application at certain points.

The different layout strategies can be found here

A more detailed look at the layouts are here

Visual State manager

Since UWP allows for adaptive design, there are some new tags to know about. To be able to implement these, we need to look at Visual Statemanagers.

- `<VisualStateManager.VisualStateGroups>`
- `<VisualStateGroup>`
- `<VisualState>`
- `<VisualState.StateTriggers>`
- `<AdaptiveTrigger>`
- `<VisualState.Setters>`
- `<Setter>` which makes use of the `Target` and `Value` attribute

The Visual state lives inside the main UI tag - by default a UWP app comes with a grid, so it is placed inside of the Grid.

For the most part, you can create a little code snippet to start you off:

```
<VisualStateManager.VisualStateGroups>
    <VisualStateGroup>
        <VisualState>
            <VisualState.StateTriggers>
                <AdaptiveTrigger MinWindowWidth="100" />
            </VisualState.StateTriggers>
            <VisualState.Setters>

              <!-- <Setter> Tags go here -->

            </VisualState.Setters>
        </VisualState>
    </VisualStateGroup>
</VisualStateManager.VisualStateGroups>
```

The code behind

Just like with WPF, UWP also has a code behind file which we will use for some things.

One thing that will need to be put into the constructor of the page is the option to control the minimum and start size of the application.

```
ApplicationView.PreferredLaunchViewSize = new Size(300, 300);
ApplicationView.PreferredLaunchWindowingMode =
ApplicationViewWindowingMode.PreferredLaunchViewSize;
ApplicationView.GetForCurrentView().SetPreferredMinSize(new Size(100,
100));
```

In additon to the code behind file we are able to use other classes and use a paradigm called MVVM - which we will look at later.

What is the same?

So outside of some of the tag names that come with the rename of classes that are used and the visual state manager there are very little changes syntactically.

The UI that is produced however is quite a bit different. Over the next view weeks we will look at split navigation and using MVVM.

The future of UWP

UWP will continue to be the GUI framework for Windows 10 application, but it only runs on the Windows 10 eco system.
A lot of legacy applications are still running on WPF and Windows Forms and those are still supported by Microsoft.

Electron is an open source platform owned by Github which is owned by Microsoft allows you to create desktop application using web technologies.

All in all there a lot of options for you to choose from - so UWP is still growing in the market....