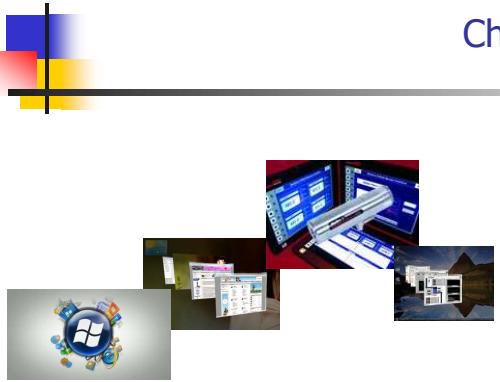


## IG sous DotNet

Ch C - WPF - UWP - .Net Core



B. Chervy  
2023

## Sommaire



1. - Généralités sur WPF
2. - Les contrôles WPF et le XAML
4. - Application Universelle UWP
5. - .Net Core et WinUI

## 1. - Windows Presentation Foundation



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

3

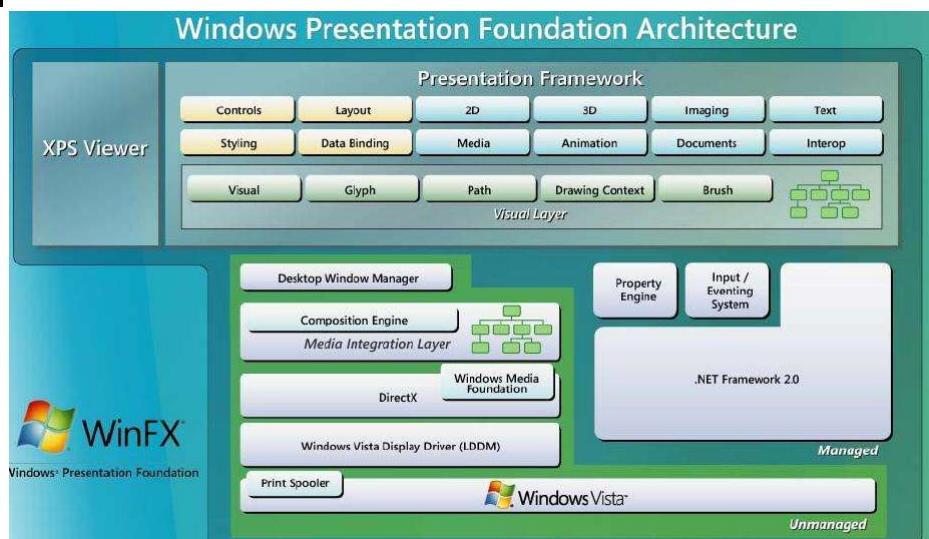
## Windows Presentation Foundation

- **Windows Presentation Foundation (WPF) :**
  - ⇒ nouveau système d'affichage graphique et de création d'interfaces graphiques de Microsoft Windows intégré depuis la version 3.0 de .Net (WinFX)
  - ⇒ **API entièrement managées** pour la fabrication **d'interfaces utilisateurs**
    - ⇒ vu comme une surcouche logicielle à DirectX, entièrement vectorielle pour le dessin comme pour le texte
    - ⇒ fournit également un environnement d'exécution pour les pages web (Silverlight)
    - ⇒ permet de séparer la conception des ihm du code logique grâce à l'introduction du **XAML** (eXtensible Application Markup Language) pour la description de l'interface
    - ⇒ prend directement en charge les fonctionnalités de lecture de média (vidéo ou son)
    - ⇒ permet des fonctionnalités d'interface modernes comme la transparence et les dégradés (moteur de rendu vectoriel et indépendant de toute résolution, créé pour tirer parti du matériel graphique moderne)
  - Cœur de WPF : un moteur graphiques 2D et 3D + un jeu complet de fonctionnalités de développement d'applications (animation, styles, modèles, documents, médias, texte et typographie, ...)

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

4

## WPF et .Net 3.0 (WinFx)



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

5

## Windows Presentation Foundation

- **WPF** repose sur 4 grands axes :
  1. Une **approche unifiée des GUI**, des documents et des animations : devient un Framework de présentation unifié, qui intègre et gère toutes les parties d'une interface utilisateur (animations, images, contrôles, etc..)
  2. Un **moteur de composition** basé sur des vecteurs intégrés : moteur d'exécution (Runtime) de composition afin d'expédier les requêtes de rendu de l'interface graphique au bon composant logiciel ou matériel (APIs Windows, carte vidéo compatible DirectX)  
⇒ utilisation **DirectX** pour vous fournir un affichage vectoriel
  3. Un **modèle de programmation déclarative** : apporte la puissance de la programmation déclarative (le **XAML**)  
⇒ séparation distincte entre le designer et le développeur
  4. Une **méthode de déploiement** : gestion des applications de manière facile et sécurisée

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

6

## L'interface XAML

- **.Net** (WPF/Silverlight/UWP) intègre le langage descriptif **XAML** qui permet de l'utiliser en développement d'une manière proche d'une page HTML
  - ⇒ séparation entre les données et leur présentation
  - ⇒ le graphisme repose sur un nouveau moteur de rendu vectoriel basé sur DirectX (meilleure qualité finale) qui interprète le XAML
  - ⇒ mapping des événements XAML/Code sous-jacent (appel dans le code XAML et définition dans le code C#, VB.NET, etc...)



⇒ Se retrouve dans les interfaces .Net de type ModernUI (Windows 8 et RT), Silverlight, Windows Phone et Mobile, UWP

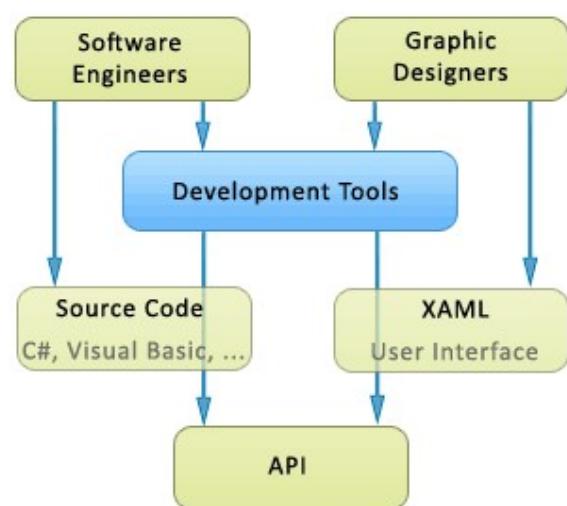
© 3IL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3IL

7

## WPF

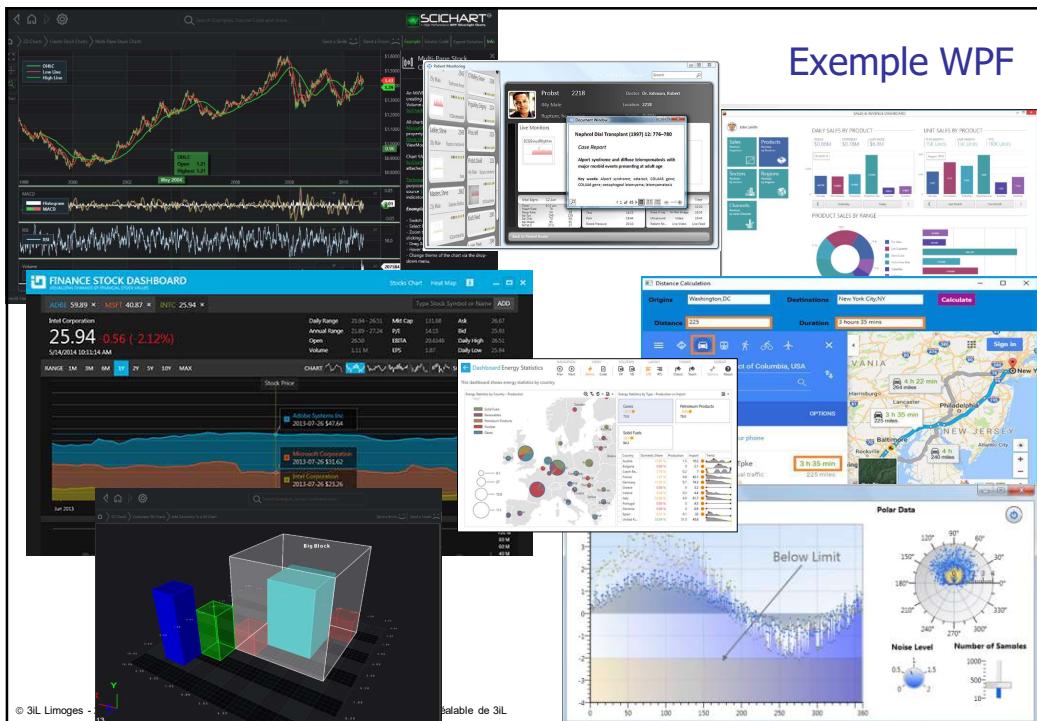
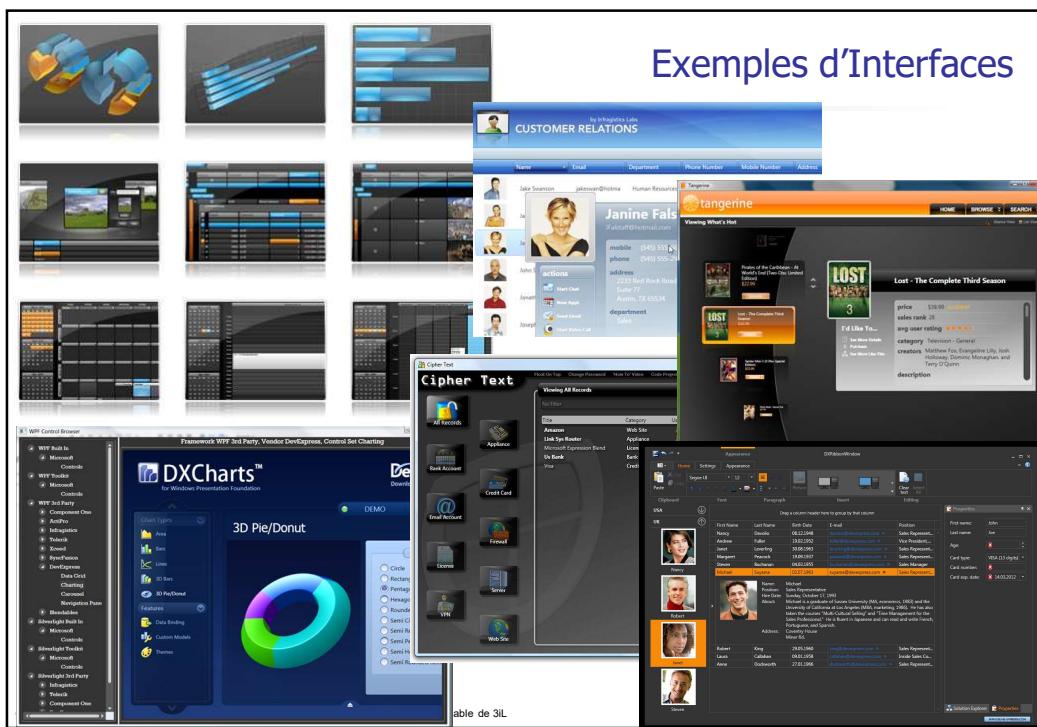
### Principe de base :

Séparation entre  
- **traitement des données**  
(fonctionnalités)  
et  
- **représentation des données**



© 3IL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3IL

8





## Autres fonctionnalités

- ⇒ séparation entre les données et leur présentation
- ⇒ gère les **bases de données** pour les applications ou le web et des modèles de présentation
- ⇒ fournit différents moyens de créer des **composants propres** :  
par agrégation (UserControl) ou dérivation de composants existants (CustomControl)
- ⇒ optimise le concept de virtualisation des applications prises en main à distance grâce à la diminution des informations transitant sur le réseau
- ⇒ permet l'installation d'application indépendante par installateur comme Windows Installer (package msi) ou clickOnce
- ⇒ intègre la gestion du **Multitouch** (WPF4) : technologie intégrée de manière native et applicable à tous les éléments graphiques grâce à une API de gestion du tactile



## XAML

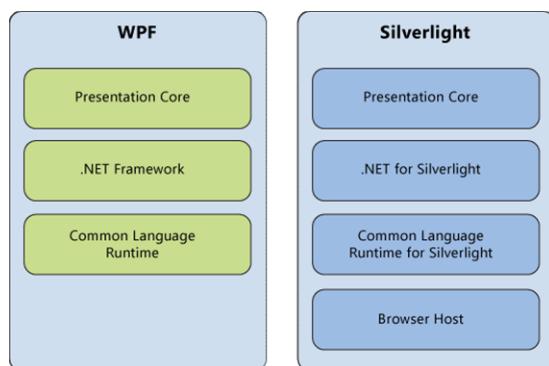
- **XAML** (*eXtensible Application Markup Language*)
  - ⇒ modèle de programmation déclarative **.NET** basé sur le **XML**
  - ⇒ fonctionne sur le principe de la **sérialisation** de graphe d'objets
  - ⇒ comme beaucoup de langage .Net, on y retrouve :
    - des espaces de nom (*namespaces* en anglais),
    - des classes,
    - des propriétés,
    - un mapping des événements (appel dans le code XAML et définition dans le code C#, VB.NET, etc...),
    - etc...
  - ⇒ c'est un langage de description **qui n'est pas dédié à WPF !**

```
<Window x:Class="DemosWPF.Window1"
xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation
xmlns:x=http://schemas.microsoft.com/winfx/2006/xaml
Title="DemosWPF" Height="768" Width="1024"
WindowStartupLocation="CenterScreen"
>
```

## Applications WPF

### WPF = 2 types d'applications :

- **applications "stand-alone"** :  
(installées sur la machine cliente)  
installateur type Windows-  
Installer(msi) ou clickOnce



- **applications "navigateur"** :  
téléchargées depuis Internet et  
qui s'exécutent au travers d'un navigateur Web sur la version pour navigateur de  
la CLR : **Silverlight**  
⇒ application XBAP : XAML Browser Applications  
⇒ IE depuis la version 3.0 de .NET, Mozilla Firefox depuis la version 3.5

© 3IL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3IL

13

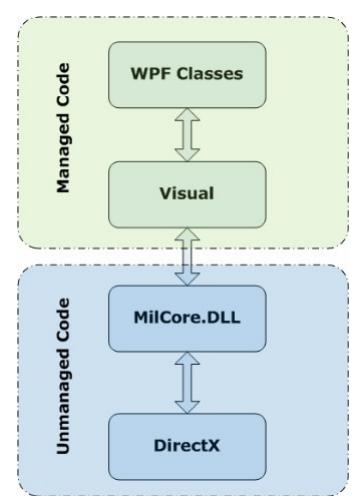
## WPF et le Graphisme

- **Graphisme de WPF** :
  - ⇒ remplace les Windows Forms (GDI+) et la couche graphique Win32 (GDI)
  - ⇒ repose sur un nouveau moteur de rendu: le **Graphics Processing Unit** (moteur permettant un rendu vectoriel ⇒ meilleure qualité finale)
  - ⇒ basé sur **DirectX** pour profiter de l'accélération matérielle
  - ⇒ entièrement vectoriel pour le dessin comme pour le texte
  - ⇒ permet d'augmenter la taille des objets en fonction de la résolution de l'écran sans effet de pixellisation
  - ⇒ fournit tous les éléments d'interface graphique nécessaires : "widgets", fenêtres, boutons, champs de texte, menus, listes, ...
  - ⇒ affichage du texte améliorant le lissage des caractères (procédés ClearType, TrueType ou OpenType)
  - ⇒ supporte l'affichage de nombreux formats d'images ou vidéos ( MPEG, AVI, WMV, ...)
  - ⇒ permet toujours de générer dynamiquement des interfaces en code managé

© 3IL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3IL

14

## Architecture WPF



⇒ N'utilise ni GDI ni GDI+  
 ⇒ **Surcouche logicielle à DirectX** : l'affichage s'effectue via le moteur DirectX pour le rendu matériel et l'efficacité logicielle

⇒ **Visual** : assure la prise en charge du rendu, les tests de positionnement, les transformations de coordonnées et les calculs de zones

⇒ **Milcore** : moteur de composition non managé

⇒ **CLR** : assure la gestion de la mémoire, la gestion des erreurs, le système de type commun, ...

PresentationFramework

PresentationCore

Common Language Runtime

milcore

User32

DirectX

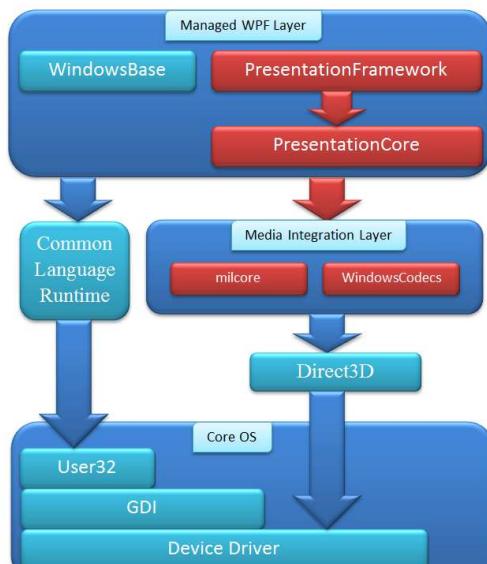
Kernel

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

15

## Architecture WPF

- Un message User32 est converti en message d'entrée brut WPF puis envoyé à un répartiteur
  - ⇒ WPF permet de convertir les événements d'entrée bruts en différents événements réels
  - ⇒ Chaque événement est converti au moins en **deux événements** : un événement « aperçu » et l'événement classique

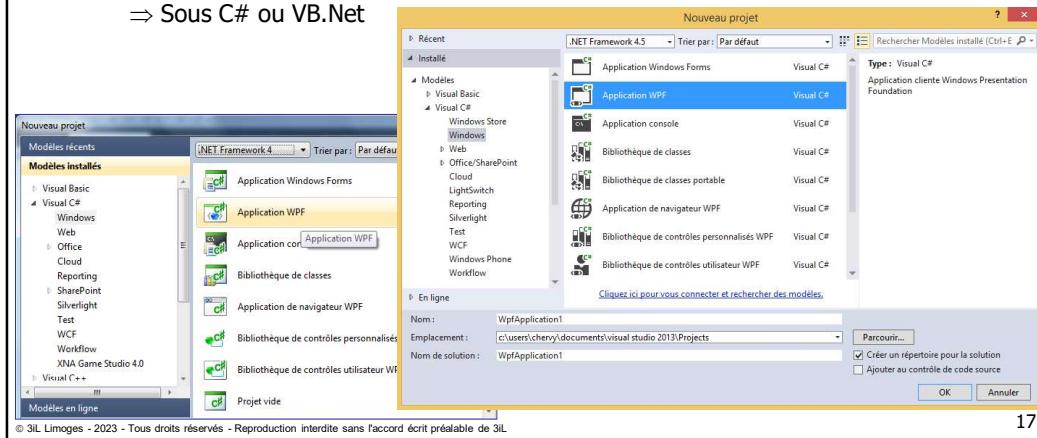


© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

16

## WPF et Visual Studio

- WPF Application : application WPF standard
  - WPF Browser Application : application WPF s'exécutant dans un navigateur Web
  - WPF User Control Library : bibliothèque de contrôles
  - WPF Custom Control Library : contrôle WPF existant personnalisé (en étendant ses fonctionnalités)
- ⇒ Sous C# ou VB.Net



17

## 2. - Les contrôles WPF et le XAML

## Les contrôles WPF



**Tous les contrôles WPF**

- Pointeur
- Border
- Button
- Calendar
- Canvas
- CheckBox
- ComboBox
- ContentControl
- DatePicker
- DockPanel
- DocumentViewer
- Ellipse
- Expander
- Frame
- Grid
- GridSplitter
- GroupBox
- Image
- Label
- ListBox
- ListView
- MediaElement
- Menu
- >PasswordBox
- ProgressBar
- RadioButton
- Rectangle
- RichTextBox
- ScrollBar
- ScrollView
- Separator
- Slider
- StackPanel
- Statusbar
- TabControl
- TextBlock
- ToolBar
- ToolBarPanel
- ToolBarTray
- TreeView
- Viewbox
- WebBrowser
- WindowsFormsHost
- WrapPanel

**Les contrôles WPF**

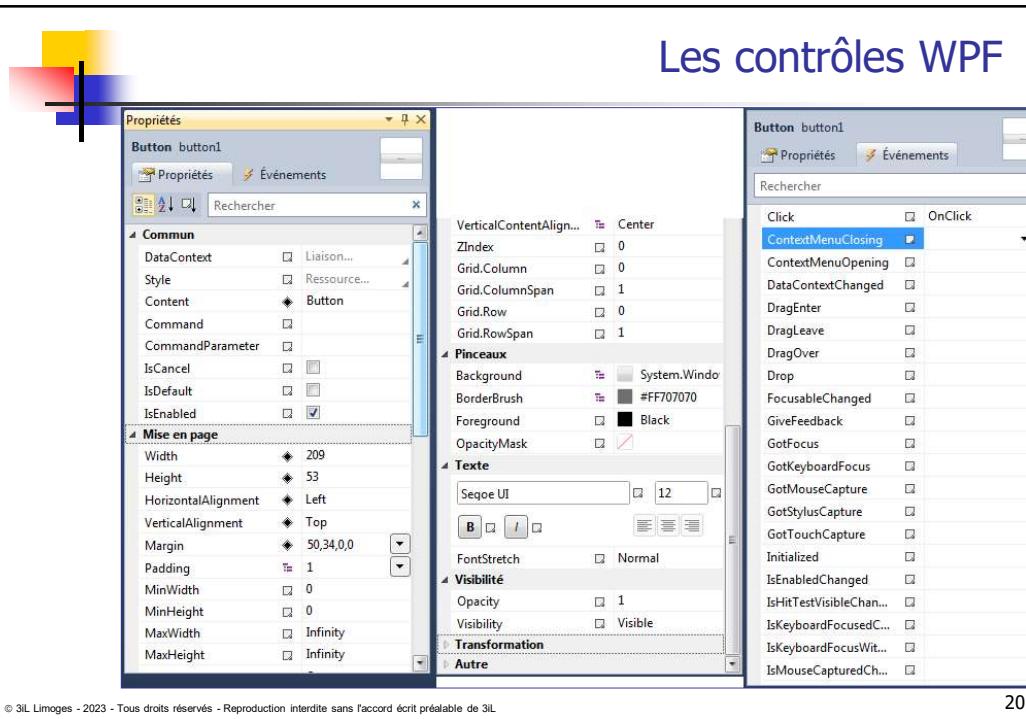
- ⇒ contrôles utilisés pour visualiser des données et pour permettre à l'utilisateur d'interagir avec l'application
- ⇒ possèdent des propriétés, des événements et des méthodes
- ⇒ contrôles standards dans **WPF** : les boutons, les labels, les textbox, etc...

**Mais les contrôles WPF :**

- ⇒ peuvent contenir n'importe quoi : possibilité de personnaliser complètement le rendu d'un contrôle **WPF**
- ⇒ sont des éléments fonctionnels auxquels on applique un style qui peut se combiner avec d'autres grâce au système de Template

19

## Les contrôles WPF



**Propriétés**

**Button button1**

**Propriétés**      **Événements**

**Rechercher**

**Common**

- DataContext
- Style
- Content
- Command
- CommandParameter
- IsCancel
- IsDefault
- IsEnabled

**Mise en page**

- Width
- Height
- HorizontalAlignment
- VerticalAlignment
- Margin
- Padding
- MinWidth
- MinHeight
- MaxWidth
- MaxHeight

**VerticalContentAlignment**: Center

**Pinceaux**

- Background
- BorderBrush
- Foreground
- OpacityMask

**Texte**

- FontFamily: Segoe UI
- FontSize: 12
- FontStyle
- FontStretch

**Visibilité**

- Opacity
- Visibility

**Transformation**

**Autre**

**Événements**

**Button button1**

**Propriétés**      **Événements**

**Rechercher**

**Click**:  **OnClick**

**ContextMenuClosing**:

**ContextMenuOpening**

**DataContextChanged**

**DragEnter**

**DragLeave**

**DragOver**

**Drop**

**FocusedChanged**

**GiveFeedback**

**GotFocus**

**GotKeyboardFocus**

**GotMouseCapture**

**GotStylusCapture**

**GotTouchCapture**

**Initialized**

**EnabledChanged**

**IsHitTestVisibleChanged**

**IsKeyboardFocusedChanged**

**IsKeyboardFocusWithinChanged**

**IsMouseCapturedChanged**

20

WPF

The screenshot shows the Visual Studio IDE with two open files: `MainWindow.xaml.cs` and `MainWindow.xaml`. The `MainWindow.xaml.cs` file contains C# code for the `button1_Click` event. The `MainWindow.xaml` file shows a simple window with a single button centered in a grid.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WpfApplication1
{
    /// <summary>
    /// Logique d'interaction pour MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
        }
    }
}

```

21

## Disposition des contrôles

- Différentes façons de disposer les contrôles dans une fenêtre :
  - ⇒ **Canvas** : organisation en coordonnées X et Y
  - ⇒ **Grid** : organisation tabulaire automatique en lignes et colonnes
  - ⇒ **StackPanel** : pile verticale ou horizontale automatique
  - ⇒ **DockPanel** : alignement automatique sur un côté

The screenshot shows the Visual Studio IDE with the `Window2.xaml.cs` file open. It displays a window with a button positioned in a `Grid` control. The `XAML` code defines the window's structure, including the `Grid` and its child `Button`.

```

<Window x:Class="WpfApplication1.Window2"
       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       Title="Window2" Height="300" Width="386">
    <Grid>
        <Button Content="Button" Height="51" HorizontalAlignment="Left" Margin="50,36,0,0" Name="button1" VerticalAlignment="Top" Width="120" />
    </Grid>
</Window>

```

22

## Ressources

- **Les ressources** : propriété de chaque composant qui permet de définir les ressources qui lui seront associées

Par ressources, on entend beaucoup de choses :

- ⇒ les **styles** (définissent l'apparence d'un contrôle)
- ⇒ les **templates** (définit l'affichage des données d'un contrôle)
- ⇒ les **animations**
- ⇒ les **transformations** (rotation, etc...)

⇒ permet d'avoir accès et de réutiliser des objets définis pour l'ensemble d'une application (style ou template à appliquer communs à la fenêtre)

⇒ centralisation de styles d'objets

⇒ assimilées aux fichiers **CSS**

⇒ style (ou template) spécifique à chaque type de contrôle peut quand même s'appliquer lors de leur déclaration

⇒ mot clé : **StaticResource**

## XAML

- **Les styles** : permettent de définir l'apparence de vos contrôles

⇒ l'attribut **Key** permet d'identifier le style

⇒ l'attribut **TargetType** permet d'indiquer le type de contrôle concerné

```
<Style TargetType="{x:Type Button}" x:Key="MaPolice">
    <Setter Property="Button.FontStyle" Value="Italic" />
    <Setter Property="Button.FontWeight" Value="Bold" />
</Style>
```

...

```
<Button Style="{StaticResource MaPolice}" Content="..." Name="button1" Width="222" Click="button1_Click" />
```

- **Les templates** : utilisés pour décrire la structure visuelle d'un contrôle

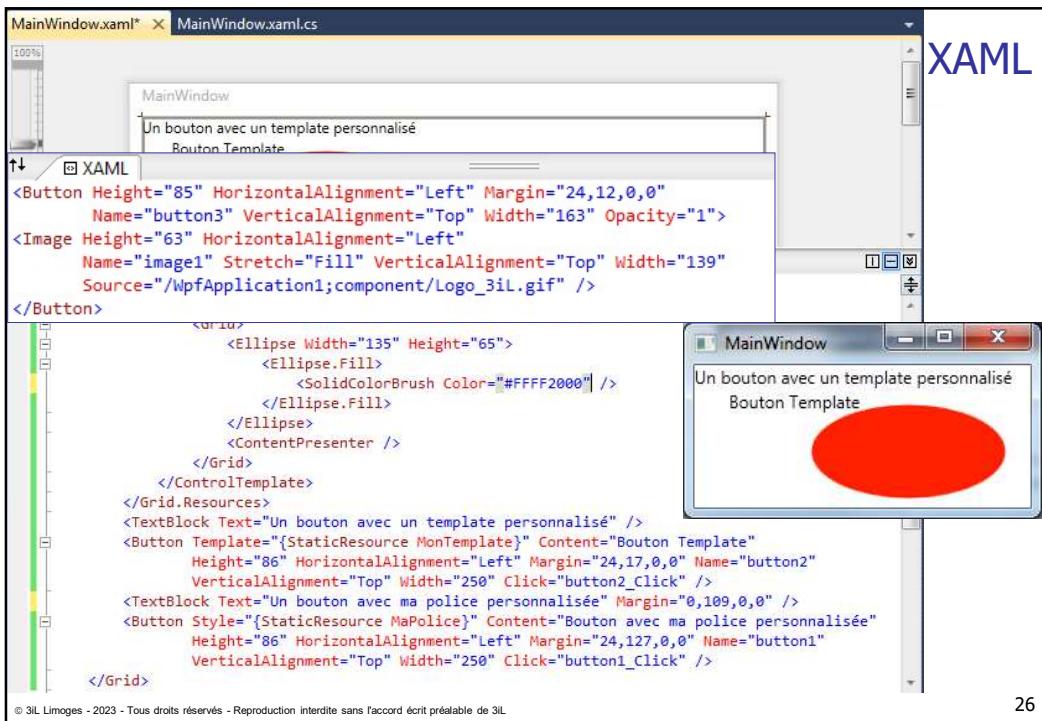
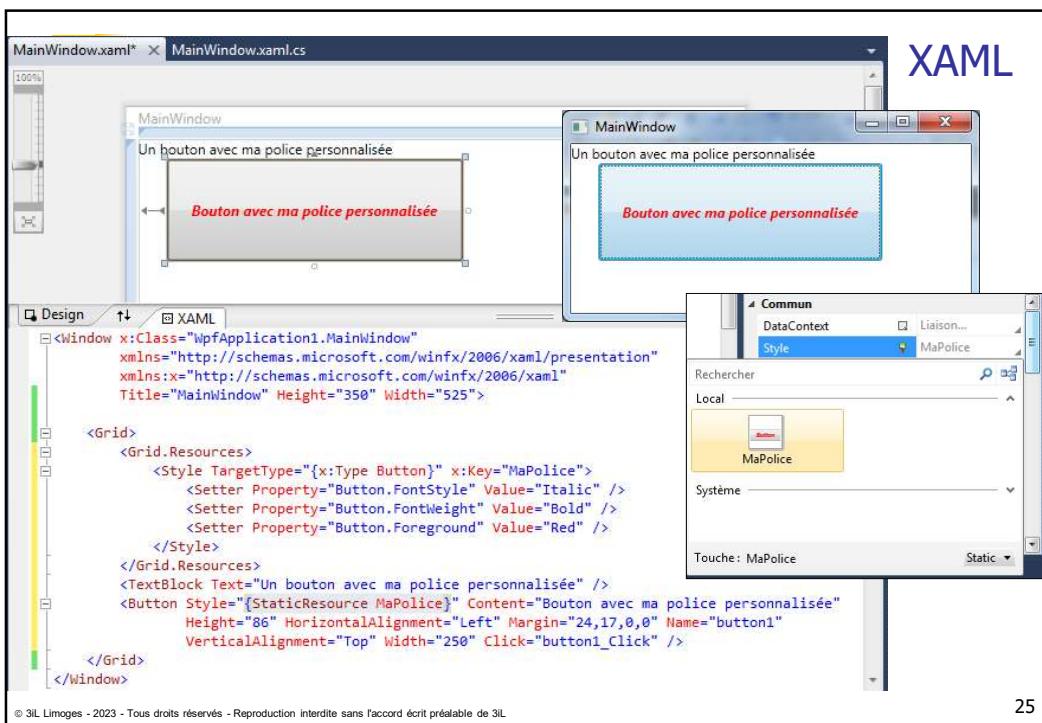
propriété **Template** : permet d'associer un template défini à un contrôle

```
<ControlTemplate TargetType="{x:Type Button}" x:Key="MonTemplate">
```

...

```
<Button Template="{StaticResource MonTemplate}" Content="..." .....
```

(⇒ Séparation entre l'arbre visuel et l'arbre logique)



## XAML

```
<Button x:Name="button" HorizontalAlignment="Left" Height="100"
        Margin="36,27,0,0" VerticalAlignment="Top" Width="245">
    <Image x:Name="image" HorizontalAlignment="Left" Height="82"
           VerticalAlignment="Top" Width="269" Source="téléchargement.png" />
</Button>
```

```
<Button x:Name="button" HorizontalAlignment="Left" Height="100"
        Margin="36,36,0,0" VerticalAlignment="Top" Width="245">
    <Image x:Name="image" HorizontalAlignment="Left" Height="82"
           VerticalAlignment="Top" Width="269" Source="téléchargement.png" />
</Button>
<Calendar HorizontalAlignment="Left" Margin="168,10,0,0" VerticalAlignment="Top" Opacity="0.5"/>
```

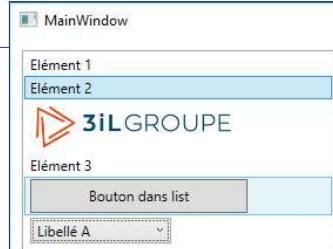


© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

27

## Exemple avec une listBox

```
<ListBox x:Name="listBox" HorizontalAlignment="Left" Height="173" VerticalAlignment="Top"
          Width="266" Margin="10,10,0,0">
    <ListBoxItem>Elément 1</ListBoxItem>
    <ListBoxItem>Elément 2</ListBoxItem>
    <ListBoxItem>
        <Image x:Name="image1" HorizontalAlignment="Left" Height="42"
               VerticalAlignment="Top" Width="211" Source="3ilGroupe.png" OpacityMask="#FF722323" />
    </ListBoxItem>
    <ListBoxItem>Elément 3</ListBoxItem>
    <Button x:Name="button" HorizontalAlignment="Left" Height="29" VerticalAlignment="Top"
            Width="185" Content="Bouton dans list"/>
    <ComboBox x:Name="comboBox" HorizontalAlignment="Left" Height="29" VerticalAlignment="Top"
              Width="120">
        <ComboBoxItem>Libellé A</ComboBoxItem>
    </ComboBox>
</ListBox>
```



Une listBox est aussi un contrôle conteneur

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

28

## Transformations XAML

- **Les transformations** : pour « translater » un objet (déplacement à partir d'un point de départ donné)  
**TranslateTransform, RotateTransform, ScaleTransform**  
**MatrixTransform, SkewTransform, TransformGroup**

```
<Button x:Name="button_Copy" HorizontalAlignment="Left" Height="60"
        VerticalAlignment="Top" Width="183" Margin="22,105,0,0">
    <Image x:Name="imageG3il" HorizontalAlignment="Left" Height="42"
           VerticalAlignment="Top" Width="211" Source="3ilGroupe.png" />
    <Button.RenderTransform>
        <ScaleTransform ScaleX="1" ScaleY="-1.5" />
    </Button.RenderTransform>
</Button>
```



```
<Button.RenderTransform>
    <RotateTransform CenterX="20" CenterY="0" Angle="60"/>
</Button.RenderTransform>
```



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

29

## Triggers XAML

- **Les triggers** : déclencheur permettant de réaliser des interactions riches et dynamiques à utiliser conjointement avec les styles et les templates  
Les Triggers sont activés lorsqu'une condition spécifique devient vraie
  - ⇒ « **Property Dependency** » (utilisation de Trigger)
  - ⇒ **propriété .NET** (utilisation de DataTrigger) : déclenché lorsque la propriété spécifiée est modifiée.
  - ⇒ **événement** (utilisation d'EventTrigger) : déclenchée lorsque l'événement indiqué survient (par exemple, le clic sur un bouton)

```
<Grid.Resources>
    <Style TargetType="{x:Type Button}" x:Key="MonStyle">
        <Setter Property="Button.FontStyle" Value="Italic" />
        <Setter Property="Button.FontWeight" Value="Bold" />
        <Setter Property="Button.Foreground" Value="Red" />
        <Style.Triggers>
            <Trigger Property="IsMouseOver" Value="True">
                <Setter Property="Button.Background" Value="Red" />
            </Trigger>
            <Trigger Property="IsFocused" Value="True">
                <Setter Property="Button.Foreground" Value="Green" />
            </Trigger>
        </Style.Triggers>
    </Style>
</Grid.Resources>
<Button Content="Coucou le bouton" Style="{StaticResource MonStyle}" Height="51"
       Margin="30,12,35,0" Name="button1" VerticalAlignment="Top" Click="OnClick" >
</Button>
```



Coucou le bouton

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

30

## Animations et StoryBoards

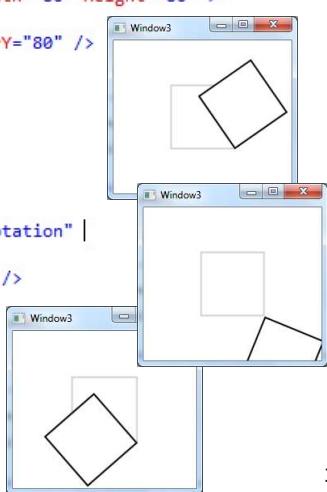
- **Les animations** : permettent d'animer les propriétés des éléments (contrôles) dont la classe correspond au type de la variable propriété
  - ColorAnimation** : pour animer la valeur d'une propriété de type Color
  - DoubleAnimation** : pour animer la valeur d'une propriété de type Double
  - ByteAnimation** : pour animer la valeur d'une propriété de type Byte
  - SpeedRatio** : permet de spécifier la vitesse d'exécution d'une animation
  - KeyFrames** : utilisés pour définir des « étapes » d'une animation (vu comme un point d'arrêt dans une animation)
- **Les storyboards** : éléments **XAML** permettant de définir un ensemble d'actions ⇔ ensemble **d'animations/transformations**
  - ⇒ permettent de définir le temps des animations, les propriétés à modifier sur les objets, etc....
  - ⇒ permettent un paramétrage complet des animations

© 3IL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3IL

31

## Animations et StoryBoards

```
<Grid>
    <Rectangle Stroke="LightGray" StrokeThickness="2" Fill="White" Width="80" Height="80" />
    <Rectangle Stroke="Black" StrokeThickness="2" Fill="White" Width="80" Height="80" >
        <Rectangle.RenderTransform>
            <RotateTransform x:Name="rotation" CenterX="80" CenterY="80" />
        </Rectangle.RenderTransform>
    </Rectangle>
    <Grid.Triggers>
        <EventTrigger RoutedEvent="Rectangle.MouseDown">
            <EventTrigger.Actions>
                <BeginStoryboard>
                    <Storyboard>
                        <DoubleAnimation Storyboard.TargetName="rotation" |
                            Storyboard.TargetProperty="Angle"
                            Duration="0:0:4" From="0" To="360" />
                    </Storyboard>
                </BeginStoryboard>
            </EventTrigger.Actions>
        </EventTrigger>
    </Grid.Triggers>
</Grid>
```



© 3IL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3IL

32

## Contenus

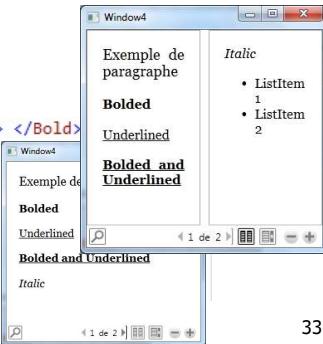
### ■ Flux Multimédia

L'objet MediaElement permet de diffuser un flux audio ou video

```
<MediaElement Width="100" Source="video.mpg" Volume="1"  
Margin="94,77,93,97"></MediaElement>
```

### ■ Documents

```
<FlowDocumentReader>  
  <FlowDocument>  
    <Paragraph FontSize="18">Exemple de paragraphe</Paragraph>  
    <Paragraph><Bold>Bolded</Bold></Paragraph>  
    <Paragraph><Underline>Underlined</Underline></Paragraph>  
    <Paragraph><Bold><Underline>Bolded and Underlined</Underline></Bold></Paragraph>  
    <Paragraph><Italic>Italic</Italic></Paragraph>  
    <List>  
      <ListItem><Paragraph>ListItem 1</Paragraph></ListItem>  
      <ListItem><Paragraph>ListItem 2</Paragraph></ListItem>  
    </List>  
  </FlowDocument>  
</FlowDocumentReader>
```



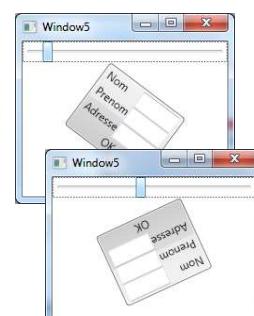
© 3IL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3IL

33

## DataBinding entre contrôles

Possibilité de faire du **databinding** entre différents contrôles de l'application  
⇒ liaison et interaction XAML entre contrôles

```
<StackPanel>  
  <Slider Minimum="0" Maximum="360" Name="sldValue"/>  
  <Button HorizontalAlignment="Stretch" Width="100">  
    <Button.LayoutTransform>  
      <RotateTransform Angle="{Binding ElementName=sldValue, Path=Value}"/>  
    </Button.LayoutTransform>  
    <Grid>  
      <Grid.ColumnDefinitions>  
        <ColumnDefinition/> <ColumnDefinition/>  
      </Grid.ColumnDefinitions>  
      <Grid.RowDefinitions>  
        <RowDefinition/> <RowDefinition/>  
      </Grid.RowDefinitions>  
      <TextBlock Grid.Column="0" Grid.Row ="0">Nom</TextBlock>  
      <TextBox Width="100" VerticalAlignment="Top" Height="20" Grid.Column="1" Grid.Row ="0"/>  
      <TextBlock Grid.Column="0" Grid.Row ="1">Prenom</TextBlock>  
      <TextBox Width="100" VerticalAlignment="Top" Height="20" Grid.Column="1" Grid.Row ="1"/>  
      <TextBlock Grid.Column="0" Grid.Row ="2">Adresse</TextBlock>  
      <TextBox Width="100" VerticalAlignment="Top" Height="20" Grid.Column="1" Grid.Row ="2"/>  
      <TextBlock HorizontalAlignment="Center" VerticalAlignment="Center" Grid.Column="0" Grid.Row  
      </Grid>  
    </Button>  
</StackPanel>
```



© 3IL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3IL

34

## XAML et la 3D

### ■ De nouvelles notions avec de nouveaux objets :

- ⇒ **Viewport3D** : objet fonctionnant comme une fenêtre classique mais en 3D
- ⇒ classe **Camera** : objet qui permet de spécifier le point de vue de celui qui regarde la scène 3D
  - ⇒ Se définit par sa position dans le plan 3D, sa direction, son champ de vision (angle), et un vecteur qui définit le haut de la scène
  - ⇒ **Type PerspectiveCamera** : permet de définir une caméra qui se projette autour de la scène
  - ⇒ **OrthograficCamera** : permet de définir une projection orthogonale d'un modèle 3D, vers une surface visuelle en deux dimensions
- ⇒ objets de la classe **Ligth** : pour illuminer une scène (DirectionalLigth, ...)
- ⇒ **ScreenSpaceLine3D** : objet permettant de créer des polygones 3D
- ⇒ transformations 3D : TranslateTransform3D, RotateTransform3D, ...

© 3IL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3IL

35

## 3 - Application Universelle UWP



© 3IL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3IL

36

## Application Windows Universelle

- Windows 10 a introduit la **plateforme Windows universelle (UWP)**
  - ⇒ modèle de Windows Runtime intégrée dans le noyau unifié Windows 10
  - ⇒ offre désormais une **plateforme d'application commune** disponible sur chaque appareil exécutant **Windows 10**
  - ⇒ applications ciblant l'UWP : peuvent appeler les **API WinRT** communes à tous les appareils, mais aussi des API (notamment des API Win32 et .NET) spécifiques de la famille d'appareils sur lesquels l'application s'exécute

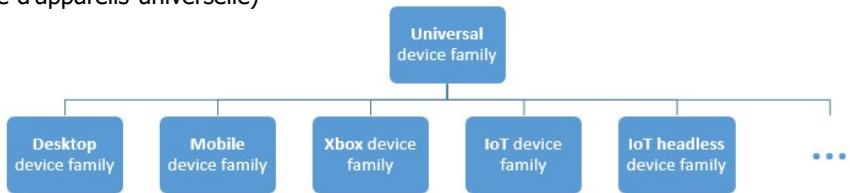


© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

37

## Application Windows Universelle

- application ne cible plus un système d'exploitation, mais une ou plusieurs **familles d'appareils**
  - ⇒ famille d'appareils
  - identifie les API, les caractéristiques système et les comportements attendus sur les différents appareils de cette famille.
  - détermine l'ensemble des appareils sur lesquels une application peut être installée à partir du Store
    - ⇒ choix d'une famille d'appareils : surface d'API qui peut être appelée de façon inconditionnelle par l'application + nombre d'appareils supportant l'application
    - ⇒ application UWP cible spécifiquement la famille d'appareils universelle et sera disponible pour tous les appareils (supposer la présence uniquement des API de la famille d'appareils universelle)

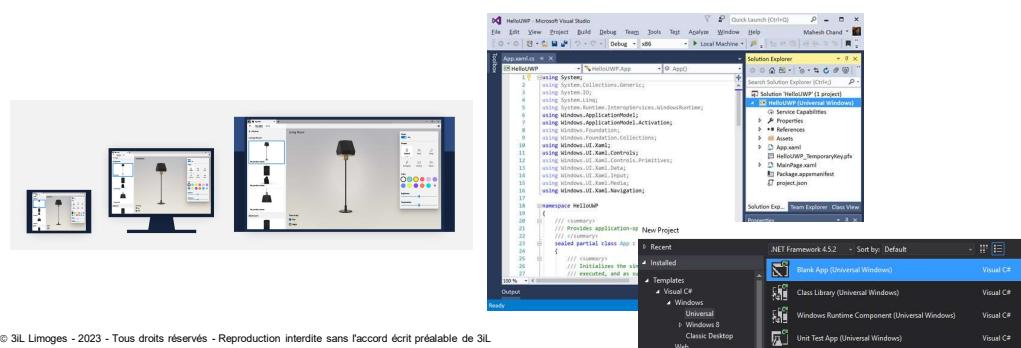


© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

38

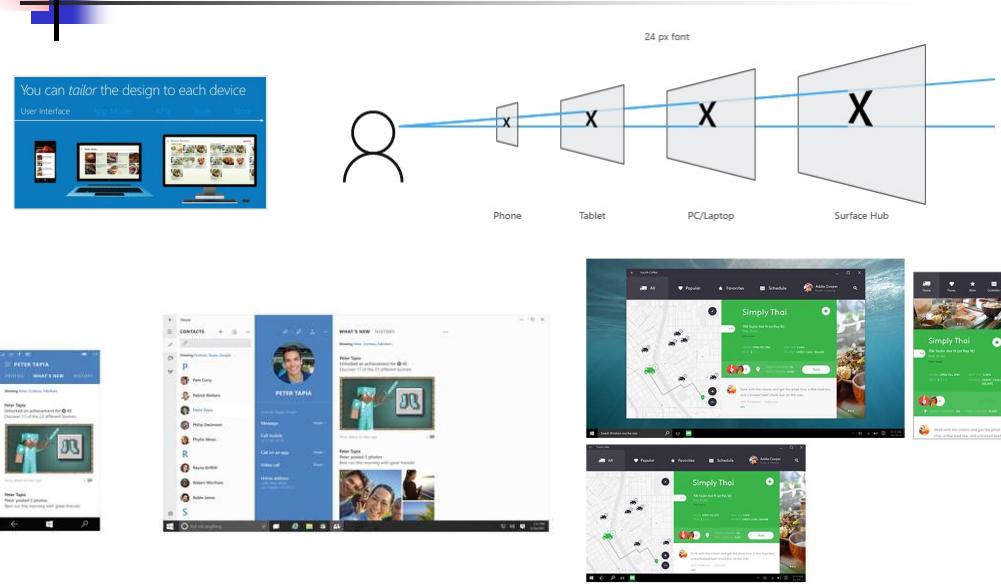
## UWP

- **Universal Windows Platform (UWP)** : plateforme d'application universelle Windows
  - ⇒ architecture homogène pour développer des applications universelles qui fonctionnent sur plusieurs types de périphériques et sous Windows 10, Windows 10 Mobile, Xbox One et Hololens sans besoin de réécrire un nouveau code source pour chacun de ces systèmes
  - ⇒ prend en charge C++, C#, VB.NET et XAML, F# et JavaScript.
  - ⇒ extension de la plate-forme Windows Runtime introduite pour la première fois dans Windows Server 2012 et Windows 8



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

## Application Windows Universelle



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

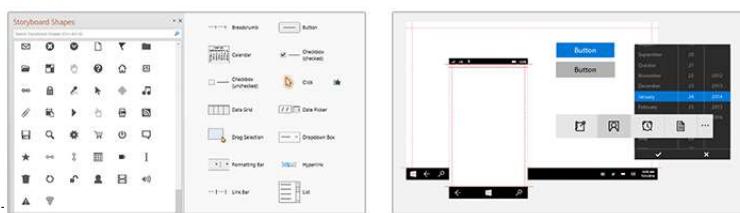
40

## Application Windows Universelle

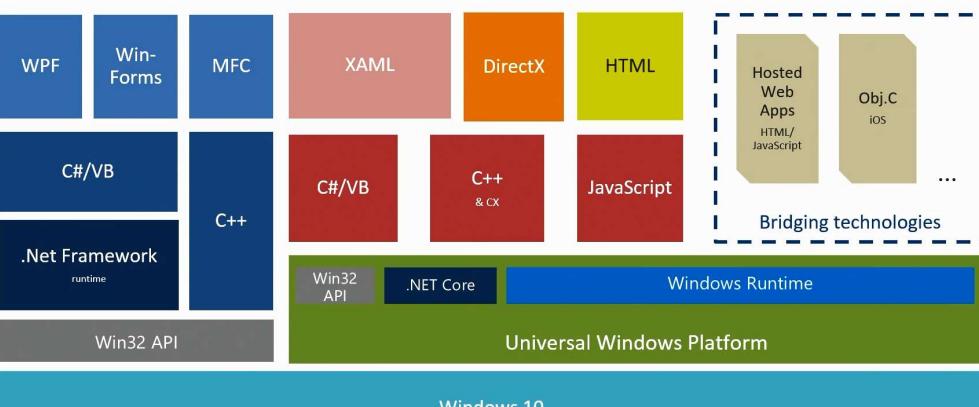
- **Applications Windows Universelles** (Windows Store App) :
  - ⇒ cible Windows 8, Windows Phone 8, Windows 10 et suivante
  - ⇒ associe plusieurs projets avec une partie du code source en commun
  - ⇒ VS 2013 et 2015 : de nouveaux Templates d'application commune Windows
- **Langages de programmation** dans Visual Studio pour Windows 10 :
  - Visual C++ avec DirectX avec ou sans XAML
  - C# ou Visual Basic avec XAML, JavaScript avec HTML
- **Windows 10** : nouveaux contrôles, panneaux de disposition et outils universels :
  - contrôles universels et panneaux de disposition pour optimiser l'UI pour la résolution d'écran de l'appareil (UI adaptable à différentes résolutions d'écran)
  - gestion commune des entrées pour les recevoir par le biais d'impulsions tactiles, d'un stylet, d'une souris, d'un clavier ou d'un contrôleur (manette Ms Xbox)

© 3iL Limoges - 2023 - Tous droits réservés -

41

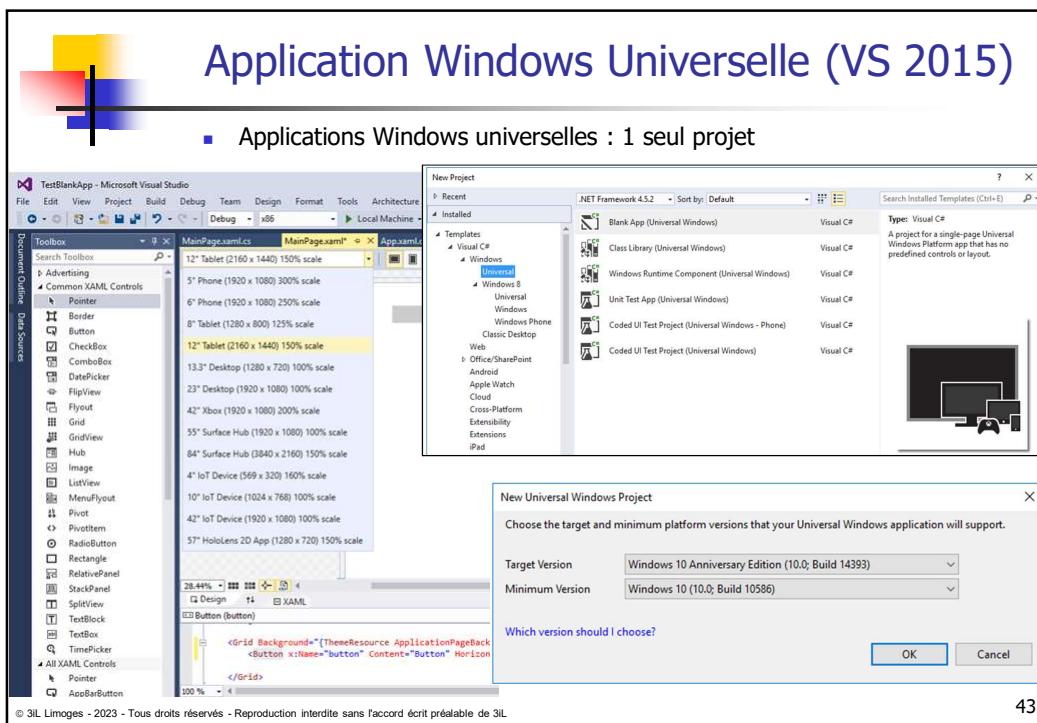


## UWP

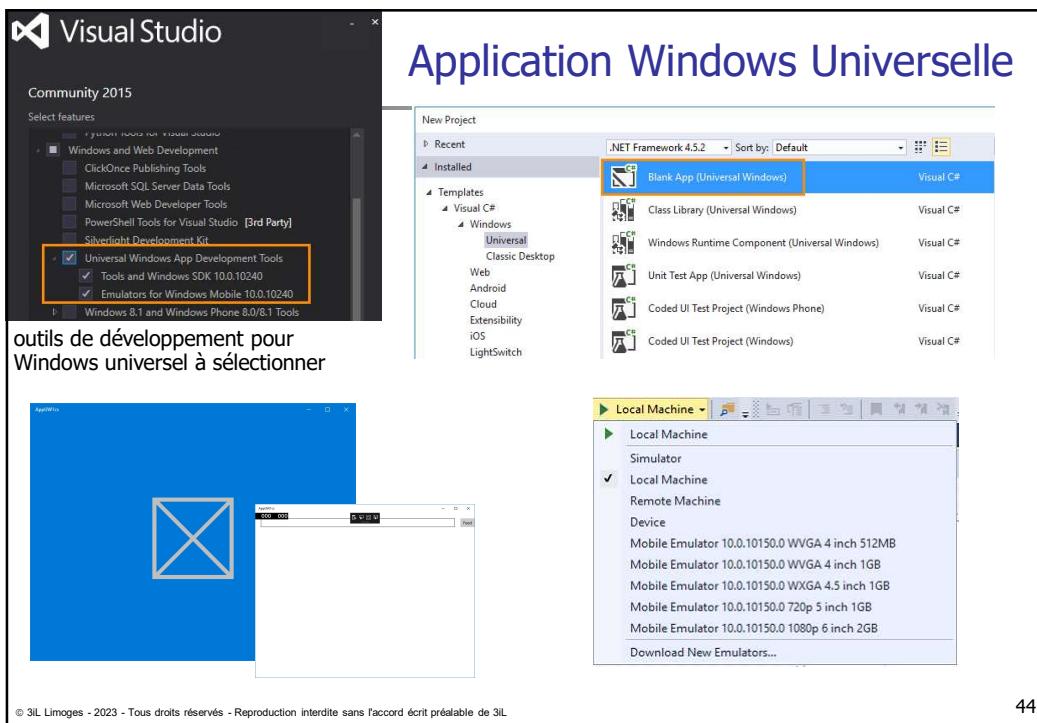


© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

42



43



44

## Application Windows Universelle (VS 2015)

**Solution 'TestBlankApp' (1 project)**

- TestBlankApp (Universal Windows)**
  - Properties
  - References
  - Assets
  - App.xaml
  - App.xaml.cs
  - MainPage.xaml
  - MainPage.xaml.cs
  - Package.appxmanifest
  - project.json
  - TestBlankApp\_TemporaryKey.pfx

```

namespace TestBlankApp
{
    /// <summary>
    /// Provides application-specific behavior
    /// </summary>
    sealed partial class App : Application
    {
        /// <summary>
        /// Initializes the singleton application
        /// object, and as such is the logi
        /// </summary>
        public App()
        {
            this.InitializeComponent();
            this.Suspending += OnSuspending;
        }
    }
}

if (rootFrame.Content == null)
{
    // When the navigation stack isn't restored navigate to the first page,
    // configuring the new page by passing required information as a navigation
    // parameter
    rootFrame.Navigate(typeof(MainPage), e.Arguments);
}
// Ensure the current window is active
Window.Current.Activate();

```

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

45

## Application Windows Universelle (VS 2015)

**TestBlankApp**

Charger

**Open**

App.xaml

- ApplicationInsights.config
- MainPage.xaml
- MainPage.xaml.cs
- MyApp\_TemporaryKey.pfx
- Package.appxmanifest

```

private async void button1_Click(object sender, RoutedEventArgs e)
{
    var picker = new Windows.Storage.Pickers.FileOpenPicker();
    picker.ViewMode = Windows.Storage.Pickers.PickerViewMode.Thumbnail;
    picker.SuggestedStartLocation = Windows.Storage.Pickers.PickerLocationId.PicturesLibrary;
    picker.FileTypeFilter.Add(".jpg");

    Windows.Storage.StorageFile file = await picker.PickSingleFileAsync();
    if (file != null)
    {
        var streams = await file.OpenAsync(Windows.Storage.FileAccessMode.Read);
        var img = new BitmapImage();
        img.SetSource(streams);
        image1.Source = img;
    }
}

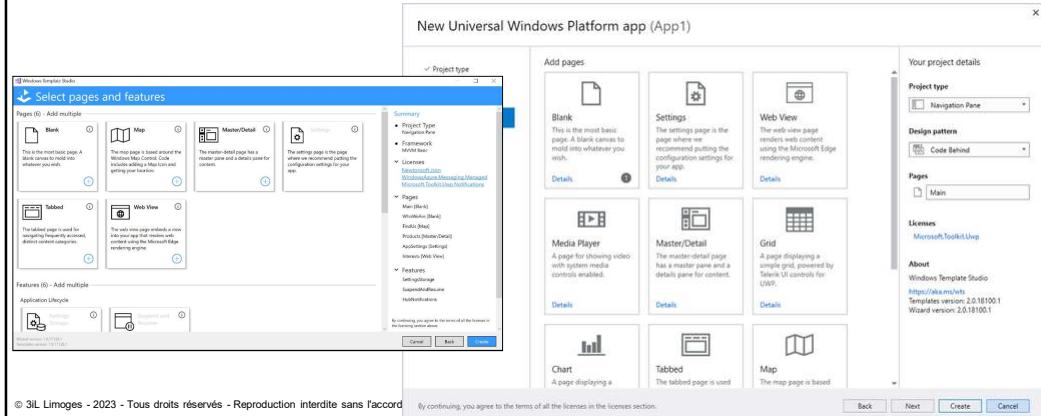
```

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

46

## Visual Template Studio

- **Visual Template Studio** (remplace l'outil en ligne gratuit Windows App Studio)
  - ⇒ extension Visual Studio qui permet le développement de squelettes graphiques d'applications destinées à tourner sur des appareils munis de Windows 10
  - ⇒ L'outil génère une ossature de code C# ou VB.net à compléter pour atteindre les objectifs définis d'avance ([UWP](#), [WPF](#), [WinUI 3](#))



## Appel d'une API non universelle

- Code adaptif : appel d'une API non universelle
  - ajouter une référence au SDK de la famille d'appareils ciblée
  - utiliser la classe **Windows.Foundation.Metadata.ApiInformation** dans une condition à l'intérieur du code pour tester la présence de l'API recherché :

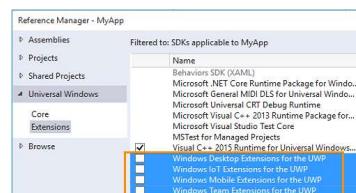
```
bool isHardwareButtonsAPIPresent =
    Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons");

if (isHardwareButtonsAPIPresent)
{
    Windows.Phone.UI.Input.HardwareButtons.CameraPressed +=  
        HardwareButtons_CameraPressed;
}
```

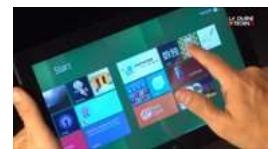
(teste la présence de l'API Windows Phone et implémente l'événement CameraPressed)

Autre possibilité :

```
bool isHardwareButtons_CameraPressedAPIPresent =
    Windows.Foundation.Metadata.ApiInformation.IsEventPresent
    ("Windows.Phone.UI.Input.HardwareButtons", "CameraPressed");
```



## 5. - .Net Core / WinUI

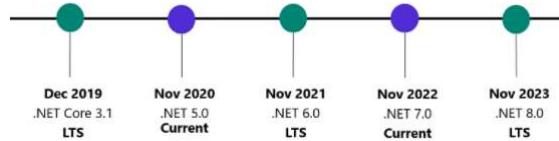


© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

49

### .Net Core

- **.NET Core** peut être considéré comme une [version multiplateforme de .NET Framework](#), située au niveau de la couche des bibliothèques de classes de base (BCL). Il implémente la spécification de .NET Standard.
  - ⇒ plateforme unifiée pour les développeurs combinant dev .NET, Cloud, gaming, IoT, web, et l'IA
  - ⇒ **NET 5.0 Core et suivant ne remplace pas .NET Standard (2021)**
  - ⇒ Inclus : WinRT (disponible depuis Windows 8), compilateurs Roslyn C# et VB.NET, langages C#, VB.NET et F# et C++/CLI et :
    - ⇒ le desktop avec WPF, WinForms et UWP,
    - ⇒ Le web avec ASP.NET,
    - ⇒ Le cloud avec Azure,
    - ⇒ Le mobile avec Maui,
    - ⇒ Le gaming avec Unity,
    - ⇒ L'IoT,
    - ⇒ L'AI avec ML.NET.

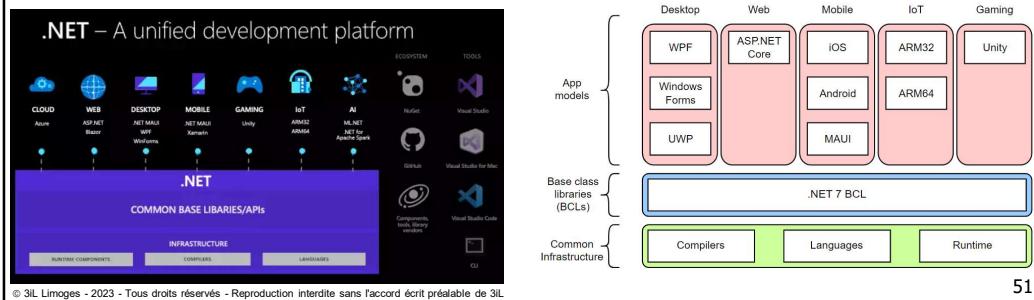


© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

50

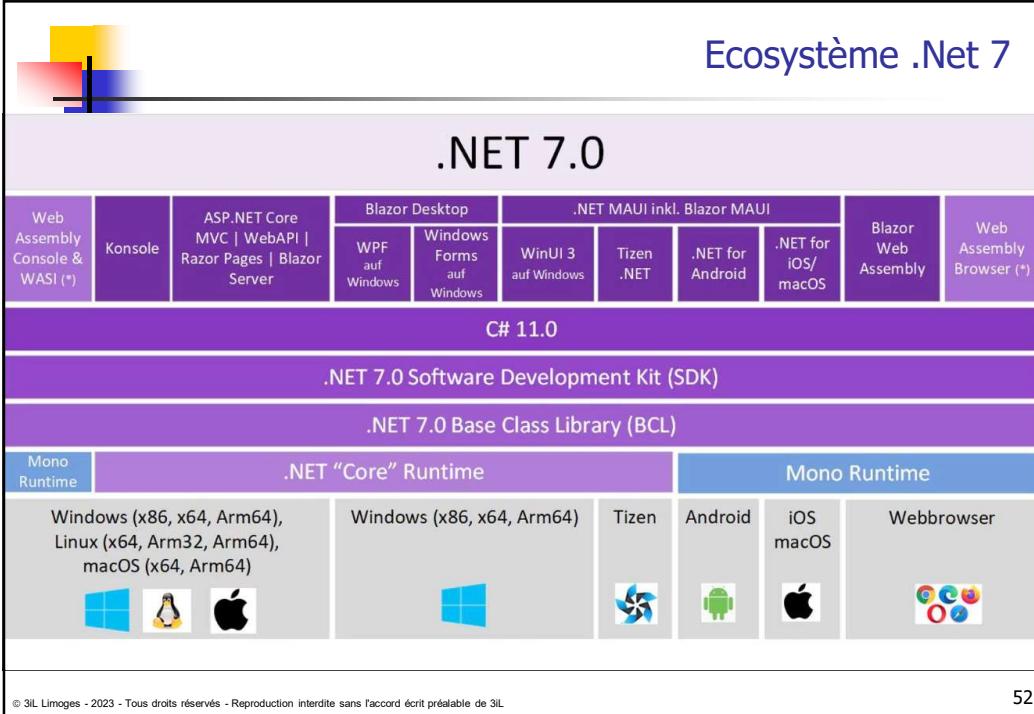
## .Net Core

- .Net 7 ⇒ .Net Framework 4.8 et .NET Core 7.0
- ⇒ **.Net Framework 4.8** : runtime pour Windows (nombreuses améliorations dans les classes de base, SSL dans WPF, chiffrement dans ADO, ...)
- ⇒ **.Net Core 7** : version modulaire du .Net Frwk portable sur plusieurs plateformes (Linux et OS X), open source et ouvert aux communautés (partage et réutilisation de code, pas de contraintes de livraison du Frwk , moteur lié à l'application qui l'exécute)
- ⇒ C#11, C++, Windows Forms, WPF, CoreWCF, EF Core, ASP.NET Core
- ⇒ Compilateur JIT 64 bits, .NET Native (applications Windows 10 ciblant .Net Core en C# ou VB peuvent être compilée en code natif plutôt qu'en code IL)

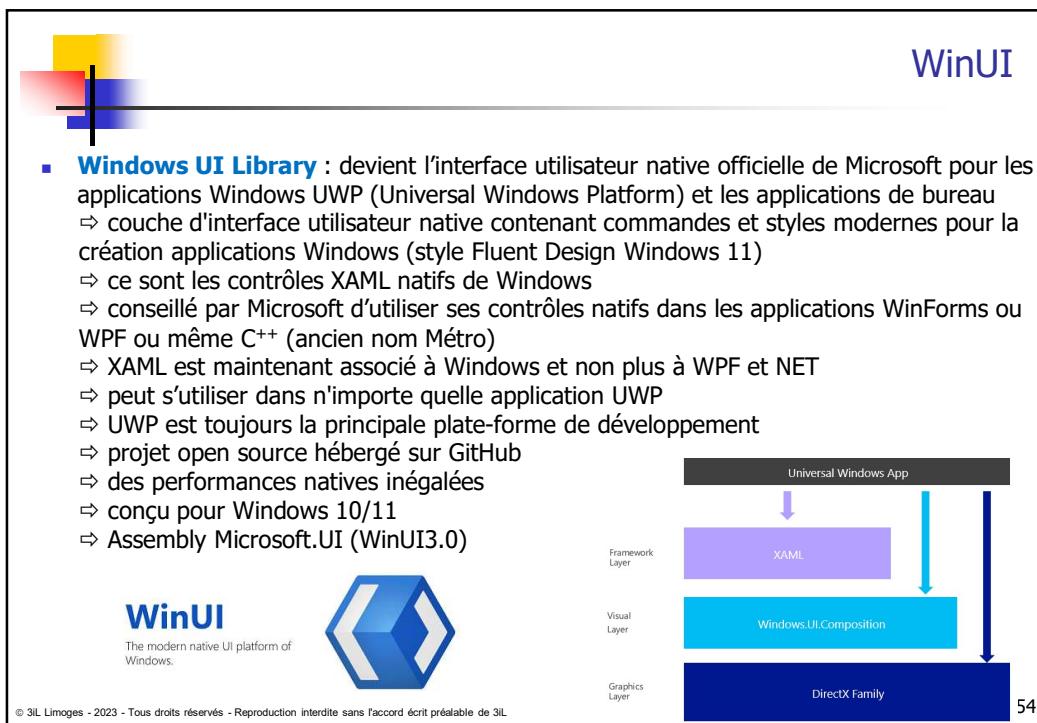


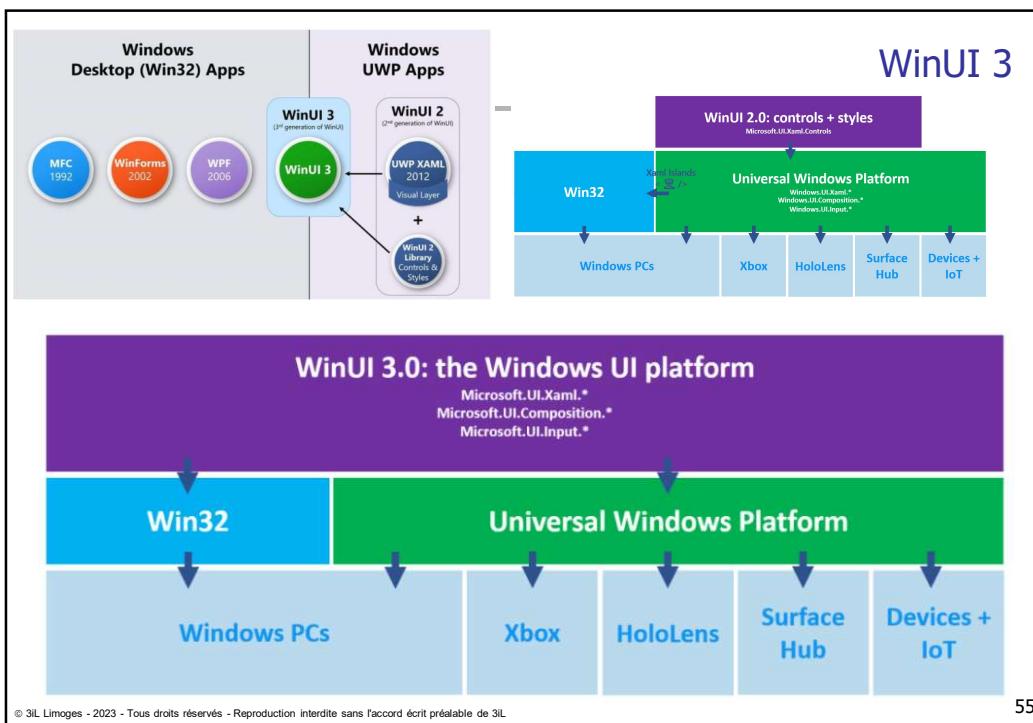
51

## Ecosystème .Net 7



52





## WinUI

- **WinUI** : Des interfaces utilisateur modernes et transparentes, aspect et une sensation modernes (Fluent Design System) offre des commandes flexibles et modernes telles que NavigationView et TeachingTip

The GitHub repository shows samples for WinUI 3 and WinUI in UWP, including:

- Blank App (WinUI in UWP)**: A project for creating a Universal Windows Platform (UWP) app based on the Windows UI Library (WinUI).
- Blank App, Packaged (WinUI in Desktop)**: A project for creating a Desktop app based on the Windows UI Library (WinUI) along with a MSIX package for side-loading or distribution via the Microsoft Store.

**WinUI Samples** (GitHub repository):

- What's New**: WinUI 3 Reunion 0.5
- Recently Added Samples**: WebView2 (A Microsoft Edge (Chromium) based control that hosts HTML content in an app.)
- Recently Updated Samples**: AppTabBar (Provides a tab bar for navigating between multiple pages in an application. It provides a standard appearance for tabs and allows users select only one option at a time.)
- Portfolio**: A financial portfolio management application showing charts for AAPL and MNGR.
- Calendar**: A calendar application showing events for May 2020.

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

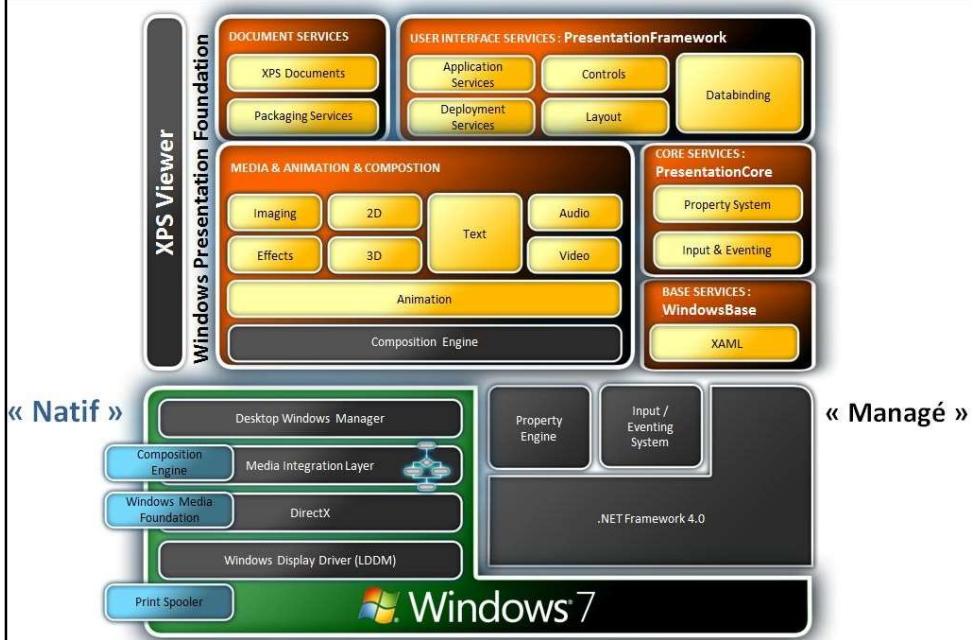
56

## Annexes

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

57

## Architecture WPF



58

## Outils de développement

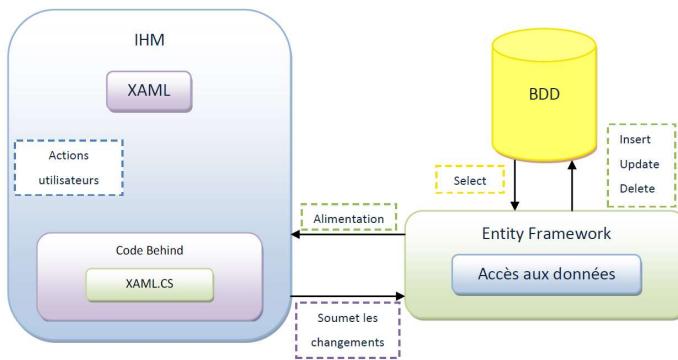
- **Microsoft Visual Studio 2010-2013-2015-2017-2019-2022**  
conception de projets WPF ou Silverlight (attention à la version du .Net)  
(Visual C# Express et Visual Basic Express permettent de réaliser des projets WPF)
- **(Visual Studio .Net 2005 + PlugIn Microsoft Cider** (outil de conception graphique XAML, fourni comme Add-in de Visual Studio 2005 CTP)
- **Microsoft Expression Design** : conceptions graphiques en vue d'une exportation en XAML pour WPF (2D, 3D, textes, animations, vidéo, etc.)
- **Microsoft Expression Blend** : (écrit en WPF)  
outil professionnel de conception de GUI RIA WPF et/ou Silverlight  
permet d'utiliser des fichiers XAML exportés de Expression Design
- **XamlPad** : éditeur graphique de base pour XAML (SDK Microsoft)
- Mono n'implémente pas WPF, **Moonlight** implémente un sous-ensemble de WPF

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

59

## Architecture classique d'une application .NET

- Architecture classique avec une couche d'accès aux données (Entity Framework ou Linq to SQL )  
⇒ Rapidité de développement , pas besoin de formations, Framework .NET suffit , s'approprier un Framework spécialisé  
⇒ Manque de lisibilité, difficulté de maintenances et d'évolutions, code difficilement réutilisable



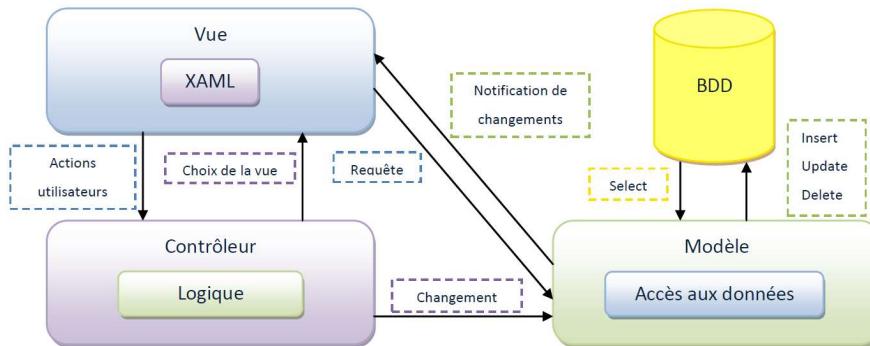
© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

60

## L'architecture Modèle-Vue-Contrôleur

### ■ Architecture MVC

- ⇒ gain de temps lors de maintenances et d'évolutions, conception claire et efficace, souplesse d'organisation
- ⇒ IHM pas indépendante, complexité de mise en place, besoin de formation



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

61

## L'architecture Modèle-Vue-Vue Modèle

- **Architecture M-V-VM :** premier pattern à faire intervenir les designers/graphistes ⇒ séparation complète entre IHM et reste du code pour que le designer puisse travailler
- La couche ViewModel fait le lien entre la couche View et Model : capte les interactions de la vue et notifie à celle-ci lorsque des données sont modifiées ⇒ contient la logique de l'application

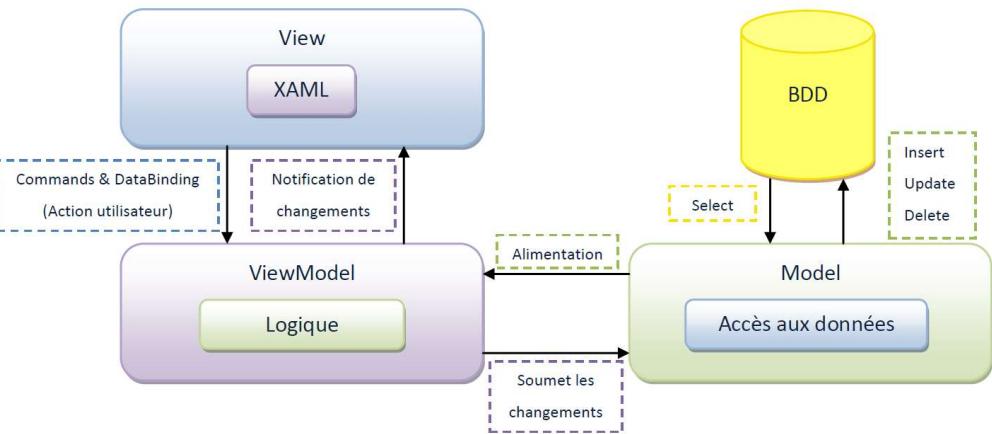
- ⇒ *gain de temps de maintenance et d'évolutions, conception claire et efficace, souplesse d'organisation, facilement testable*
- ⇒ *Complexité de mise en place, formation, Data Binding*
- ⇒ *Utilisation restreinte aux technologies WPF, Silverlight et Windows Phone 7*

Framework : **M-V-VM Light Toolkit, Cinch, Prism**

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

62

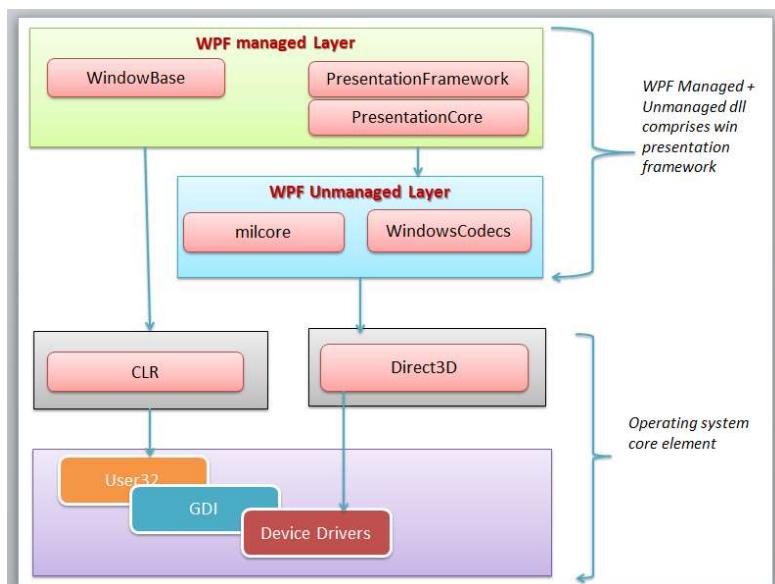
## L'architecture Modèle-Vue-Vue Modèle



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

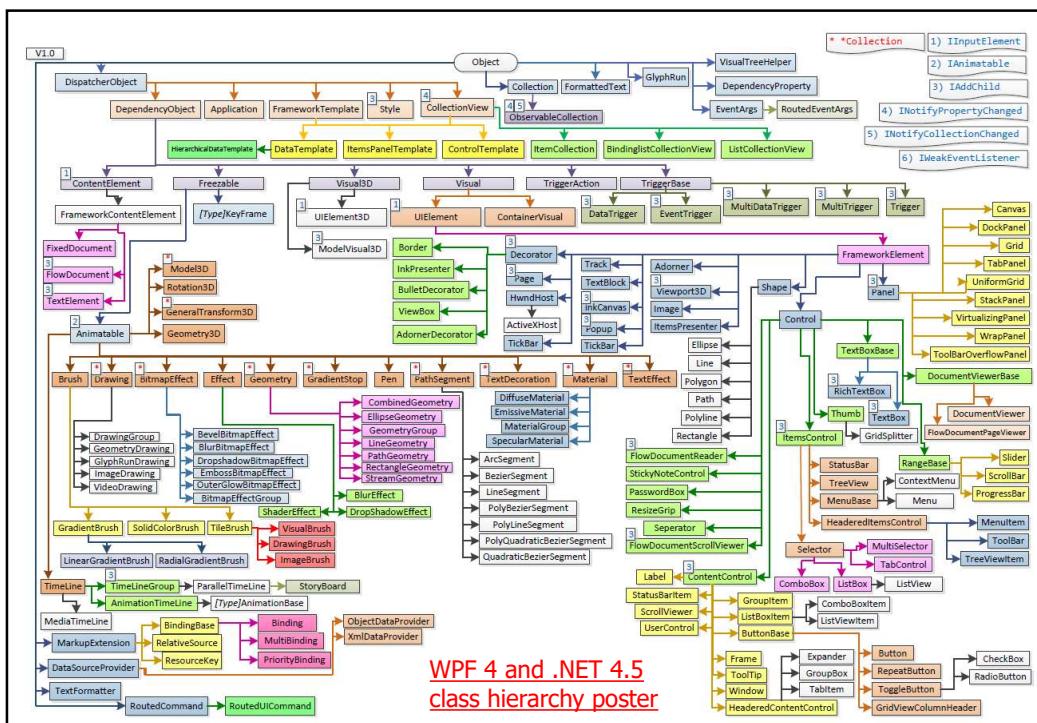
63

## Moteur WPF



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

64



WPF 4 and .NET 4.5  
class hierarchy poster