PLATEFORMES LOGICIELLES POUR L'INFORMATIQUE MOBILE

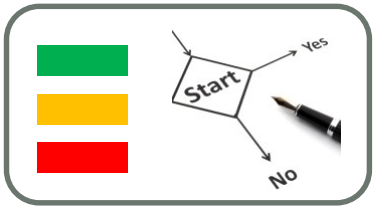
INTRODUCTION À LA PROGRAMMATION SOUS WINDOWS PHONE

Jean-Yves Tigli, http://www.tigli.fr
Email : tigli@polytech.unice.fr

POLYTECH NICE-SOPHIA

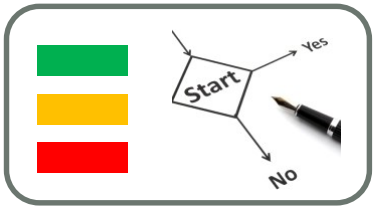
Université Nice Sophia Antipolis

Quelques principes généraux



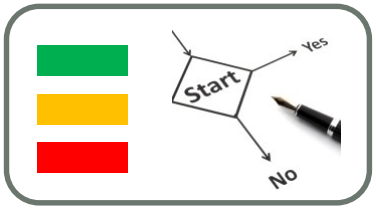
- Programmation en deux temps
- Le rendu graphique avec le Designer et le langage XAML
- La logique de l'application en C# dans le events handlers entre autre

Quelques principes généraux



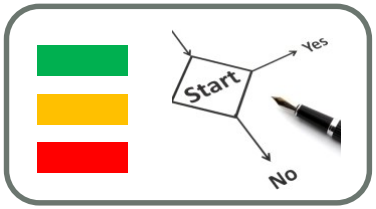
- Le layout graphique est produit grâce au designer de visual studio
  - Vous pouvez utiliser la boîte à outils pour « drag and droper » les widgets (aussi appelés contrôles) que vous voulez utiliser dans votre application
  - Vous pouvez interactivement modifier le positionnement de ces contrôles et leur propriétés.
- Tout cela correspond à des modification du fichier XAML (eXtensible Application Markup Language) de l'application
  - Même si XAML n'est pas dédié qu'au rendu graphique comme nous le verrons

Liste des contrôles (XAML)



- L'infrastructure d'interface utilisateur XAML pour Windows fournit une bibliothèque complète de contrôles qui prennent en charge le développement d'une interface utilisateur.
- Certains de ces contrôles ont une représentation visuelle, tandis que d'autres font office de conteneurs d'autres contrôles ou d'autre contenu, par exemple des images ou du contenu multimédia.
- Vous pouvez observer le fonctionnement des contrôles d'interface utilisateur Windows en téléchargeant l'exemple Galerie de contrôles et de dispositions et l'exemple Contrôles fondamentaux XAML.

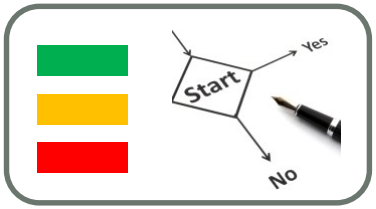
XAML



- Définition: un langage XML qui décrit votre interface.

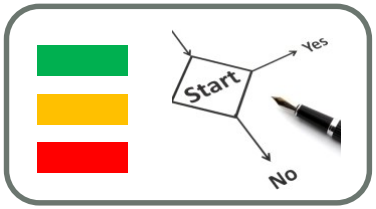
- Utilité: utilisé pour créer deux types de programme:
  - Un logiciel classique, sous Windows. C'est alors WPF qui se charge de comprendre le XAML pour créer votre interface graphique et la faire interagir avec votre code .net (C#, Vb.net, ...).
  - Une interface Web. Dans ce cas c'est Silverlight qui prend en charge votre code. Le programme s'exécutera dans votre navigateur, comme les applications Flash par exemple.

Hello World: XAML



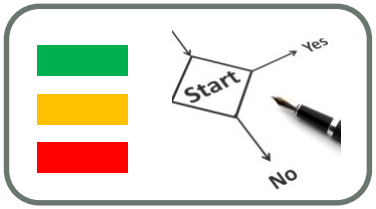
- <Window x:Class="WpfApplication1.MainWindow"
- xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
- xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
- Title="MainWindow" Height="350" Width="525">
- <TextBlock Text="Hello WPF" />
- </Window>
- TextBlock est l'équivalent à label en winforms

Equivalent C#




```
using System;
using System.Windows;
using System.Windows.Controls;
namespace HelloWorld1
{
    class HelloWindow
    {
        static void Main()
        {
            // Création de la fenetre
            Window fenetre = new Window();
            // Assignation des propriétées
            fenetre.Title = "Window1";  fenetre.Width = 300;  fenetre.Height = 300;
            // Création du TextBlock
            TextBlock tblock = new TextBlock(); tblock.Text = "Hello le SDZ";
            // Ajout du contrôle à la fentre
            fenetre.Content = tblock;
            // Création de l'application et démarrage avec notre fentre
            Application app = new Application();
            app.Run(fenetre);
        }
    }
}
```

Première action



----------MainWindow.xaml----------

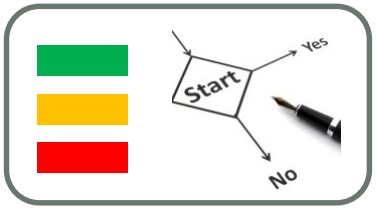
```
<Window x:Class="WpfApplication3.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Button Click="Button_Click">Cliquez ici !</Button>

</Window>
```

----------MainWindow.xaml.cs---------

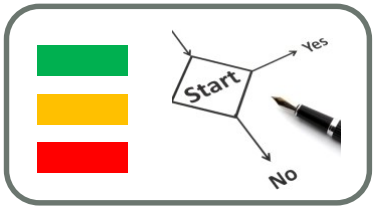
```
 private void Button_Click(object sender, RoutedEventArgs e)
 {
            MessageBox.Show("Coucou");
            this.Close();
 }
```

Conteneurs : la grille



- Le contrôle Window ne peut contenir qu'un seul élément.
- Les éléments chargés d'en contenir d'autres sont appelés « conteneurs ».
- La grille peut être vue comme un tableau en XHTML. Ce tableau représente, comme son nom l'indique, une grille avec plusieurs colonnes et lignes. Chaque contrôle est alors positionné dans une des cellules du tableau. Évidemment, chaque ligne et colonne peut être de dimension différente. De plus un contrôle peut appartenir à plusieurs cases à la fois.

Dimensions (Width, Height)



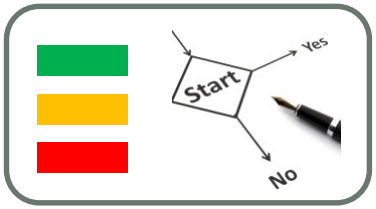
- Les dimension peuvent être définit de 3 manières:
  - une valeur fixe, permettant de définir précisément la taille de l'élément (plusieurs unités sont disponibles) ;
  - une portion de l'espace disponible, représentée par une étoile.
  - la valeur « Auto » pour que l'élément s'adapte au contenu.

- Une valeur fixe peut être de plusieurs types :
  - le dip (unité px) qui est celle par défaut (ex : « 10pix » ou « 10 ») ;
  - le pouce (unité in, 1in=96pix, ex : 10in) ;
  - le centimètre (unité cm, 1cm=96/2.54pix, ex : 10cm) ;
  - le point (unité pt, 1pt = 96/72 pix, ex 10pt).

Définition des lignes et colonnes

```
<Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="5cm"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="70"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="2*"/>
        </Grid.RowDefinitions>
</Grid>
```

Positionnement des contrôles dans les cellules

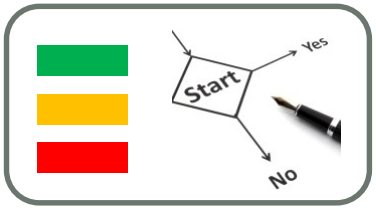


- Si vous ajoutez des contrôles à votre grille, ils vont tous se trouver par défaut assignés à la première cellule.

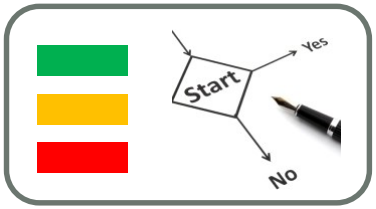
- Pour spécifier la position dans la grille on utilise les propriétés « Grid.Column » et « Grid.Row» pour spécifier la colonne

- Il s'agit d'un nouveau concept. Les propriétés attachées (atached property). Ainsi en dehors d'une grille, aucun contrôle ne possède ces propriétés et tous les contrôles les possèdent à l'intérieur.

- Il peut parfois être intéressant qu'un contrôle remplisse plusieurs cellules. Cela se réalise en utilisant les propriétés attachées «Grid.RowSpan» et « Grid.ColumnSpan ».



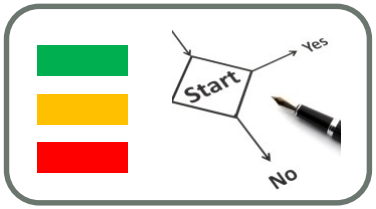

```
<Button Grid.Column="0" Grid.Row="0" Content="Bouton 1"/>
<Button Grid.Column="1" Grid.Row="0" Content="Bouton 2"/>
<Button Grid.Column="2" Grid.Row="0" Content="Bouton 3"/>
<Button Grid.Column="0" Grid.Row="1" Content="Bouton 4"/>
<Button Grid.Column="1" Grid.Row="1" Content="Bouton 5"/>
<Button Grid.Column="2" Grid.Row="1" Content="Bouton 6"/>
<Button Grid.Column="0" Grid.Row="2" Content="Bouton 7"/>
<Button Grid.Column="1" Grid.Row="2" Content="Bouton 8"/>
<Button Grid.Column="2" Grid.Row="2" Content="Bouton 9"/>
```



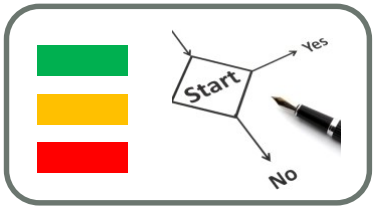
Window1

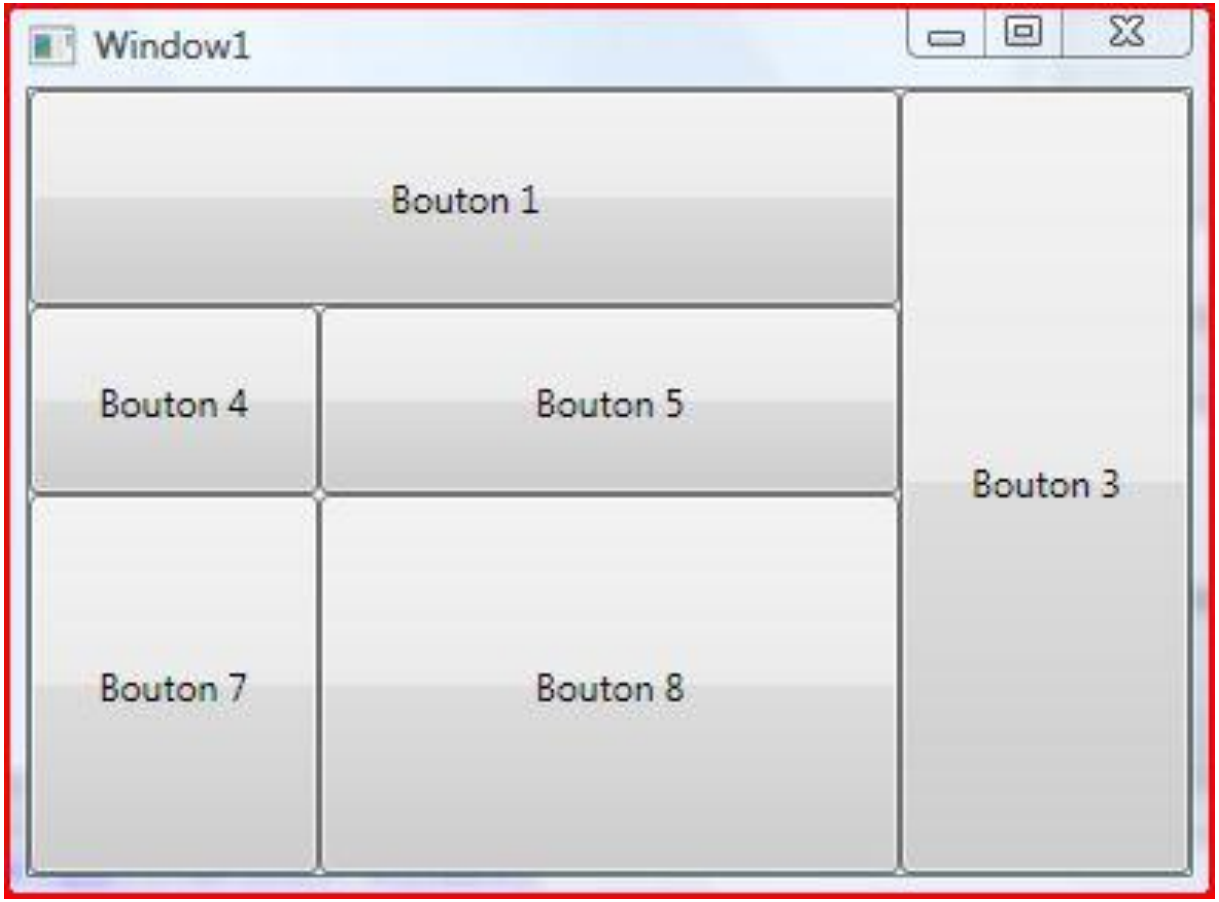
Bouton 1	Bouton 2	Bouton 3
Bouton 4	Bouton 5	Bouton 6
Bouton 7	Bouton 8	Bouton 9

Remplissage de plusieurs cellules

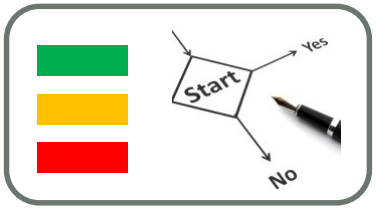



```
<Grid> <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="5cm"/>
                    <ColumnDefinition Width="*"/>
          </Grid.ColumnDefinitions>
          <Grid.RowDefinitions>
                    <RowDefinition Height="70"/>
                    <RowDefinition Height="*"/>
                    <RowDefinition Height="2*"/>
          </Grid.RowDefinitions>
                    <Button   Grid.Column="0" Grid.Row="0" Grid.ColumnSpan="2" Content="Bouton 1"/>
                    <Button   Grid.Column="2" Grid.Row="0" Grid.RowSpan="3" Content="Bouton 3"/>
                    <Button   Grid.Column="0" Grid.Row="1" Content="Bouton 4"/>
                    <Button   Grid.Column="1" Grid.Row="1" Content="Bouton 5"/>
                    <Button   Grid.Column="0" Grid.Row="2" Content="Bouton 7"/>
                    <Button   Grid.Column="1" Grid.Row="2" Content="Bouton 8"/>
</Grid>
```


Start
Yes
No


Window1
Bouton 1
Bouton 4
Bouton 5
Bouton 3
Bouton 7
Bouton 8

Cas particulier : les grilles uniformes

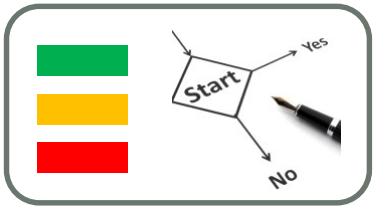


- Il s'agit de grilles dont toutes les colonnes ont la même largeur et toutes les lignes la même hauteur.

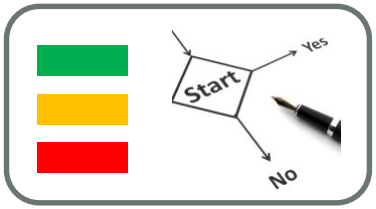
- Il n'y a donc pas besoin de « ColumnDefinition » ou de « RowDefinition ». Il y a deux propriétés Rows et Columns qui définissent le nombre de lignes et de colonnes.

- Les contrôles sont ajoutés à la grille dans l'ordre

Exemple avec une grille 3x2 remplie de boutons.

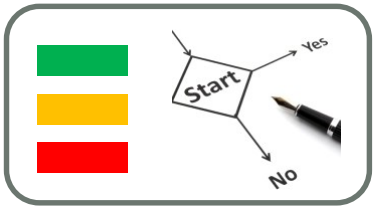


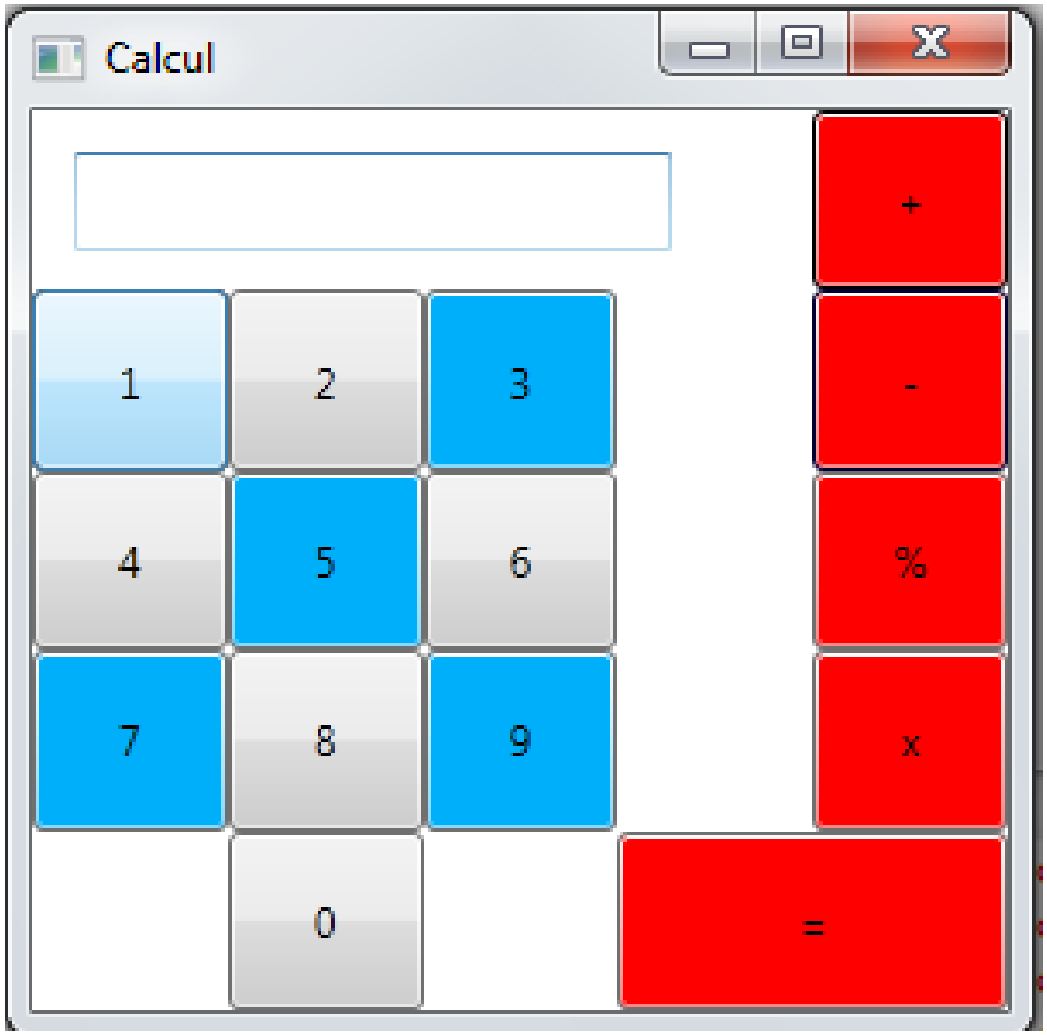

```
<UniformGrid Rows="2" Columns="3">
    <Button>Bouton 1</Button>
    <Button>Bouton 2</Button>
    <Button>Bouton 3</Button>
    <Button>Bouton 4</Button>
    <Button>Bouton 5</Button>
    <Button>Bouton 6</Button>
</UniformGrid>
```


Start
Yes
No

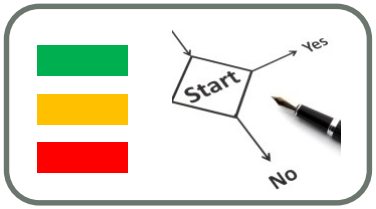

Window1
Bouton 1
Bouton 2
Bouton 3
Bouton 4
Bouton 5
Bouton 6

Atelier Calculatrice





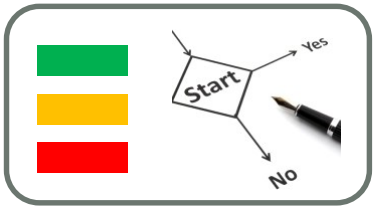
Grid Splitter

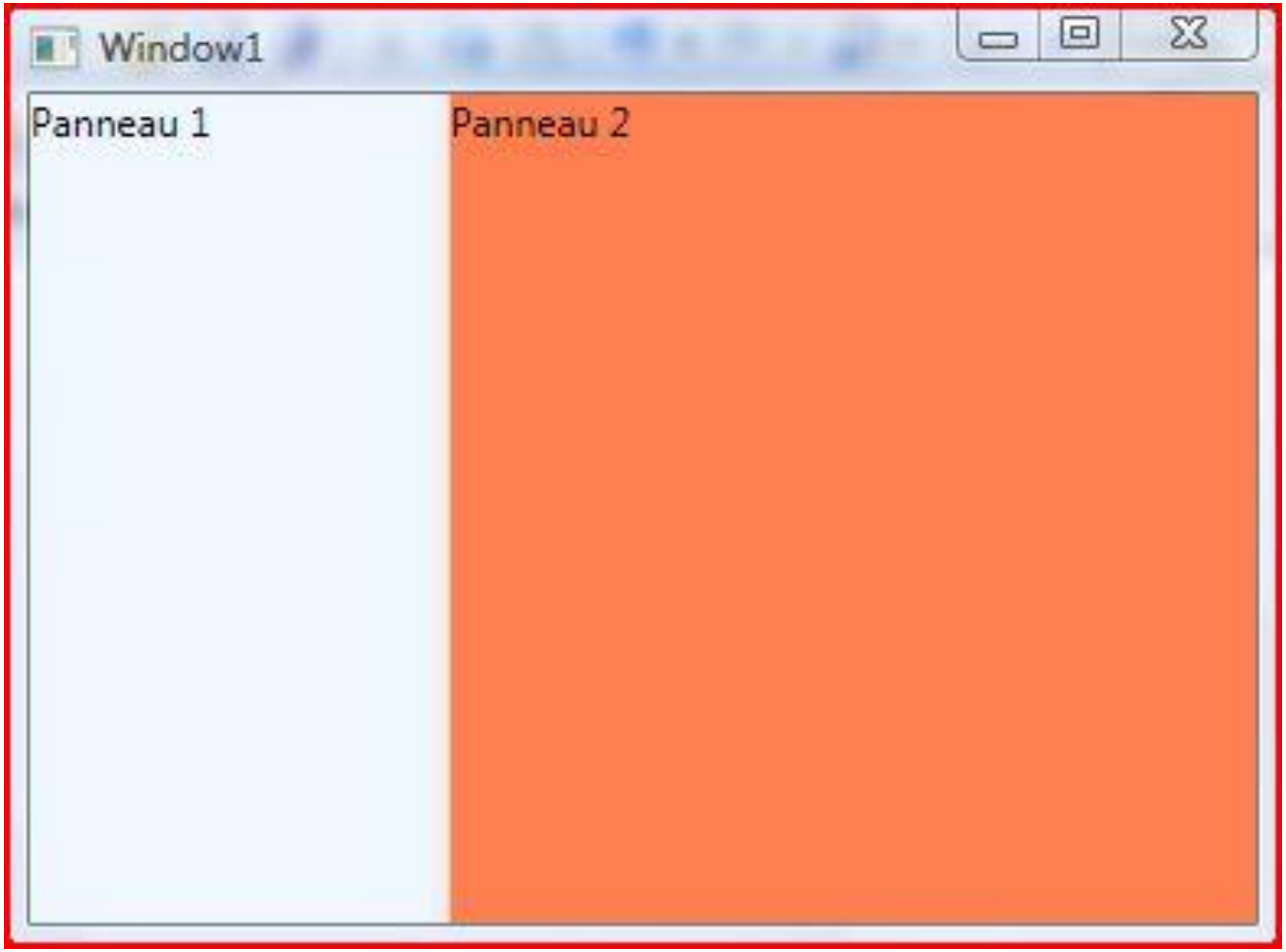


- Lorsqu'un logiciel a plusieurs panneaux, il est très pratique de pouvoir changer leur taille en fonction de son utilisation.
- Le « Grid Splitter » permet de redimensionner automatiquement les autres colonnes ou lignes.

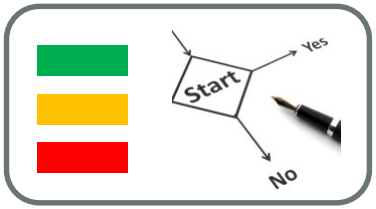
```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*"/>
        <ColumnDefinition Width="5"/>
        <ColumnDefinition Width="2*"/>
    </Grid.ColumnDefinitions>
    <TextBlock Grid.Column="0" Text="Panneau 1" Background="AliceBlue" />
    <GridSplitter Grid.Column="1" HorizontalAlignment="Stretch" />
    <TextBlock Grid.Column="2" Text="Panneau 2" Background="Coral" />
</Grid>
```

- La propriété «HorizontalAlignment» permet de spécifier la position horizontale du contrôle vis-à-vis de son conteneur (ici notre cellule). Nous positionnons cette propriété à « Strech » pour que le contrôle prenne toute la largeur (les 5dip)


Start
Yes
No

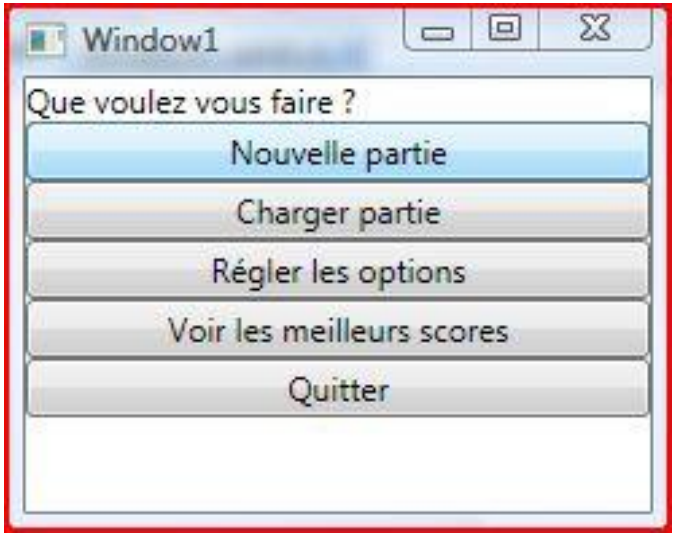

Window1
Panneau 1
Panneau 2

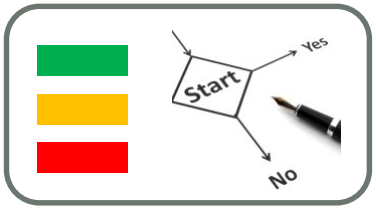
Les conteneurs à empilement: StackPanel



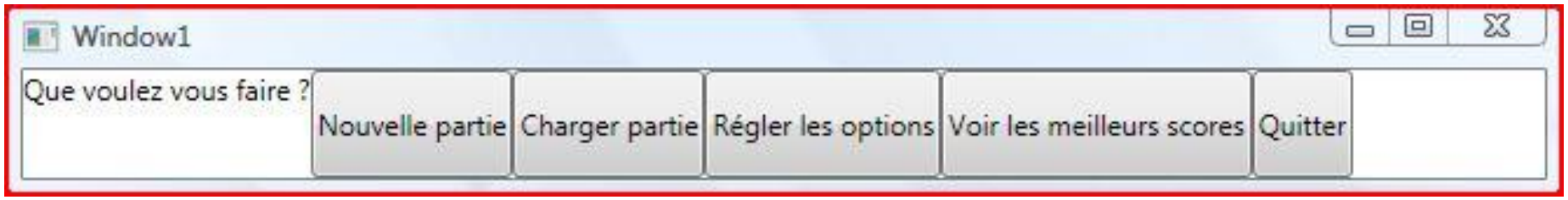
- Le « StackPanel » positionne automatiquement les contrôles qu'on lui ajoute les uns en dessous des autres, ou les uns à la suite des autres en fonction de la valeur de la propriété « Orientation ». Cette propriété peut logiquement prendre les valeurs « Vertical » ou « Horizontal ».

```
<StackPanel Orientation="Vertical">
        <TextBlock Text="Que voulez vous faire ?"/>
        <Button Content="Nouvelle partie"/>
        <Button Content="Charger partie"/>
        <Button Content="Régler les options"/>
        <Button Content="Voir les meilleurs scores"/>
        <Button Content="Quitter"/>
</StackPanel>
```

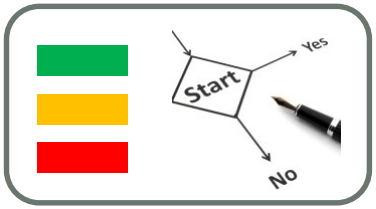




- Si vous changez la propriété « Orientation » en «Horizontal», vous devriez obtenir le résultat suivant :

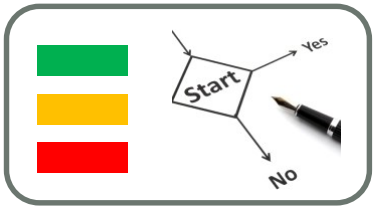


Les conteneurs à empilement: WrapPanel

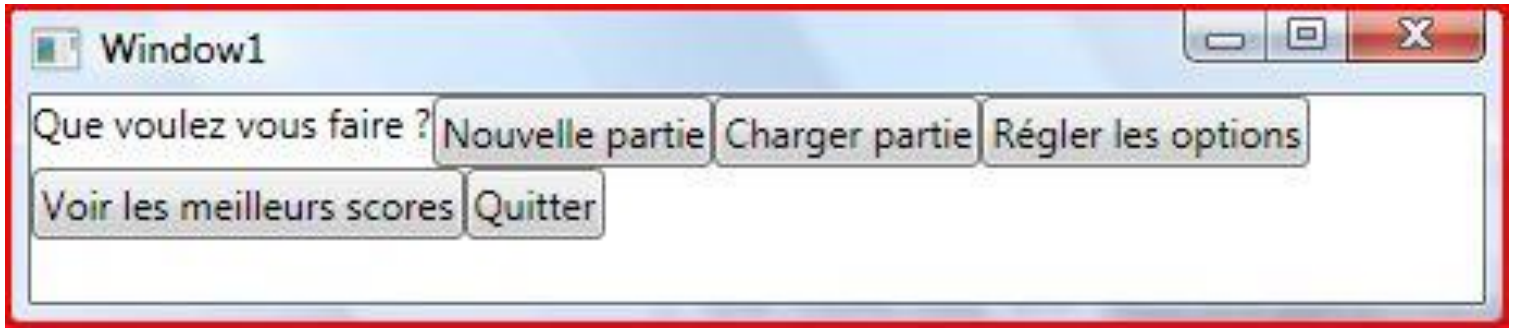


- Le conteneur précédent est très pratique mais si on redimensionne la fenêtre, les contrôles disparaissent. Il est parfois intéressant de permettre l'extension sur une deuxième ligne ou colonne des éléments. C'est le rôle du WrapPanel.

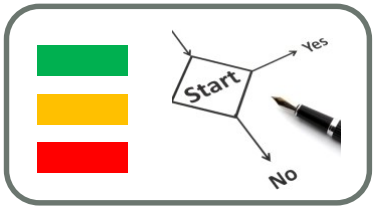
- Pour l'exemple de l'empilement horizontal, le WrapPanel ne va pas chercher à occuper directement toute la hauteur disponible. Il s'en servira pour positionner les contrôles s'il n'y a plus d'espace disponible en bout de ligne.



- Si on redimentionne:



Les conteneurs à empilement: Canvas et DockPanel

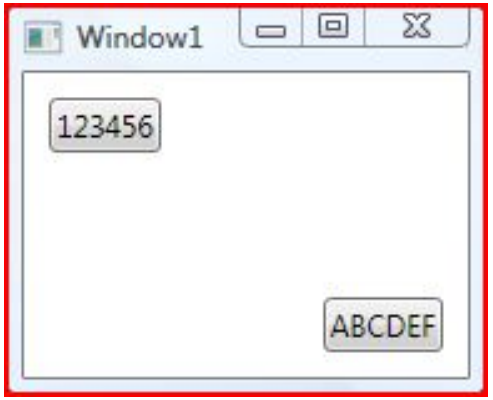


- Nous avons vu jusqu'ici quelques conteneurs forts pratiques. Cependant il peut être intéressant de pouvoir, quelquefois, posséder des conteneurs plus proches du fonctionnement des Windows Forms.

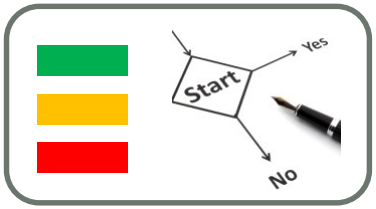
- Le conteneur Canvas représente une zone où le comportement est le même que celui des Windows Forms. La position est fournie par les propriétés attachées « Canvas.Top », « Canvas.Left », « Canvas.Bottom » et « Canvas.Right ».

- Le contrôle Canvas peut être très pratique pour effectuer des dessins, où il est plus simple de travailler en coordonnées absolues plutôt que relatives.

```
<Canvas>
    <Button Canvas.Top="10" Canvas.Left="10"
Content="123456"/>
    <Button Canvas.Right="10" Canvas.Bottom="10"
Content="ABCDEF" />
</Canvas>
```



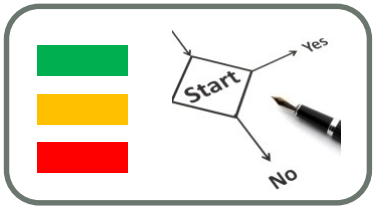
Modifier une propriété attaché



- Les propriétés attachées ne sont pas définies au niveau de l'objet lui-même, mais au niveau du contrôle qui définit ces propriétés.

- Le contrôle bouton par exemple ne sait pas utiliser directement les propriétés Canevas.Right et Canevas.Left, c'est en le plaçant dans un Canvas qu'elles sont exploitées.

Exemple:

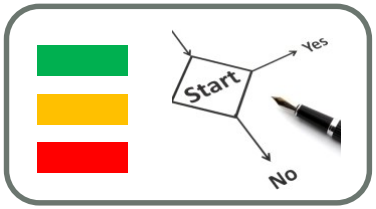



```
<Canvas>
  <Button Name="Boutton1" Canvas.Top="10" Canvas.Left="10" Content="123456“ Click="Button_Click"/>
  <Button Canvas.Right="10" Canvas.Bottom="10" Content="ABCDEF" />
</Canvas>
```

- Pour déplacer notre bouton après chaque click, nous devons tout d'abord connaître sa position courante. Nous l'obtenons en utilisant les méthodes statiques « GetLeft » et « GetTop » appliquées à notre contrôle.

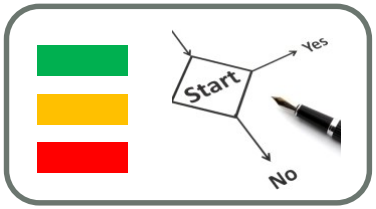
- Pour changer la position il faut utiliser les méthodes statiques « SetLeft » et « SetTop ». Vous devez fournir à ces méthodes deux paramètres. Le premier est le contrôle auquel on souhaite effectuer la modification, et le deuxième, la nouvelle valeur du paramètre.

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    // Récupération de la position de l'élément
    double PositionLeft = Canvas.GetLeft(Boutton1);
    double PositionTop = Canvas.GetTop(Boutton1);
    // Modification de la position
    Canvas.SetLeft(Boutton1, PositionLeft + 10);
    Canvas.SetTop(Boutton1, PositionTop + 10);
}
```

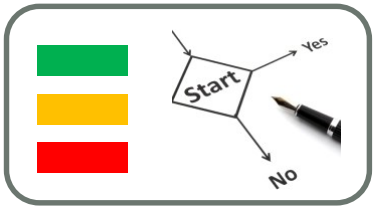


- De même, pour modifier l'affectation d'un élément à une cellule de grille, vous pouvez de la même manière utiliser les méthodes statiques « SetRow », « SetColumn », « SetRowSpan » et « SetColumnSpan » de la classe « Grid ».

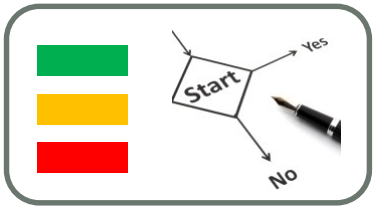
DockPanel



- DockPanel est un exemple typique de l’utilisation des propriétés attachés.
  - Tous les contrôles Windows Forms possédaient une propriété « Dock » nous permettant de les coller aux bords d'une fenêtre. Nous pouvons faire pareil avec WPF en utilisant le « DockPanel ».
  - L'ajout de contrôles dans le DockPanel leur permet de définir les propriétés « DockPanel.Dock » qui peuvent prendre les valeurs « Left », « Right », « Top » ou « Bottom ».

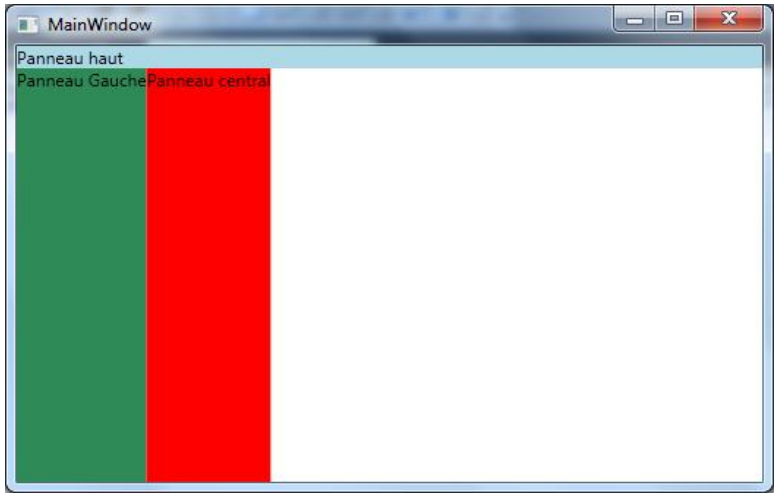



```
<DockPanel>
    <TextBlock DockPanel.Dock="Top" Background="LightBlue" Text="Panneau haut"/>
    <TextBlock DockPanel.Dock="Left" Background="SeaGreen" Text="Panneau Gauche"/>
    <TextBlock Background="Red" Text="Panneau central"/>
</DockPanel>
```

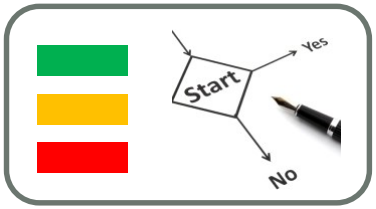


- Comme vous venez de voir, par défaut, le dernier contrôle remplit automatiquement tout l'espace restant. Ce comportement peut être modifié en positionnant la propriété « LastChildFill » du « DockPanel » à False.

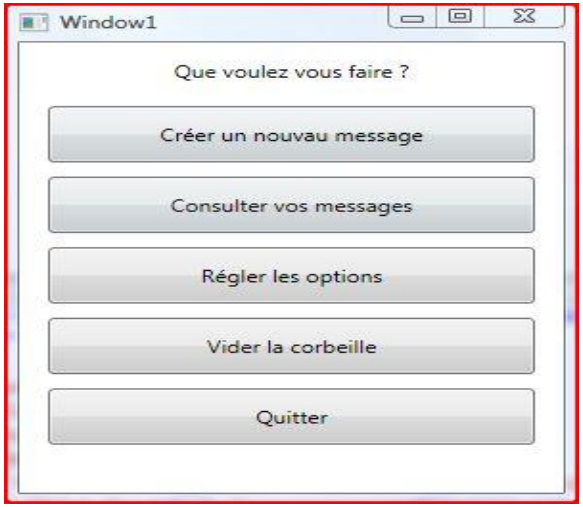
```
<DockPanel LastChildFill="False">
 <TextBlock DockPanel.Dock="Top" Background="LightBlue" Text="Panneau haut"/>
 <TextBlock DockPanel.Dock="Left" Background="SeaGreen"   Text="Panneau Gauche"/>
  <TextBlock Background="Red" Text="Panneau central"/>
</DockPanel>
```



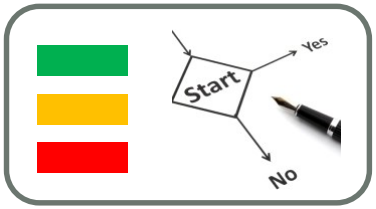
Position et taille des contrôles



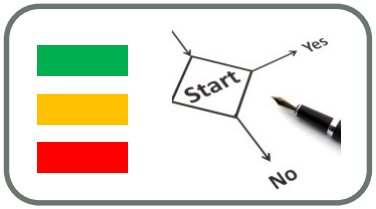

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="40"/>
        <RowDefinition Height="*"/>
    </Grid.RowDefinitions>
    <TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Grid.Row="0" Text="Que voulez-vous faire ?"/>
    <StackPanel Grid.Row="1" Orientation="Vertical">
        <TextBox Height="37"  Name="txtMessage" />
        <Button Height="40" Margin="15,5" Content="Créer un nouveau message"/>
        <Button Height="40" Margin="15,5" Content="Consulter vos messages"/>
        <Button Height="40" Margin="15,5" Content="Régler les options"/>
        <Button Height="40" Margin="15,5" Content="Vider la corbeille"/>
        <Button Height="40" Margin="15,5" Content="Quitter"/>
    </StackPanel>
 </Grid>
```



Personnalisation des fenêtres

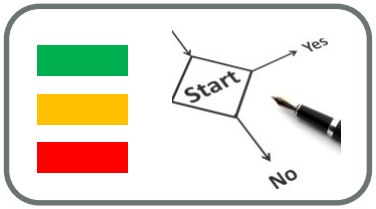


- Barre de titres
- Taille et position
- Apparence de la fenêtre

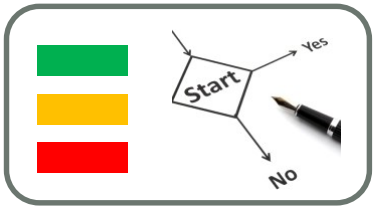



```
<Window x:Class="WpfApplication3.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Choix de l’action" Height="300" Width="350"
    ResizeMode="NoResize“
    SizeToContent="Height"
    WindowStartupLocation="CenterScreen"
    WindowStyle="ToolWindow"
    ShowInTaskbar="False"
Foreground="Navy">
```

Apparence de la fenêtre



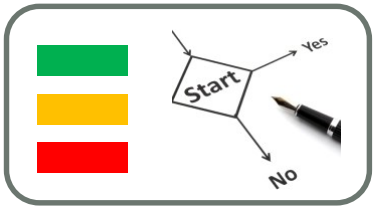
- Pour rendre une fenêtre totalement transparente il faut activer la propriété « AllowsTransparency » en la positionnant à « True ».



AllowsTransparency="True«
Opacity="0.75"

AllowsTransparency="True"
Opacity="1"
Background="Transparent"

Formes géométriques simples



- Rectangle

  Stroke : Changer la couleur des bordures

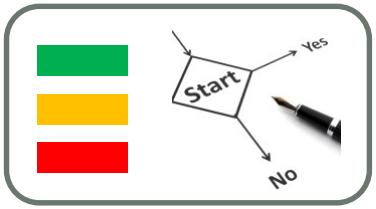
  Fill : Changer la couleur du font

  StrokeTickness : fixer l'épaisseur du trait

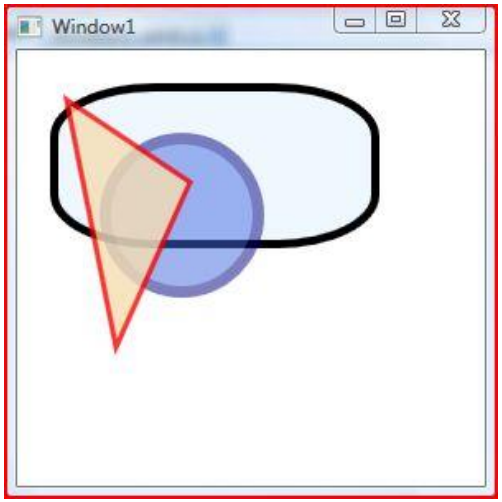
  RadiusX et RadiusY : arrondir les coins

- Ellipse
- Polygone

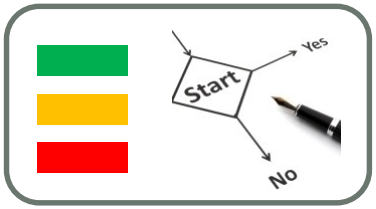
  Courbe fermée reliant des points par des lignes droites. Ces points doivent être définis grâce à la propriété « Points » de l'objet « Polygon ».




```
<Window x:Class="WpfApplication3.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title= "window1" Height="300" Width="350"
    <Canvas>
        <Rectangle Canvas.Left="20" Canvas.Top="20" Height="100" Width="200" Stroke="Black"
        Fill="AliceBlue" StrokeThickness="5" RadiusX="60" RadiusY="30" />
        <Ellipse Canvas.Left="50" Canvas.Top="50" Height="100" Width="100" Stroke="Navy"
        Fill="RoyalBlue" StrokeThickness="7" Opacity="0.5" />
        <Polygon Stroke="Red" StrokeThickness="3" Fill="Wheat" Opacity="0.8" Points="0,0 75,50
        30,150“  Canvas.Left="30" Canvas.Top="30" />
    </Canvas>
</Window>
```



les images



Source de l'image

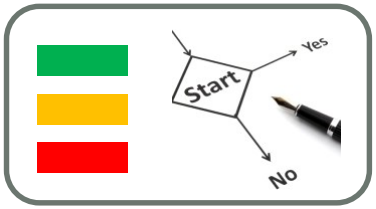
1- disque dur:

```
<Grid>
        <Image Margin="10,10,10,10" />
        <Image Margin="10,10,10,10” Source=“E:\photo.jpg" />
 </Grid>
```

2- Répertoire de l’application:

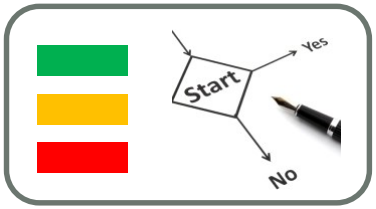
```
<Image Margin="10,10,10,10" Source="Resources\photo.png" />
```

3- Une Adresse Internet:



- La propriété Stretch : Par défaut, cette propriété est fixée à Uniform. Un redimensionnement uniforme entraîne une image qui est agrandie ou rétrécie de manière à utiliser tout l'espace disponible en s'assurant que l'image reste affichée entièrement.
- La valeur UniformToFill conserve, elle aussi, les proportions en utilisant tout l'espace disponible mais elle ne s'assure pas que l'image s'affiche entièrement.

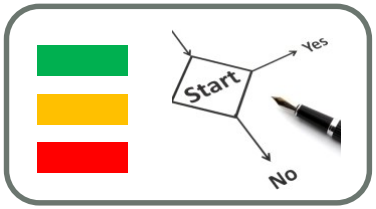
- <Image Margin="10,10,10,10" Source="Resources\photo.png" Stretch="Uniform" />



Ajout d’une image à l’aide du code C#

```
private void Window_Loaded(object sender, RoutedEventArgs e)
 {
        BitmapImage bmp = new BitmapImage();
        bmp.BeginInit();
        bmp.UriSource = new Uri(@"Resources\logo_sdz_fr.png",UriKind.Relative);
        bmp.EndInit(); MonImage.Source = bmp;
 }
```

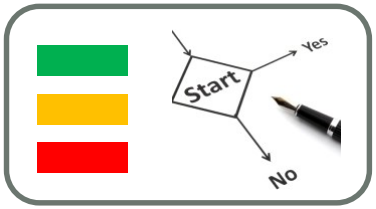
Les boutons



- Nous avons déjà utilisé la propriété Content des boutons pour y spécifier le texte du bouton.

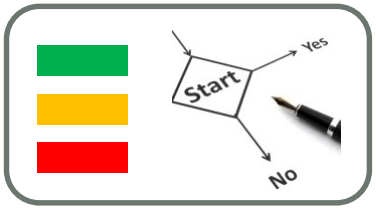
- Cependant, celle-ci accepte n'importe quel type d'objet tout comme la propriété Content des fenêtres. Nous pouvons par exemple y mettre une image, un checkbox...

- Le plus intéressant est de pouvoir mettre un conteneur comme élément Content. Il est ainsi possible de mettre à la fois du texte et des images par exemple.

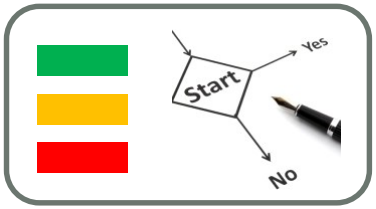



```
<Button Margin="10,10,10,10" >
<StackPanel Orientation="Vertical">
    <Button>
      <Image Height="80" Source="Resources\Burn.png" />
     </Button>
    <TextBlock Text="Un bouton" HorizontalAlignment="Center" FontSize="24" Foreground="Blue" />
    <TextBlock Text="plutôt original non ?" HorizontalAlignment="Center" />
 </StackPanel>
</Button>
```

Propriétés spécifiques des boutons



- La première propriété IsEnabled n'est pas spécifique qu'aux boutons et permet de désactiver les contrôles.
- La deuxième, IsDefault permet de spécifier le bouton par défaut. L'activation de cette propriété permet de pré-valider le bouton. Sous Vista par exemple vous verrez le bouton clignoter lentement. De plus, lorsque cette propriété est utilisée, l'appui sur la touche Entrée permet de valider l'action associée au bouton.
- Enfin, la dernière propriété que nous allons voir est IsCancel. Cette propriété permet de spécifier le bouton d'annulation. La principale caractéristique de cette option est que l'action du bouton pourra être déclenchée par la touche Échap de votre clavier.



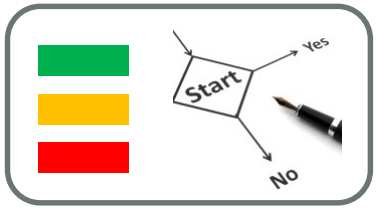

```
<Window x:Class="WpfApplication3.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title= "window1" Height="300" Width="350">
<StackPanel Orientation="Horizontal" >
        <Button Content="Bouton 1" Margin="5" IsDefault="True" Click="Button1_Click" />
        <Button Content="Bouton 2" Margin="5" IsEnabled="False" Click="Button2_Click" />
        <Button Content="Bouton 3" Margin="5" IsCancel="True" Click="Button3_Click" />
</StackPanel>
</Window>
```

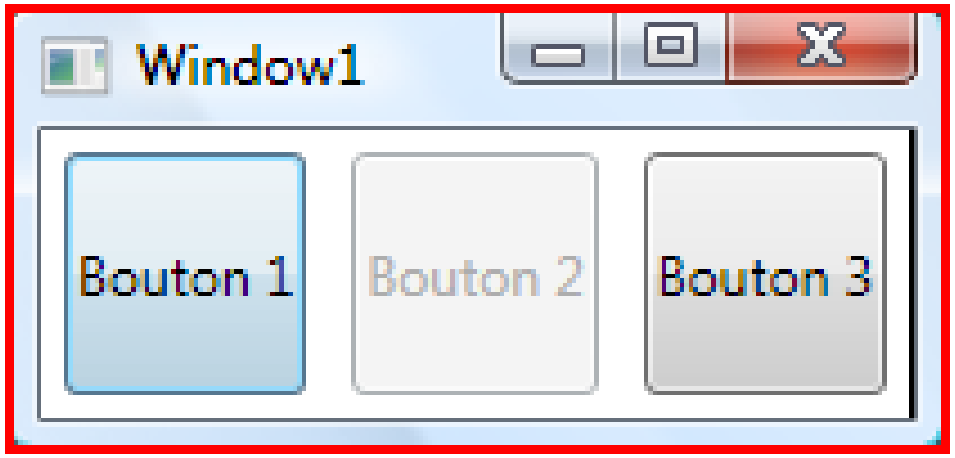
- Code C#

```
private void Button1_Click(object sender, RoutedEventArgs e)
{ MessageBox.Show("Bouton 1"); }

private void Button2_Click(object sender, RoutedEventArgs e)
{ MessageBox.Show("Bouton 2"); }

private void Button3_Click(object sender, RoutedEventArgs e)
 { MessageBox.Show("Bouton 3"); }
```

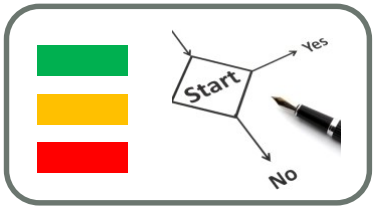




Vous pourrez facilement vérifier le fonctionnement :

- le bouton 1 peut être activé en cliquant dessus, mais aussi via la touche Entrée ;
- le bouton 2 ne peut pas être activé ;
- le bouton 3 peut être activé en cliquant dessus, mais aussi via la touche Échap.

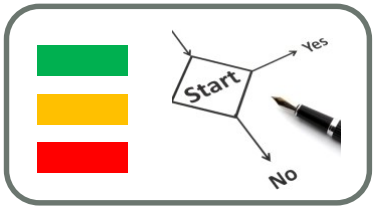
boutons radio



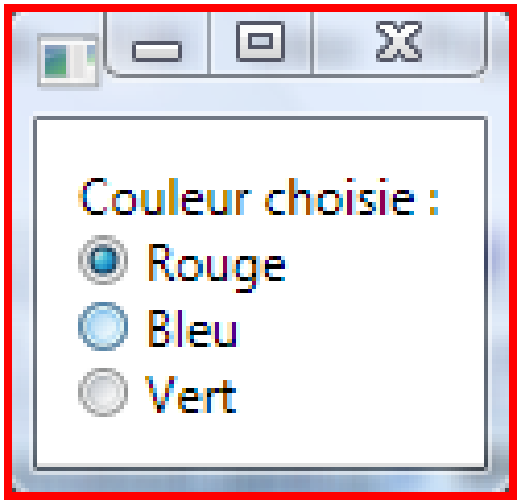
- Les boutons radio permettent de faire un choix parmi plusieurs en ne conservant qu'une seule option active.

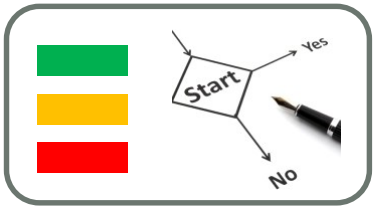
- Les boutons radio se déclarent très simplement grâce à l'objet « RadioButton ».

- La propriété « IsChecked » permet de spécifier l'élément coché par défaut.

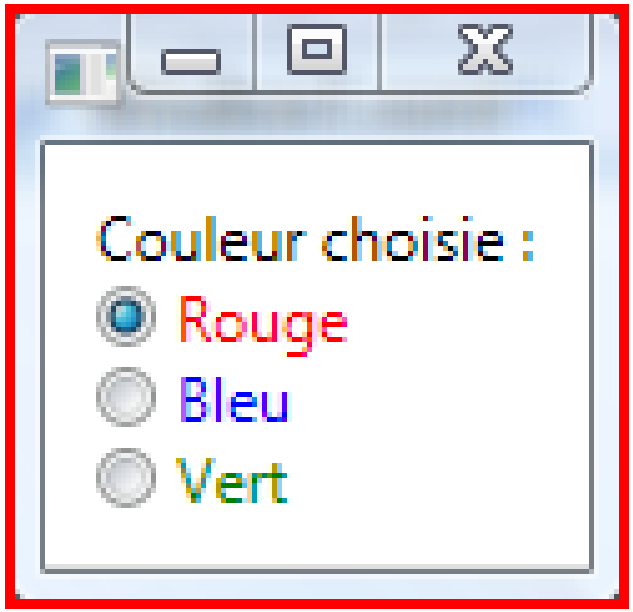


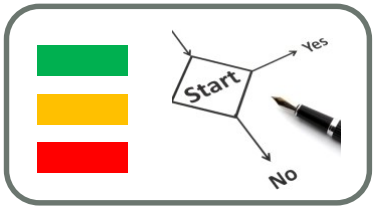

```
<Window x:Class="WpfApplication3.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title= "window1" Height="300" Width="350"
        SizeToContent="WidthAndHeight">
        <StackPanel Orientation="Vertical" Margin="10" >
                <TextBlock Text="Couleur choisie :"/>
                <RadioButton Content="Rouge" IsChecked="True" />
                <RadioButton Content="Bleu" />
                <RadioButton Content="Vert" />
        </StackPanel>
</Window>
```



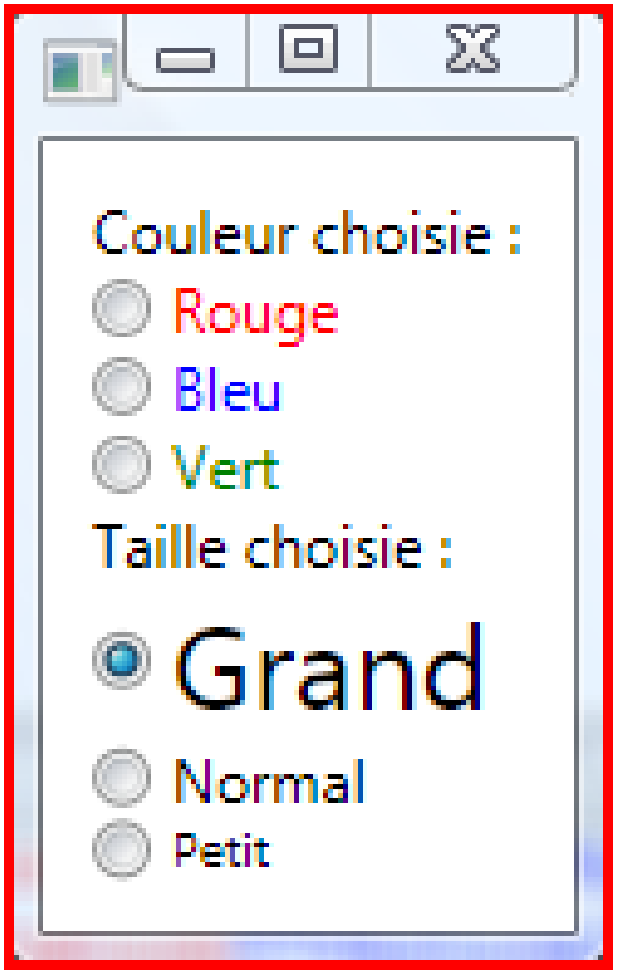


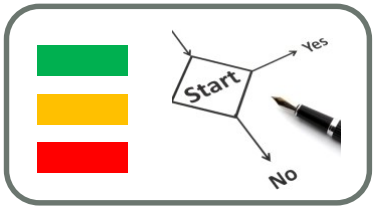
- En ajoutant la propriété Foreground à chacune des boutons d'option, on peut avoir une fenêtre de cette forme
- This.background = System.windows.media.brushes.green;





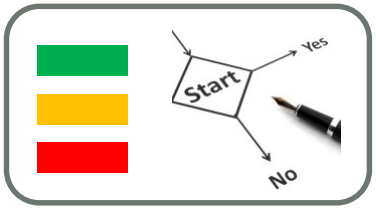

```
<StackPanel Orientation="Vertical" Margin="10" >
                <!-- ZONE DE SÉLECTION DE LA COULEUR -->
                <TextBlock Text="Couleur choisie :"/>
                <RadioButton IsChecked="True">
                                <TextBlock Text="Rouge" Foreground="Red"/>
                </RadioButton>
                <RadioButton>
                                <TextBlock Text="Bleu" Foreground="Blue"/>
                </RadioButton>
                <RadioButton>
                                <TextBlock Text="Vert" Foreground="Green"/>
                </RadioButton>
                <!-- ZONE DE SÉLECTION DE LA TAILLE -->
                <TextBlock Text="Taille choisie :"/>
                <RadioButton IsChecked="True">
                                <TextBlock Text="Grand" FontSize="24" />
                </RadioButton>
                <RadioButton>
                                <TextBlock Text="Normal" FontSize="12" />
                </RadioButton>
                <RadioButton>
                                <TextBlock Text="Petit" FontSize="10" />
                </RadioButton>
</StackPanel>
```





- Par défaut, WPF regroupe ensemble tous les boutons radio qui sont dans le même conteneur.
- Pour avoir des boutons radio existant dans le même conteneur mais avec un traitement différent, on utilise la propriété « GroupName » qui permet de spécifier un nom de groupe.

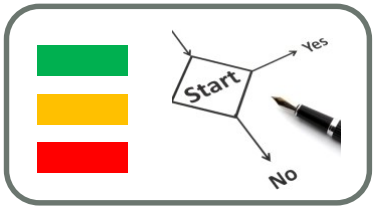
```
<RadioButton IsChecked="True" GroupName="GroupeCouleur">
```



- Il est aussi possible d'utiliser un troisième état pour les boutons radio

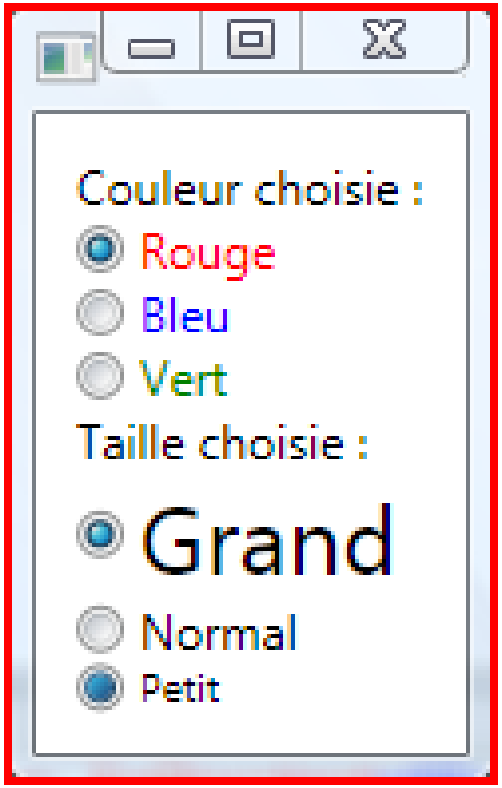
- Par défaut cet état est inaccessible. Pour pouvoir y avoir accès, il faut mettre la propriété « IsThreeState » à « True »

- Lorsque cette option est activée, il est possible d'accéder au troisième état en plaçant la valeur de « IsChecked » à « null »

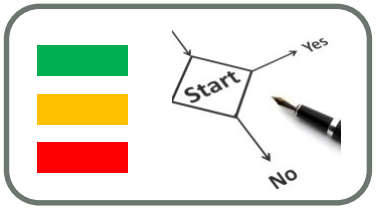



```
<RadioButton GroupName="GroupeTaille" IsThreeState="True" IsChecked="{x:Null}">
        <TextBlock Text="Petit" FontSize="10" />
</RadioButton>
```

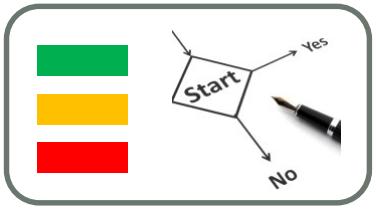
-



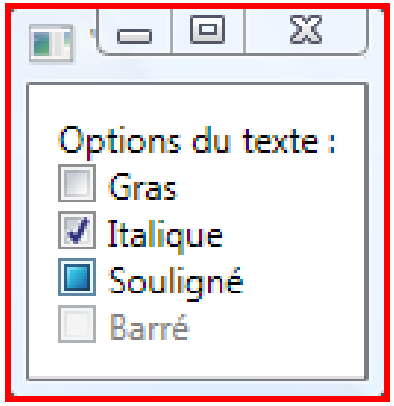
Cases à cocher



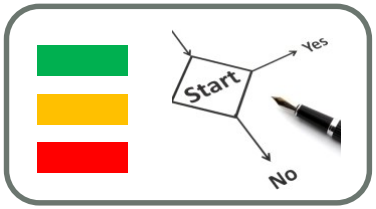
- Les cases à cocher sont très similaires aux boutons radio mis à part qu'il n'y a pas de nom de groupe
- L'activation d'une case se fait grâce à la propriété « IsChecked »
- Tout comme les boutons radios, il est possible de faire passer les éléments dans un troisième état.




```
<Window x:Class="WpfApplication3.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title= "window1" Height="300" Width="350"
        SizeToContent="WidthAndHeight">
        <StackPanel Orientation="Vertical" Margin="10" >
                <TextBlock Text="Options du texte :"/>
                <CheckBox Content="Gras"/>
                <CheckBox Content="Italique" IsChecked="True"/>
                <CheckBox Content="Souligné" IsThreeState="True" IsChecked="{x:Null}"/>
                <CheckBox Content="Barré" IsEnabled="False"/>
        </StackPanel>
</Window>
```

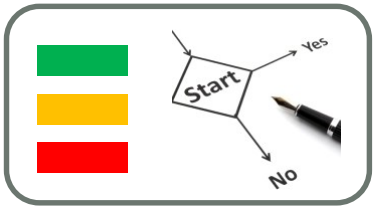


ToolTips

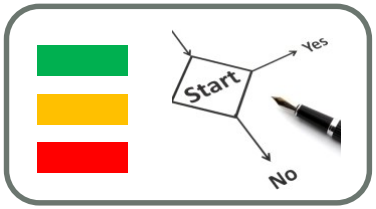


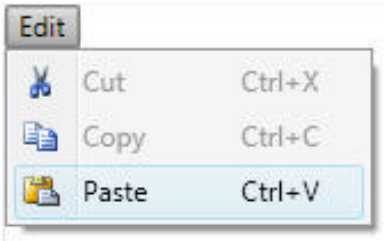

```
<Button Content="Submit">
    <Button.ToolTip>
        <ToolTip>
            <StackPanel>
                <TextBlock FontWeight="Bold">Submit Request</TextBlock>
                <TextBlock>Submits the request to the server.</TextBlock>
            </StackPanel>
        </ToolTip>
    </Button.ToolTip>
</Button>
```

Menu



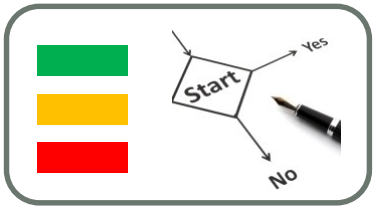

```
<Menu>
        <MenuItem Header="_File">
          <MenuItem Header="_New..." />
          <Separator />
          <MenuItem Header="_Open..." />
          <Separator />
           <MenuItem Header="_Save" /> <MenuItem Header="_Save As..." />
          <Separator />
          <MenuItem Header="_Exit" />
        </MenuItem>
</Menu>
```






```
<MenuItem Header="_Edit">
    <MenuItem Header="_Cut" Command="Cut">
        <MenuItem.Icon>
            <Image Source="Images/cut.png" />
        </MenuItem.Icon>
    </MenuItem>
    <MenuItem Header="_Copy" Command="Copy">
        <MenuItem.Icon>
            <Image Source="Images/copy.png" />
        </MenuItem.Icon>
    </MenuItem>
    <MenuItem Header="_Paste" Command="Paste">
        <MenuItem.Icon>
            <Image Source="Images/paste.png" />
        </MenuItem.Icon>
    </MenuItem>
</MenuItem>
```

Programmation du comportement de l'application

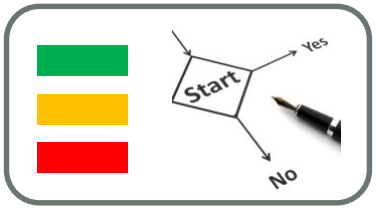


- Chaque contrôle est susceptible de lever des événements dès qu'une interaction se produit
- Exemple : événement lors d'un click sur un bouton

EVENT ET DELEGATE EN C#

Notes de Cours

Notes de Cours

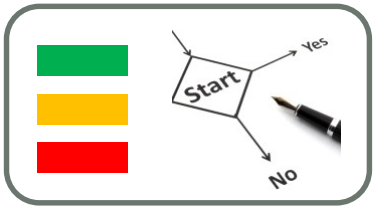


- Le principe de délégation : utilisé pour déléguer l'exécution d'un morceau de programme
- Peut référencer plusieurs méthodes déléguées

- Déclaration d'un type delegate
  - Mot-clé delegate
  - Déclaration des arguments de la méthode déléguée
  - Exemple de déclaration d'un délégué :

```
public delegate int MyDelegateType(int i);
```

Notes de Cours



- Exemple d'utilisation d'un délégué :

Méthode déléguée

Instanciation

Appel

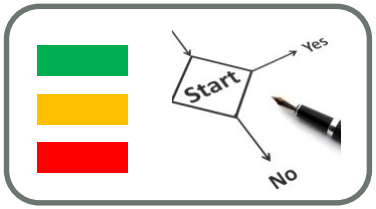
```
public static int Increment(int value);
{ return ++value; }

static void Main(string[] args)
{
    MyDelegateType myDelegate =
        new MyDelegateType(Increment);

    myDelegate(10);

}
```

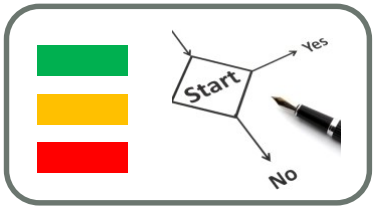
Notes de Cours



- Exemple de référencement de plusieurs méthodes :

```
public static int Increment(int value);
{ return ++value; }
public static void Show(int value);
{
    Console.WriteLine("Valeur : {0}",
        value.ToString());
}
static void Main(string[] args)
{
    MyDelegateType myDelegate =
        new MyDelegateType(Increment);
    myDelegate += new
        MyDelegateType(Show);
    myDelegate(10);
}
```

Notes de Cours

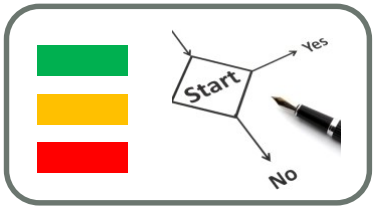


- Evènements

- Permettent de capturer une action du programme
- Méthode standard pour la gestion d'évènements
- Basé sur la délégation
  - Mot-clé event
  - Signature fixe :

```
void OnEventName(object sender, EventArgs);
```

Notes de Cours



Evènements

Déclaration du délégué :

```
public delegate
    void NewInfoHandler(object sender,
        EventArgs e);
```

Déclaration de l'évènement :

```
public event NewInfoHandler InfoHandler;
```

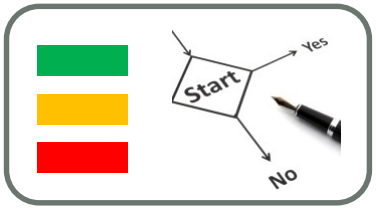
Abonnement :

```
InfoHandler += new NewInfoHandler();
```

PREMIÈRES MANIPULATIONS

Windows Phone

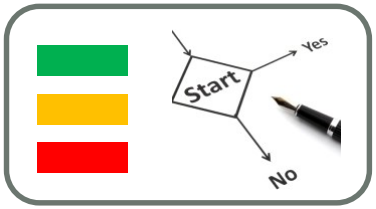
Equpements des Travaux Pratiques : Equipements Matériels



- Windows Phone HTC 8S

- Upgrade en Windows Phone 8.0 (ou 8.1) à faire en premier lieu à partir d'une connexion Wifi sur le téléphone

Equipements des Travaux Pratiques : Windows Phone

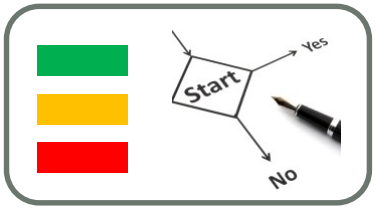


- Environnement logiciel :
  - Windows 8 ou 8.1 sur votre PC
  - Visual Studio Professional ou Ultimate 2013
  - Mobile Phone sous Windows Phone 8 ou 8.1

- Installation possible depuis un compte personnel « élève » Microsoft
- Installation possible depuis votre compte académique alliance « DreamSpark »

- Si vous n'avez pas de compte DreamSpark, me demander une ouverture

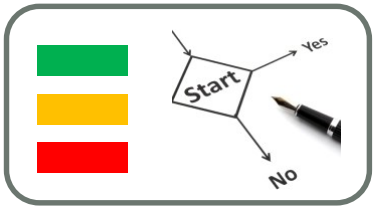
Test de l'environnement de travail sous Windows Phone



- Sous votre PC sous Windows 8, installez Visual Studio 2013 Pro (normalement c'est fait)

- Sous Visual Studio 2013 Pro, développer votre premier projet Windows Phone, exécuté sous simulateur
- Configurez la connexion Wifi de votre HTC 8S et upgradez son système vers Windows Phone 8.0
- Exécutez votre précédent projet sur la cible HTC 8S (si vous avez d'autres Windows Phone sous WP8, n'hésitez pas à tester)

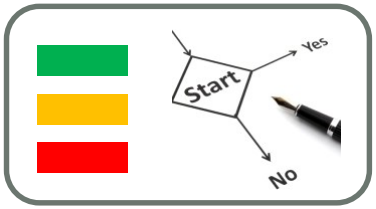
Manipulations



- A partir du tutorial en ligne de Nicolas Hilaire

- Créez votre première application « Hello World » en :
- Modifiant le XAML de votre projet pour son rendu graphique
- Modifiant le code C# de la logique de contrôle de l'application

La mise en œuvre d'autres E/S



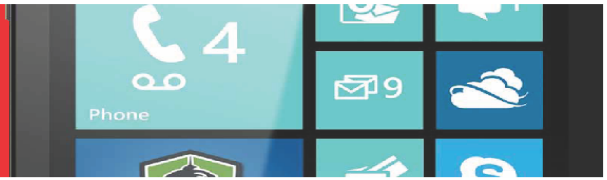
- Nombre d'APIs avec leur namespace correspondant sont disponibles sur Windows Phone 8.X.
- Un résumé de la liste se trouve sur
  - http://cmsresources.windowsphone.com/devcenter/en-us/downloads/064028-microsoft-poster.pdf

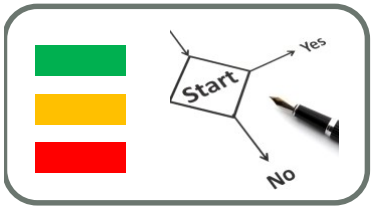
- Les documentations détaillées se trouvent sur http://msdn.microsoft.com/en-us/library/

- Des exemples d'illustration et de test se trouvent sur http://code.msdn.microsoft.com/site/search?f[0].Type=Technology&f[0].Value=Windows%20Phone%208&f[1].Type=VisualStudioVersion&f[1].Value=12.0&f[1].Text=Visual%20Studio%202013

Windows Phone

Windows Phone API QuickStart





Fundamental Types and Date / Time

Windows Runtime
Windows.Foundation[+▲]
DateTime, Uri
Windows.Foundation.Collections[+▲]
IIterable<T>, IVector<T>, IMap<TK, TV>, IVectorView<T>
Windows.Storage.Streams[+]
Buffer

.NET
System[+]
Object, Byte, Char, Int32, Single, Double, String, Array, Buffer, Uri, DateTime, TimeSpan, Convert, Math, Random, Type, Exception, Action<T..>, Func<T..>
System.Collections[+]
List<T>, Dictionary<TK,TV>, Queue<T>, Stack<T>
System.Collections.ObjectModel[+]
Collection<T>, ObservableCollection<T>
System.Linq
Note: Contains extension methods; no need to use specific types
System.Runtime.InteropServices.WindowsRuntime
WindowsRuntimeBuffer, WindowsRuntimeBufferExtensions[+]
Note: You can also use standard C# and VB types and libraries, 'dynamic' types, and 'async' methods via 'await'

Native
Platform[+]
Object, String, Array<T>, Exception
Platform::Collections[+]
Vector<T>, VectorView<T>, Map<TK,TV>, MapView<TK,TV>, VectorIterator<T>
Date & Time[+]
GetLocalTime, GetSystemTime, GetDynamicTimeZoneInformation, SystemTimeToFileTime, FileTimeToSystemTime
Note: You can also use standard C types and the C++ standard libraries

Core App Framework, Memory & Debugging

Windows Runtime
Windows.ApplicationModel.Core[■▲]
CoreApplication
Windows.Graphics.Display[■▲]
DisplayProperties
Windows.Phone.UI.Core[▲]
KeyboardInputBuffer
Windows.Phone.UI.Input[▲]
HardwareButtons
Windows.UI.Core[■▲]
CoreWindow

.NET
Microsoft.Phone.Controls
PhoneApplicationFrame, PhoneApplicationPage
Microsoft.Phone.Shell
PhoneApplicationService, ApplicationBar, SystemTray
System.Diagnostics[+]
Debug, Debugger
System.Windows
Application, MessageBox, ResourceDictionary
System.Windows.Navigation
NavigationService, NavigationContext, UriMapper

Native
COM[+]
CoInitializeEx, CoUninitialize, CoCreateInstanceFromApp, StringFromCLSID, CoTaskMemFree, CoFreeUnusedLibrariesEx
Memory & Libraries[+]
GetProcessHeap, HeapAlloc, HeapFree, LoadPackagedLibrary, GetProcAddress, FreeLibrary
Note: You can also use C++ new and delete operators
Errors & Debugging[+]
GetLastError, IsDebuggerPresent, OutputDebugStringW

Text, Globalization & Resources

.NET
Microsoft.Phone.Globalization
SortedLocaleGrouping
System.Globalization[■]
CultureInfo, RegionInfo
System.Resources
ResourceManager
System.Text[+]
StringBuilder, Encoding
System.Text, RegularExpressions[+]
Regex
System.Windows
Application.GetResourceStream()

Native
Text & Resources[+]
FormatMessageW, MultiByteToWideChar, WideCharToMultiByte, CompareStringEx
NLS[+]
GetDateFormatEx, GetTimeFormatEx, GetUserDefaultLocaleName, GetLocaleInfoEx
MUI
GetUserPrefferedUILanguages

Threading and Synchronization

Windows Runtime
Windows.System.Threading[+▲]
ThreadPool, ThreadPoolTimer
Windows.UI.Core[+]
CoreDispatcher, CoreWindow,
Note: You can get the current dispatcher from CoreWindow.GetForCurrentThread()

.NET
System
WindowsRuntimeSystemExtensions[+]
System.Threading[■]
Thread, ThreadPool, Timer, AutoResetEvent, ManualResetEvent, Mutex
System.Threading.Tasks[+]
Task<T>, TaskCompletionSource<T>, TaskFactory<T>
System.Windows.Threading
Dispatcher, DispatcherTimer
Note: You can get the global dispatcher from System.Windows.Deployment.Current

Native
Threads[+]
GetCurrentThread, GetCurrentThreadId
Synchronization[+]
WaitForSingleObjectEx, WaitForMultipleObjectsEx, CreateMutexExW, ReleaseMutex, CreateEventExW, SetEvent, ResetEvent, CloseHandle, InitializeCriticalSectionEx, EnterCriticalSection, LeaveCriticalSection, DeleteCriticalSection

File System and Streams

Windows Runtime
Windows.Storage[+]
ApplicationData, StorageFile, StorageFolder
Note: You can get install folder from Windows.ApplicationModel.Package.Current.InstalledLocation
Windows.Storage.FileProperties[+]
BasicProperties
Windows.Storage.Streams[■]
DataReader, DataWriter

.NET
Microsoft.Phone.Storage
ExternalStorage
System.IO[+]
StreamReader, StreamWriter, StringReader, StringWriter, BinaryReader, BinaryWriter, MemoryStream, WindowsRuntimeStorageExtensions[+], WindowsRuntimeStreamExtensions[+]
Note: Avoid using the File, FileInfo, Directory, DirectoryInfo, and Path types
System.IO.IsolatedStorage
IsolatedStorageFile

Native
File I/O[+]
CreateFile2, ReadFile, WriteFile, DeleteFileW, CloseHandle, FindFirstFileExW, FindClose

Networking, Web and Proximity

Windows Runtime
Windows.Networking
HostName
Windows.Networking.Proximity[+]
PeerFinder, ProximityDevice
Windows.Networking.Sockets[■]
DatagramSocket, StreamSocket

.NET
Microsoft.Phone.Controls
WebBrowser, WebBrowserExtensions[+]
Microsoft.Phone.Tasks
WebBrowserTask
System.Net[■]
WebClient, HttpWebRequest, WebUtility
System.Net.Sockets
Socket
Note: Provided for compatibility; new apps should use Windows.Networking.Sockets

Native
HTTP[+]
IXMLHTTPRequest2
Winsock
WSAStartup, socket, gethostbyname, WSAAddressToString, bind, connect, send, recv, accept, listen, closesocket, WSACleanup

Media, Sounds and Pictures

Windows Runtime
Windows.Phone.Media.Capture
PhotoCaptureDevice, AudioVideoCaptureDevice, KnownCameraPhotoProperties
Windows.Storage.Pickers[■]
FileOpenPicker
Note: Can only be used to pick photos from the Photos Hub

cont.

.NET
Microsoft.Devices
PhotoCamera, CameraButtons, CameraVideoBrushExtensions[+]
Microsoft.Phone
PictureDecoder
Microsoft.Phone.Tasks
MediaPlayerLauncher, CameraCaptureTask, PhotoChooserTask, ShareMediaTask, SaveRingtoneTask
Microsoft.Xna.Framework.Audio
Microphone, SoundEffect, DynamicSoundEffectInstance
Microsoft.Xna.Framework.Media
MediaLibrary, MediaPlayer, Song
Microsoft.Xna.Framework.Media.PhoneExtensions
MediaLibraryExtensions[+]
System.Windows.Media
CaptureSource, CaptureDeviceConfiguration, MediaStreamSource, VideoBrush
System.Windows.Media.Imaging
WriteableBitmap, BitmapImage, Extensions[+]
Note: See Messaging, Sharing & Miscellaneous for key DRM APIs

Native
Media Foundation[■]
MFSStartup, IMFMediaEngineClassFactory, IMFMediaEngine, MFShutdown
XAudio2[+]
XAudio2Create, X3DAudioInitialize, X3DAudioCalculate
CaptureDevice Native APIs
ICameraCaptureDeviceNative, IAudioVideoCaptureDeviceNative
Note: These are accessible from the Windows Runtime objects

XML, Databases and Serialization

.NET
System.Data.Linq
DataContext
System.Data.Linq.Mapping
TableAttribute, ColumnAttribute
System.Runtime.Serialization[+]
DataContractSerializer
System.Runtime.Serialization.Json[+]
DataContractJsonSerializer
System.Xml[+]
XmlReader, XmlWriter
System.Xml.Linq
XDocument, XElement, XAttribute, XNamespace, Extensions
System.Xml.Serialization[■]
XmlSerializer

In-App Purchase, Licensing and Marketplace

Windows Runtime
Windows.ApplicationModel.Store[■]
CurrentApp
Windows.System
Launcher.LaunchUriAsync[▲]

.NET
Microsoft.Phone.Marketplace
LicenseInformation
Microsoft.Phone.Tasks
MarketplaceDetailsTask, MarketplaceReviewTask, MarketplaceSearchTask

Maps, Location and Sensors

Windows Runtime
Windows.Devices.Geolocation[+]
Windows.Devices.Sensors[■]
Accelerometer, Compass, Gyrometer, OrientationSensor
Windows.Phone.Devices.Notification
VibrationDevice

.NET
Microsoft.Devices
VibrateController
Microsoft.Devices.Sensors
Accelerometer, Compass, Gyroscope, Motion

cont.

Microsoft.Phone.Maps.Controls
Map, MapLayer, MapOverlay,
Microsoft.Phone.Maps.Services
GeocodeQuery, ReverseGeocodeQuery, RouteQuery
Microsoft.Phone.Tasks
MapsTask, MapsDirectionsTask, MapDownloaderTask
System.Device.Location
GeoCoordinateWatcher

VoIP[▲]

Windows Runtime
Windows.Phone.Media.Capture
AudioVideoCaptureDevice, KnownCameraAudioVideoProperties
Windows.Phone.Media.Devices
AudioRoutingManager
Windows.Phone.Networking.Voip
VoipCallCoordinator, VoipPhoneCall

.NET
Microsoft.Phone.Media
MediaStreamerFactory
Microsoft.Phone.Networking.Voip
VoipBackgroundProcess, VoipHttpIncoming-CallTask, VoipKeepAliveTask
See also: Multitasking

Native
Audio
ActivateAudioInterface, GetDefaultAudioCaptureId, GetDefaultAudioRenderId,

Controls and Animation (XAML)

.NET
Microsoft.Advertising.Mobile.UI[▼]
AdControl
Microsoft.Expression.Interactivity.Core[▼]
GoToStateAction, NavigateToPageAction, CallMethodAction, ChangePropertyAction,
Microsoft.Expression.Interactivity.Media[▼]
ControlStoryboardAction, PlaySoundAction
Microsoft.Phone.Controls
LongListSelector, Panorama, Pivot, WebBrowser
Microsoft.Phone.Maps.Controls
Map, MapLayer, MapOverlay,
System.Windows.Controls
Button, CheckBox, RadioButton, HyperlinkButton, TextBlock, TextBox, PasswordBox, RichTextBox, Image, MediaElement, ProgressBar, Border, Canvas, Grid, ListBox, StackPanel, ScrollViewer, ViewBox
System.Windows.Controls.Primitives
Popup, ViewportControl
System.Windows.Interactivity[▼]
Interaction, InvokeCommandAction,
System.Windows.Media.Animation
Storyboard, DoubleAnimation, DoubleAnimationUsingKeyFrames, DiscreteDoubleKeyFrame, ObjectAnimationUsingKeyFrames, DiscreteObjectKeyFrame
System.Windows.Shapes
Rectangle, Ellipse, Polygon
Note: Use Visual Studio or Expression Blend for layout. Use Expression Blend to add interactivity, animation, and visual states.

3D Graphics[▲]

.NET
System.Windows.Controls
DrawingSurface, DrawingSurfaceBackgroundGrid
XNA
Note: XNA is fully supported for Windows Phone 7.5 apps only; see documentation.

Native
Direct3D
D3D11CreateDevice, CreateSwapChainForCoreWindow, ID3D11Device1, ID3D11DeviceContext1, IDXGISwapChain1
Note: See documentation for complete list of D3D interfaces and APIs

Speech

Windows Runtime
Windows.Phone.Speech.Recognition
SpeechRecognizerUI, SpeechRecognizer, InstalledSpeechRecognizers
Windows.Phone.Speech.Synthesis
SpeechSynthesizer, InstalledVoices
Windows.Phone.Speech.VoiceCommands
VoiceCommandService

Wallet[▲]

.NET
Microsoft.Phone.Tasks
AddWalletItem
Microsoft.Phone.Wallet
Wallet, Deal, WalletTransactionItem, WalletAgent

Multitasking[▲]

.NET
Microsoft.Phone.BackgroundAudio
BackgroundAudioPlayer, AudioTrack, AudioPlayerAgent, AudioStreamingAgent
Microsoft.Phone.BackgroundTransfer
BackgroundTransferService, BackgroundTransferRequest
Microsoft.Phone.Networking.Voip
VoipCallInProgressAgent, VoipForegroundLifetimeAgent
Microsoft.Phone.Scheduler
ScheduledActionService, PeriodicTask, ResourceIntensiveTask
Microsoft.Phone.Shell
ShellToast
Microsoft.Phone.Wallet
WalletAgent

Push, Live Tiles and Lock Screen

Windows Runtime
Windows.Phone.System.UserProfile
LockScreen, LockScreenManager

.NET
Microsoft.Phone.Notification
HttpNotificationChannel
Microsoft.Phone.Shell
ShellTile, FlipTileData, IconicTileData, CycleTileData,

Contacts, Appointments and Reminders

Windows Runtime
Windows.Phone.PersonalInformation
ContactStore, StoredContact, ContactInformation, KnownContactProperties

.NET
Microsoft.Phone.Scheduler
Alarm, Reminder
Microsoft.Phone.Tasks
PhoneCallTask, PhoneNumberChooserTask, EmailAddressChooserTask, AddressChooserTask, SaveAppointmentTask, SaveContactTask,
Microsoft.Phone.UserData
Contacts, Appointments

Messaging, Sharing and Miscellaneous

Windows Runtime
Windows.ApplicationModel.DataTransfer[■▲]
DataPackage, DataTransferManager
Windows.Phone.Devices.Power
Battery
Windows.Phone.Management.Deployment
InstallationManager
Windows.Phone.Storage.SharedAccess
SharedStorageAccessManager
Windows.Phone.System.Analytics
HostInformation
Windows.Phone.System.Memory
MemoryManager
Windows.Phone.System.Power
PowerManager
Windows.System
Launcher[▲]

.NET
Microsoft.Phone.Info
DeviceStatus
Microsoft.Phone.Tasks
PhoneCallTask, SmsComposeTask, EmailComposeTask, ShareLinkTask, ShareStatusTask, ShareMediaTask, SearchTask, ConnectionSettingsTask
System.Security.Cryptography
ProtectedData, RNGCryptoServiceProvider, RSACryptoServiceProvider, SHA256Managed, AesManaged
System.Security.Cryptography.X509Certificates
X509Certificate
System.Windows.Media[+▲]
DomainAcquirer, LicenseAcquirer

Native
Hardware
IsProcessorFeaturePresent

Additional Frameworks:

Windows Phone Toolkit
http://phone.codeplex.com
Microsoft Media Player Framework
http://smf.codeplex.com/
Smooth Streaming Client
http://www.iis.net/downloads/microsoft/smooth-streaming-client-sdk

NuGet Package Manager
http://nuget.codeplex.com/
DirectX Helpers
http://directxtex.codeplex.com/,
http://directxtk.codeplex.com/
OData Client
http://msdn.microsoft.com/en-us/jj658961

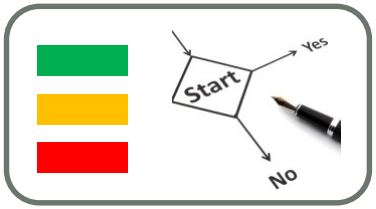
See Also:

- NFC -> Networking, Web, & Proximity
- Bluetooth -> Networking, Web, & Proximity
- GPS -> Maps, Location & Sensors; Multitasking
- Background audio -> Multitasking; Media
- Alarms & Reminders -> Multitasking; Contacts, Appointments & Reminders
- Calendar -> Contacts, Appointments & Reminders
- E-mail & SMS -> Messaging, Sharing & Miscellaneous
- Facebook & Twitter -> Messaging, Sharing & Miscellaneous
- Rumble / Vibration -> Maps, Location & Sensors
- Battery & Power -> Messaging, Sharing & Miscellaneous
- Memory -> Messaging, Sharing & Miscellaneous
- App launching -> Messaging, Sharing & Miscellaneous
- Ringtones -> Media, Sounds & Pictures
- Security -> Messaging, Sharing & Miscellaneous
- DRM -> Messaging, Sharing & Miscellaneous

Legend

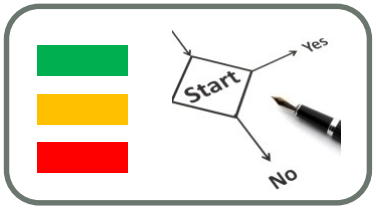
- Please see the docs for more info
- Same or very similar to Windows 8
- Mostly overlaps with Windows 8
- Limited overlap with Windows 8
- Supported in native apps only
- Extension methods
- Similar to Windows.UI.Xaml namespace in Windows
- Use Add Reference → Extensions dialog to add to project

Les informations collectables sur le HTC S8



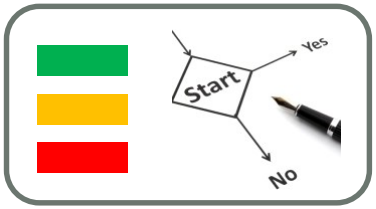
- Capteurs :
  - GPS intégré (A-GPS)
  - Accéléromètre
  - Boussole numérique
  - Détecteur de proximité
  - Détecteur de luminosité ambiante

Les informations collectables sur le HTC S8



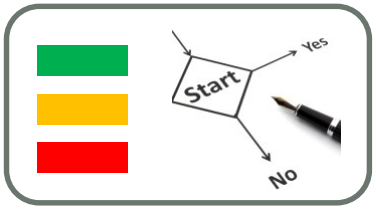
- Mais si on élargit à l'ensemble des données collectable sur le smartphone et qui peuvent caractériser l'activité d'un utilisateur alors :
  - Microphone
  - Liste des numéros de téléphone appelés, reçus, les horaires
  - Les SSID des wifi à proximité
  - Les ID des device Bluetooth à proximité (et leur nom avec SDP, service discovery protocol)
  - autre ?

Et bien d'autres APIs intéressantes …



- Texttospeech
- SpeechRecognition
- Web Service REST / SOAP
- UPnP Device
- …

Vos projets



- Choisissez et testez l'API qui sera nécessaire à votre projet :

- Pour la collecte de données
- Pour un minimum d'interactions avec l'utilisateur

- Tout en gérant la consommation d'une application qui devrait être permanente (du moins exécutée sur de longue période)

- Quelques infos sur : http://www.codeproject.com/Articles/28886/Windows-Mobile-Power-Management
- Un projet intéressant sur : http://www.microsoft.com/en-us/download/details.aspx?id=19400
- Manage Lifecycle and State of App : http://msdn.microsoft.com/en-us/library/windows/apps/hh986968.aspx
- Windows Phone 8 Battery API : http://mobile.dzone.com/articles/windows-phone-8-battery-api