```
<Button Content="Prof-MTI">
    <Button.Triggers>
        <EventTrigger RoutedEvent="Button.Click">
            <EventTrigger.Actions>
                <BeginStoryboard Storyboard="{StaticResource StudentLearning}"/>
```

COURS WPF
PART 3

Lemettre Arnaud

Arnaud.lemettre@gmail.com

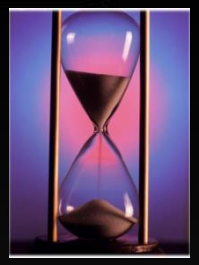
ÉCOLE D'INGÉNIEURS EN INFORMATIQUE

Sommaire

- Les animations
- Blend 3.0
- Sketchflow
- Avancé
  - DataConverter

Les Animations

Les Animations



- Les animations permettent de changer la valeur d'une propriété au cours du temps ou sur une période donnée.
- Elles permettent de créer une grande variété d'effets.
- On peut animer :
  - Une taille
  - Une couleur
  - Un déplacement
  - Ou n'importe quelle propriété d'un élément accessible

Les Animations

- Il existe trois types d'animations :
  - Linear Animation
  - Key frame based Animation
  - Path based Animation

Les Animations

- Linear Animation :

  Permet de changer une animation de façon linéaire.

  On rencontrera généralement les DoubleAnimations ou les ColorAnimations qui sont toutes les deux des linear Animation

  Généralement ces dernières sont les plus utilisées

Les Animations

- Key Frame based Animation :

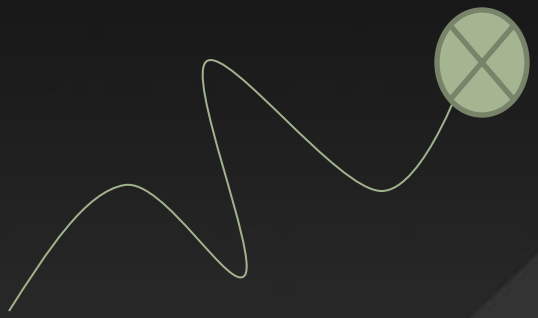
  Permet d'effectuer une animation en passant par certains « points de passages », les key frame. Ces clés permettent d'effectuer certains changements sur plusieurs composants en les synchronisant par exemple.

  La progression de l'animation se fait de manière linéaire. Cependant on verra par la suite que les animations ne sont pas forcément linéaires.

Les Animations

- Path based Animation :

  Cela permet d'utiliser un chemin comme guide pour réaliser un mouvement d'animation.



Les Animations

- Les propriétés importantes :
  - AutoReverse : boolean permet de refaire l'animation dans le sens inverse dès que celle-ci est finie
  - Duration : le temps que durera l'animation
  - From : la valeur de départ pour l'animation
  - To: la valeur d'arrivée

Linear Animation

- Il y a encore beaucoup d'autres propriétés, je vous laisse le soin de les explorer

Les Animations

Exemple de déclaration d'animation , à mettre dans le window.resources :

```
<DoubleAnimation Duration="00:00:10" From="1" To="200" />
```

C'est bien, mais par contre on ne peut pas encore l'utiliser de cette manière il manque encore une chose importante : le **storyboard**

Le storyboard est l'objet qui organise les animations. Ces fils peuvent contenir plusieurs animations. De plus les storyboards permettent d'indiquer quelle propriété va subir l'animation

En XAML, la déclaration d'animation doit forcément être contenue dans un storyboard.

Les Animations

Utilisation des storyboards :

```
<Storyboard x:Key="sbButton" TargetName="button1"
    TargetProperty="Height">
     <DoubleAnimation Duration="00:00:10" From="1" To="200" />
</Storyboard>
<!--ou-->
<Storyboard x:Key="sbButton1" >
     <DoubleAnimation Duration="00:00:10" From="30" To="200"
    Storyboard.TargetProperty="Height"/>
</Storyboard>
```

Propriété non obligatoire, si on ne la met pas, nous avons une animation générique

La deuxième façon est la plus utilisée car elle permet d'effectuer pour un storyboard plusieurs animations sur plusieurs propriétés d'un composant.

Et maintenant comment l'utiliser ??

Les Animations

Utilisation des storyboards :
Maintenant , il faut pouvoir déclencher les animations : pour cela on va utiliser les triggers. (Cf part1)

```
<Button Content="button2" Grid.Column="1" Height="30" Width="100" >
    <Button.Triggers>
        <EventTrigger RoutedEvent="Button.Click">
            <EventTrigger.Actions>
                <BeginStoryboard Storyboard="{StaticResource sbButton1}"/>
            </EventTrigger.Actions>
        </EventTrigger>
    </Button.Triggers>
</Button>
```

Bien entendu les triggers peuvent également être contenus dans des styles, et les « RoutedEvent » peuvent être tous les événements du composant sur lequel le trigger va être posé

Les Animations

On peut également avoir un contrôle sur les animations , comme par exemple les stopper.
La liste des actions est la suivante :

- BeginStoryBoard
- PauseStoryBoard
- ResumeStoryBoard
- StopStoryBoard
- …

```
<Button Content="button2" Grid.Column="1" Height="30" Width="100" >
    <Button.Triggers>
        <EventTrigger RoutedEvent="Button.MouseEnter">
            <EventTrigger.Actions>
                <BeginStoryboard x:Name="beginSb" Storyboard="{StaticResource
    sbButton1}"/>
            </EventTrigger.Actions>
        </EventTrigger>
        <EventTrigger RoutedEvent="Button.MouseLeave">
            <EventTrigger.Actions>
                <StopStoryboard BeginStoryboardName="beginSb"/>
            </EventTrigger.Actions>
        </EventTrigger>
    </Button.Triggers>
</Button>
```

Les Animations

Un exemple avec une animation de couleur :
Dans les ressources :

```
<Storyboard x:Key="sbColor">
    <ColorAnimation Duration="00:00:05" From="Red" To="LimeGreen"
    Storyboard.TargetProperty="Color" Storyboard.TargetName="myBrush"/>
</Storyboard>
```

Sur un bouton :

```
<Button Content="button1" Height="30" Width="100" x:Name="button1">
    <Button.Background>
        <SolidColorBrush x:Name="myBrush"/>
    </Button.Background>
    <Button.Triggers>
        <EventTrigger RoutedEvent="Button.Click">
            <EventTrigger.Actions>
                <BeginStoryboard Storyboard="{StaticResource sbColor}"/>
            </EventTrigger.Actions>
        </EventTrigger>
    </Button.Triggers>
</Button>
```

Les Animations

Nous allons reprendre le TP de la semaine dernière sur le cube en 3D afin d'appliquer une animation dessus. Dans les zones de ressources :

```
<Window.Resources>
    <Storyboard x:Key="myStoryBoardX">
        <DoubleAnimation Storyboard.TargetName="rotationX"
    Storyboard.TargetProperty="Angle"
    From="0" To="360" Duration="0:0:15" RepeatBehavior="Forever"/>
    </Storyboard>
    <Storyboard x:Key="myStoryBoardY">
        <DoubleAnimation Storyboard.TargetName="rotationY"
    Storyboard.TargetProperty="Angle"
    From="0" To="360" Duration="0:0:15" RepeatBehavior="Forever"/>
    </Storyboard>
</Window.Resources>
```

Les Animations

On veut déclencher l'animation au moment du chargement de la fenêtre => on utilise donc les triggers :

```
<Window.Triggers>
    <EventTrigger RoutedEvent="Window.Loaded" >
        <BeginStoryboard Storyboard="{StaticResource myStoryBoardX}"/>

        <BeginStoryboard Storyboard="{StaticResource myStoryBoardY}"/>
    </EventTrigger>
</Window.Triggers>
```

Les Animations

Au niveau du cube :

```
<ModelVisual3D.Transform>
    <Transform3DGroup>
        <RotateTransform3D>
            <RotateTransform3D.Rotation>
                <!--rajout animation-->
                <AxisAngleRotation3D  x:Name="rotationY" Angle="0" Axis="0,10,0" />
            </RotateTransform3D.Rotation>
        </RotateTransform3D>
        <RotateTransform3D>
            <RotateTransform3D.Rotation>
                <!--rajout animation-->
                <AxisAngleRotation3D  x:Name="rotationX" Angle="0" Axis="10,0,0" />
            </RotateTransform3D.Rotation>
        </RotateTransform3D>
    </Transform3DGroup>
</ModelVisual3D.Transform>
```

Les Animations

Avec WPF 4 les animations sont un peu plus simples :

```
<Rectangle Name="MyRectangle" Margin="60" Width="50" Height="50" Fill="Blue">
    <Rectangle.Triggers>
        <EventTrigger RoutedEvent="Rectangle.MouseDown">
            <BeginStoryboard>
                <Storyboard>
                    <DoubleAnimation From="1" To="2" Duration="00:00:1"
             Storyboard.TargetName="myScaleTransform"
             Storyboard.TargetProperty="ScaleX">
                        <DoubleAnimation.EasingFunction>
                            <BackEase Amplitude="0.3" EasingMode="EaseInOut" />
                        </DoubleAnimation.EasingFunction>
                    </DoubleAnimation>
                </Storyboard>
            </BeginStoryboard>
        </EventTrigger>
    </Rectangle.Triggers>
    <Rectangle.RenderTransform>
        <ScaleTransform x:Name="myScaleTransform" />
    </Rectangle.RenderTransform>
</Rectangle>
```

Exemple de code permettant d'insérer directement un effet sur l'animation

Avec l'ancienne méthode nous aurions été obligés de coder l'effet de façon manuelle

Les Animations

- Il existe bien entendu un grand nombre d'effets prédéfinis :
  - BackEase
  - BounceEase
  - CircleEase
  - CubicEase
  - ElasticEase
  - ExponentialEase
  - PowerEase
  - QuadraticEase
  - QuarticEase
  - QuinticEase
  - SinceEase

Liens vers des exemples en live

BLEND 3.0

BLEND 3.0

- **Microsoft Expression Blend** est l'outil professionnel de conception d'interfaces utilisateurs de Microsoft. Il est utilisé pour créer des interfaces graphiques riches orientées Web pour des applications de bureau mêlant le meilleur du Web et du poste de travail. Expression Blend est écrit en utilisant le .NET Framework 3.0 et le Windows Presentation Foundation (WPF).

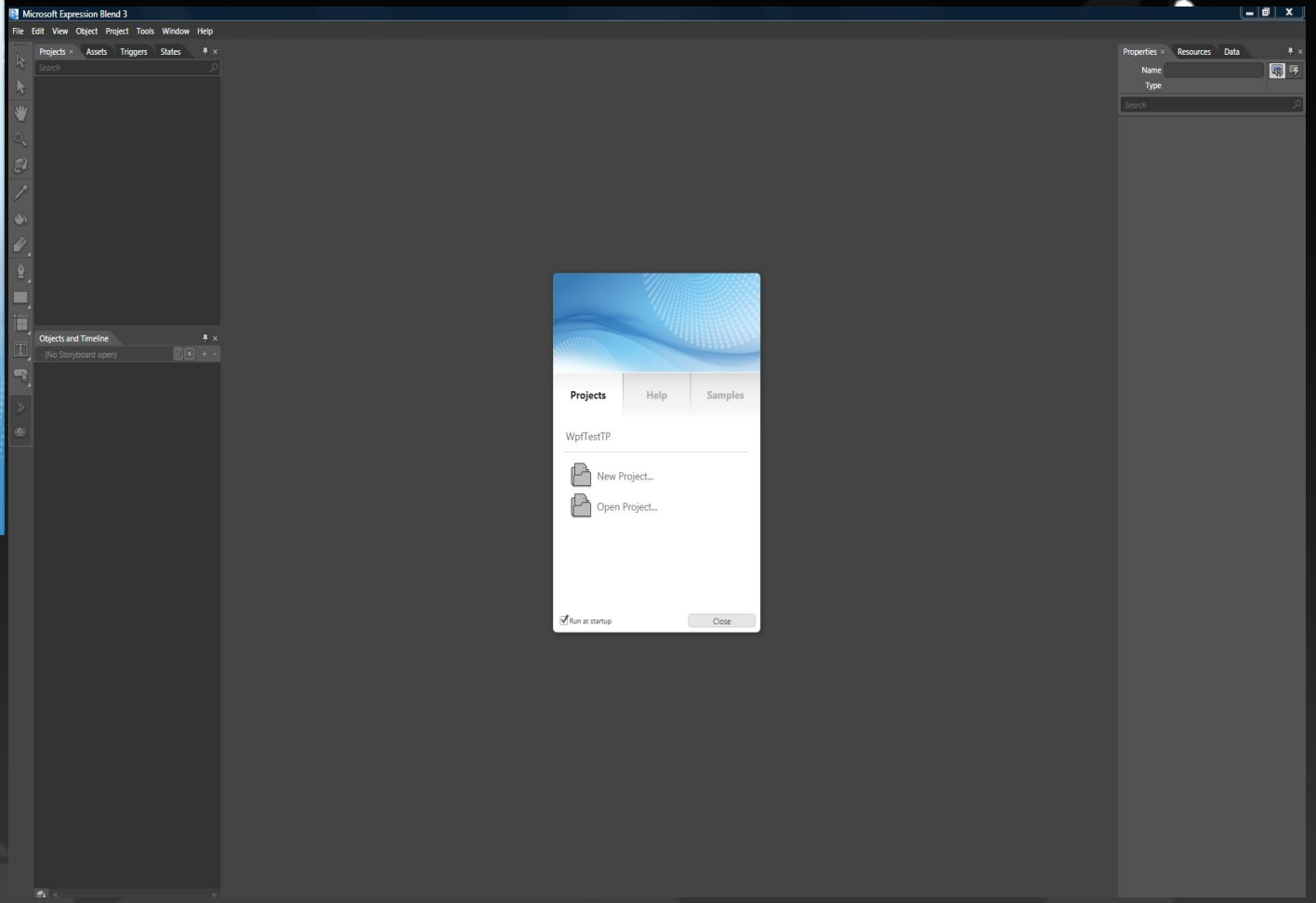
BLEND 3.0

- Toutes les actions que nous avons pu réaliser, sur les créations des interfaces dans les TP précédents peuvent être faîtes avec blend de façon plus simple et rapide.

- Blend crée des solutions (*.sln), autrement dit les solutions peuvent s'ouvrir aussi bien avec visual studio ou blend. L'intérêt est d'ouvrir une solution des 2 côtés pour travailler aussi bien sur la programmation que le design.

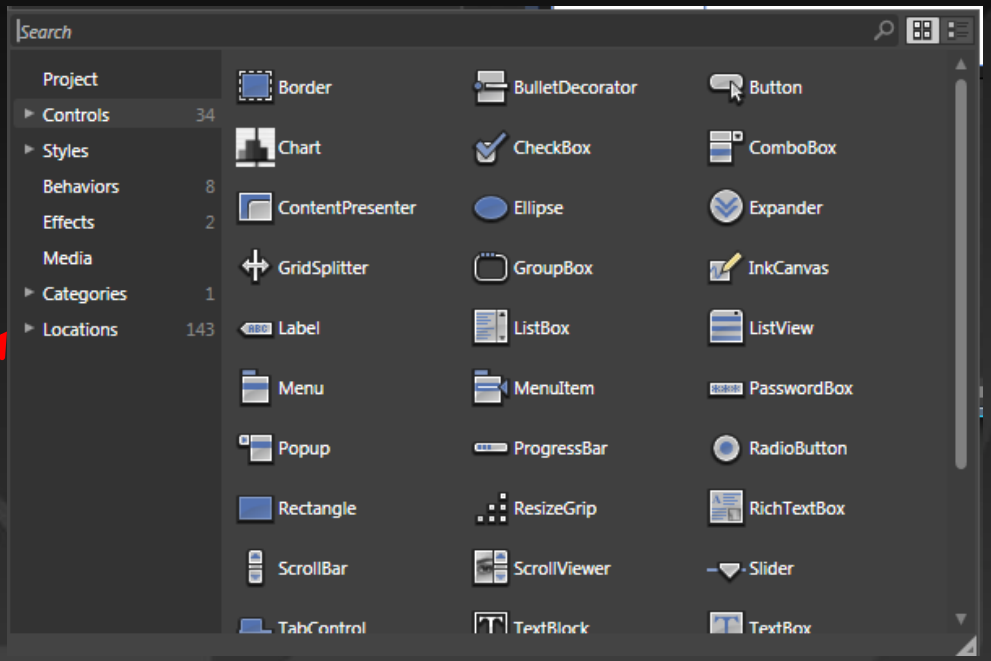
BLEND 3.0


Microsoft
Expression Blend 3
+SketchFlow
© 2009 Microsoft Corporation. All rights reserved.
This program is protected by U.S. and international copyright
laws as described in Help/About.


Microsoft Expression Blend 3
File Edit View Object Project Tools Window Help
Projects Assets Triggers States
Objects and Timeline
(No Storyboard open)
Properties Resources Data
Name
Type
Projects Help Samples
WpfTestTP
New Project...
Open Project...
Run at startup
Close

BLEND 3.0

- La barre d'outil de Blend, rassemble tous les outils pour faire les interfaces.



Permet d'accéder à l'ensemble des composants WPF ainsi qu'aux effets.

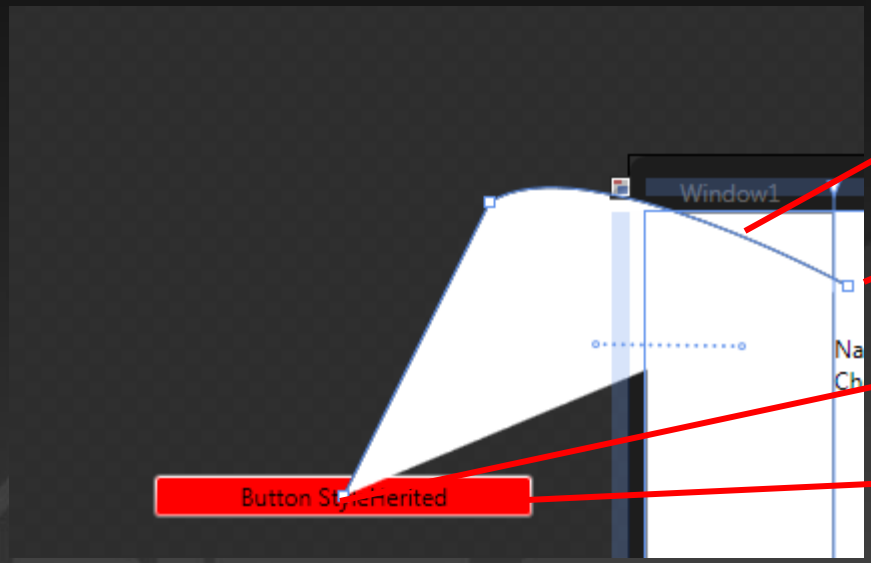
BLEND 3.0

- Pour utiliser blend, nous allons voir dans cet exemple comment réaliser une animation sur une interface.

- Pour cela nous allons reprendre l'exemple d'une interface et se servir des paths pour faire une animation

BLEND 3.0

- Utiliser l'outil « Pen » : permet de dessiner des tracés vectoriels que l'on peut ensuite déformer



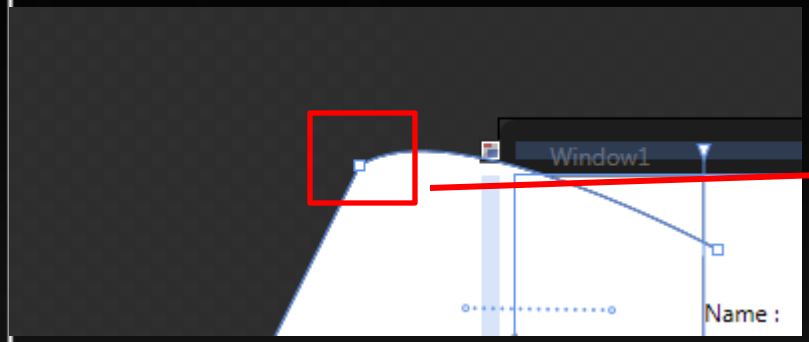
Dessin du path

Point d'arrivée

Point de départ

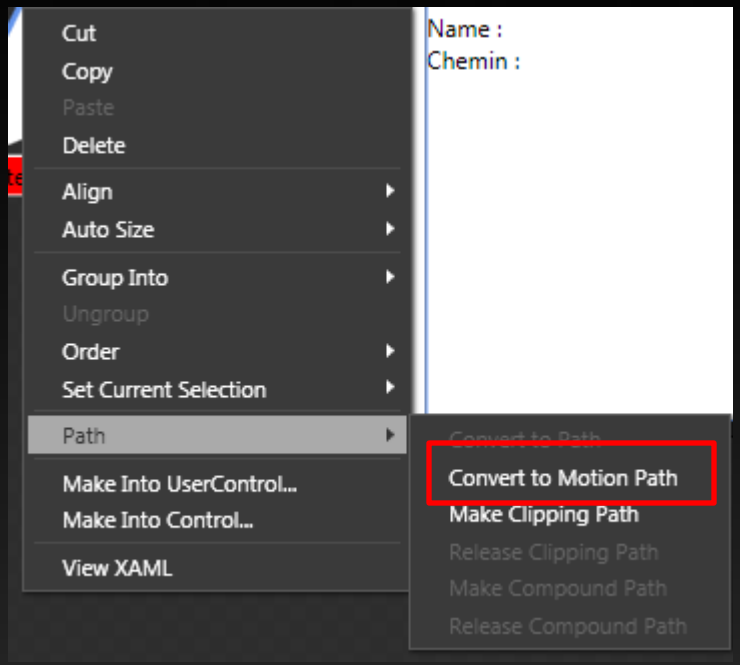
Décalage du bouton

BLEND 3.0

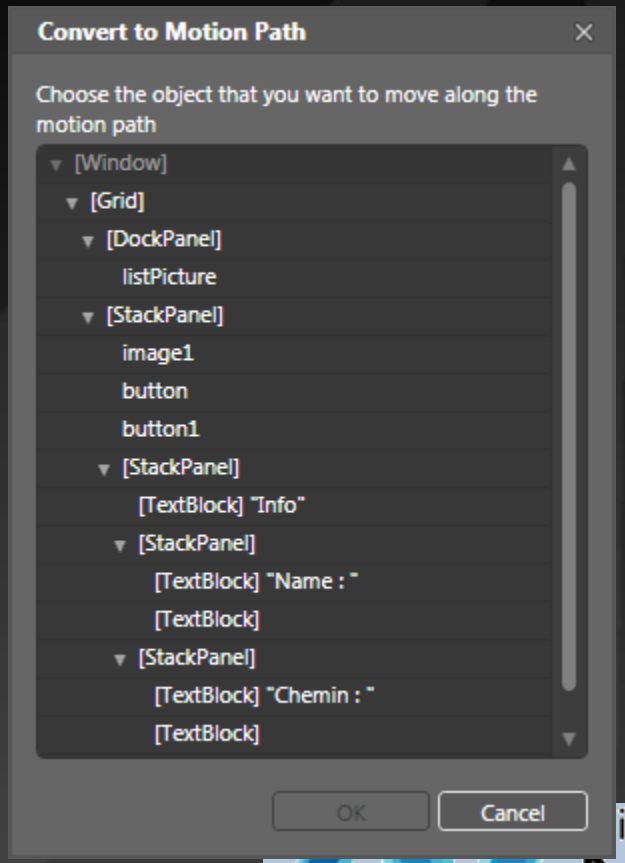


- Point permettant de déformer le chemin.
- Soit en jouant sur les arcs, soit en rajoutant des points

BLEND 3.0



Et de sélectionner l'élément sur lequel appliquer le path



Blend 3.0

- Une fois la dernière opération effectuée, blend crée le storyboard automatiquement. Son lancement sera effectué au démarrage de l'application.

BLEND 3.0

- Une autre utilité de blend est par exemple de pouvoir jouer plus finement sur les animations.

- La méthode consiste à utiliser le premier fichier en changeant légèrement l'animation.

BLEND 3.0

Nous allons repartir de l'animation suivante :

```
<Storyboard x:Key="sbButton1" >
    <DoubleAnimation Duration="00:00:10" From="30" To="200"
   Storyboard.TargetProperty="Height"/>
</Storyboard>
```

Pour la transformer comme ceci :

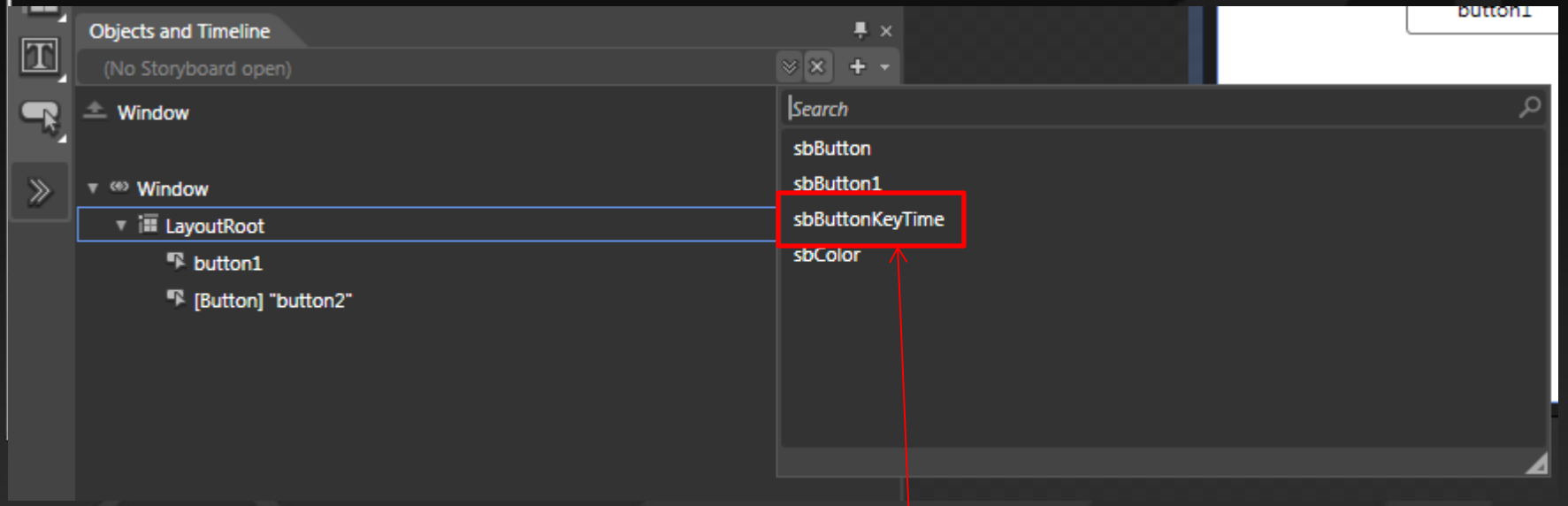
Permet d'utiliser les clès cf début de cours

```
<Storyboard x:Key="sbButtonKeyTime" >
    <DoubleAnimationUsingKeyFrames Duration="00:00:05"
   Storyboard.TargetProperty="Height">
        <SplineDoubleKeyFrame KeyTime="00:00:00" Value="30"/>
        <SplineDoubleKeyFrame KeyTime="00:00:05" Value="200" />
   </DoubleAnimationUsingKeyFrames>
</Storyboard>
```

Permet de fixer les valeurs au cours du temps, progression linéaire de l'animation

BLEND 3.0

- Repasser sur l'éditeur graphique de Blend afin de pouvoir sélectionner dans l'éditeur le storyboard que l'on vient de créer.

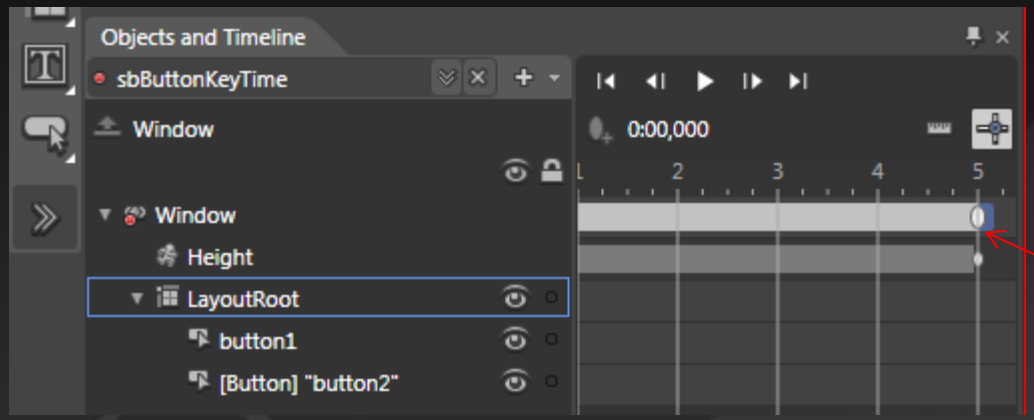


Sélectionner le storyboard

Maintenant Blend ouvre l'éditeur de timeline

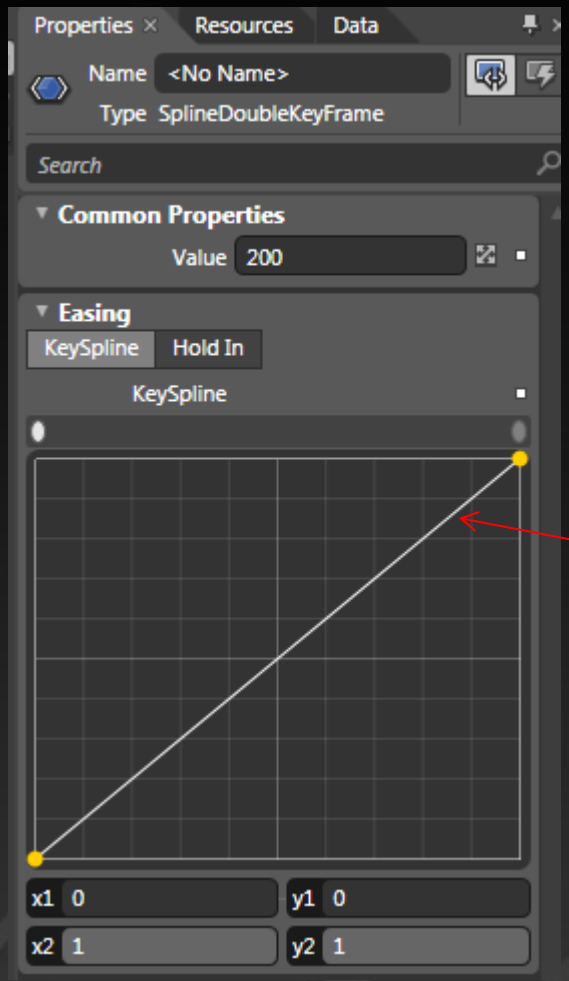
BLEND 3.0

- Déplacer la time line afin de sélectionner la dernière valeur



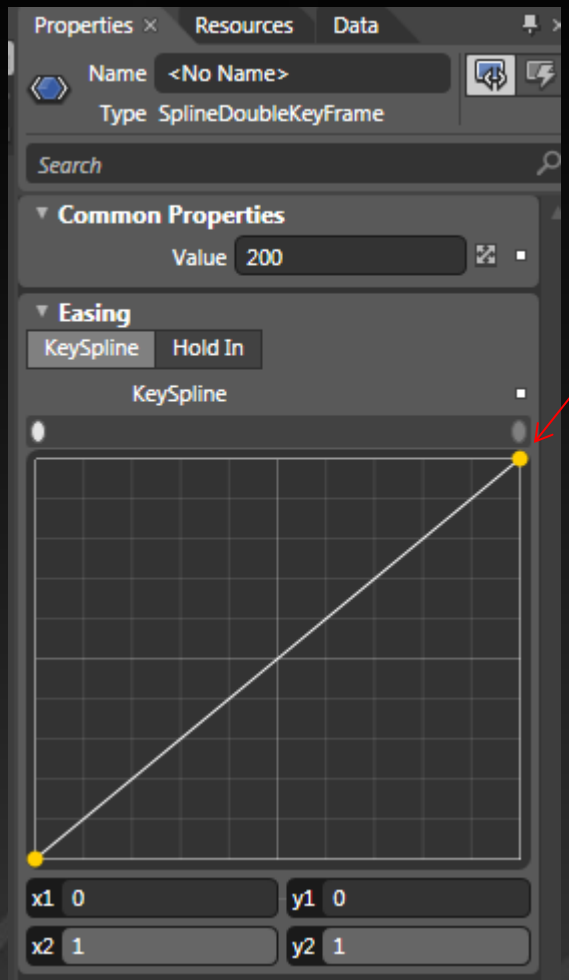
Effectuer un clic gauche sur
la clé afin d'afficher les propriétés

BLEND 3.0



- Ceci représente la vitesse de progression de l'animation. On remarque qu'elle est bien linéaire

BLEND 3.0

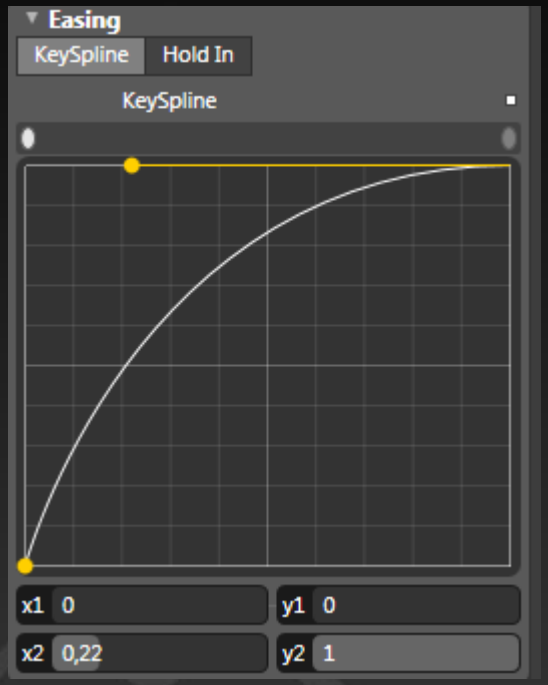


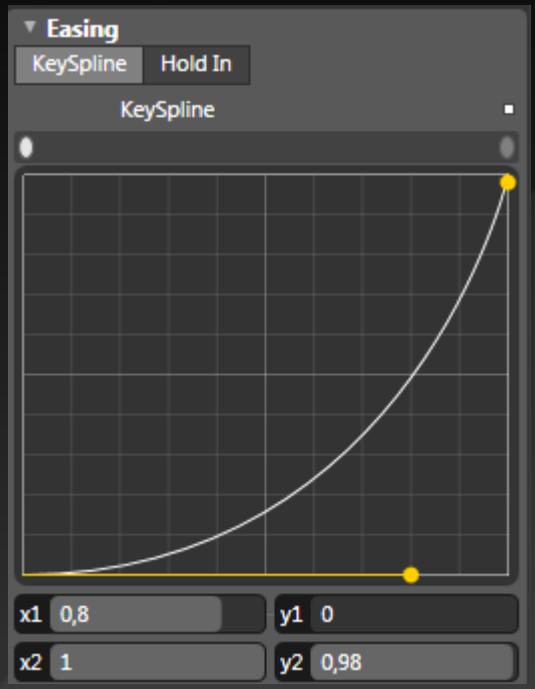
- Pour changer sa vitesse, il faut utiliser les points jaunes.

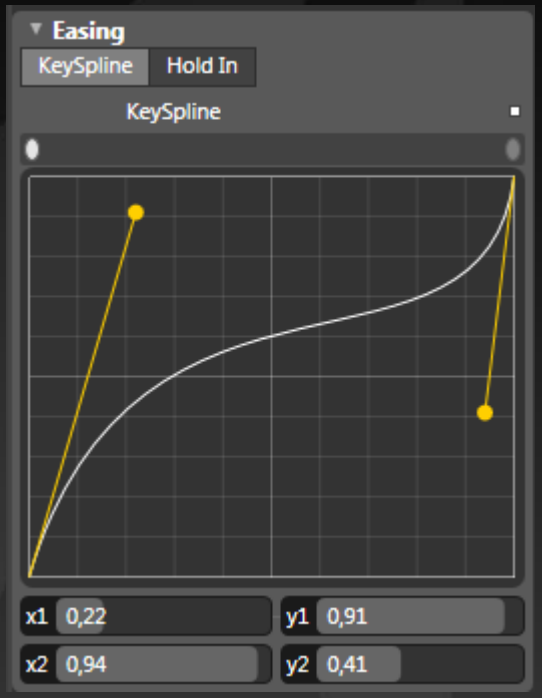
Permet d'obtenir une accélération puis une décélération de l'animation

BLEND 3.0

- Après libre cours à votre imagination







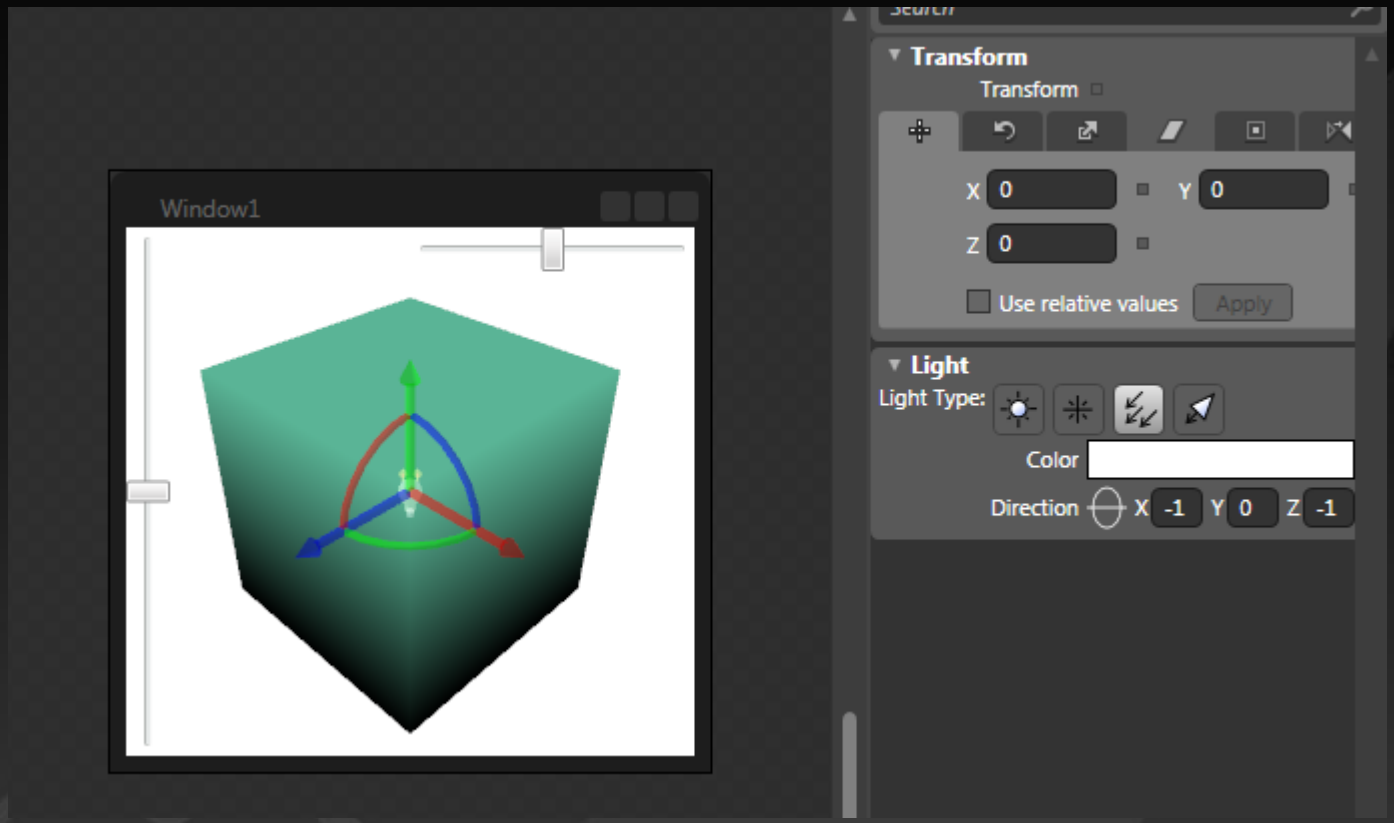
BLEND 3.0

Dernière étape : il ne reste plus qu'à déclencher le bon storyboard sur l'un de vos boutons …

```
<Button Content="button2" Grid.Column="1" Height="30" Width="100" >
    <Button.Triggers>
        <EventTrigger RoutedEvent="Button.MouseEnter">
            <EventTrigger.Actions>
                <BeginStoryboard x:Name="beginSb"
   Storyboard="{StaticResource sbButtonKeyTime}"/>
            </EventTrigger.Actions>
        </EventTrigger>
        <EventTrigger RoutedEvent="Button.MouseLeave">
            <EventTrigger.Actions>
                <StopStoryboard BeginStoryboardName="beginSb"/>
            </EventTrigger.Actions>
        </EventTrigger>
    </Button.Triggers>
</Button>
```

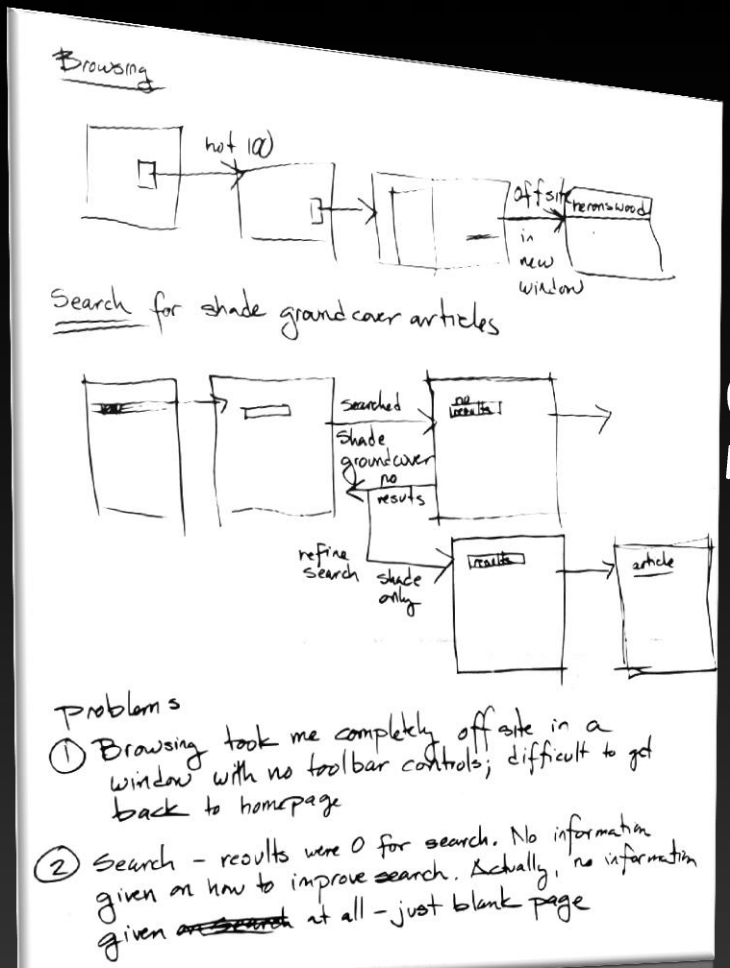

BLEND 3.0

- Et la 3D ??



Blend permet également de gagner du temps dans le développement d'interface 3D

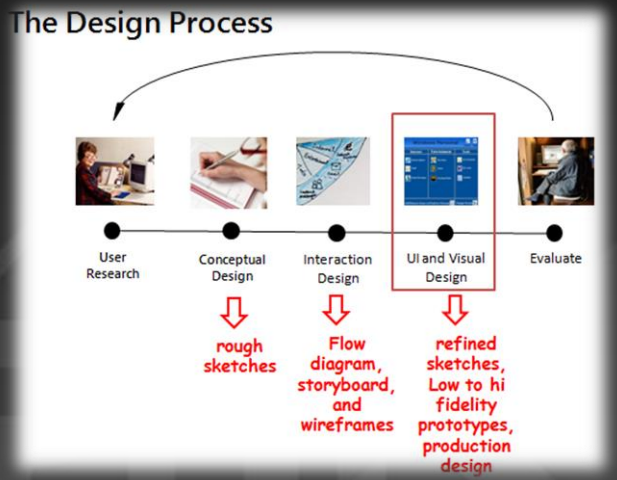
Avec par exemple l'affichage des axes, …



SketchFlow

SketchFlow

- Sketch flow est une nouvelle brique logicielle qui a été intégrée à blend 3.0.
- Skecthflow facilite le travail des designers et évite de passer par l'étape post-it + powerpoint + photoshop.



SketchFlow

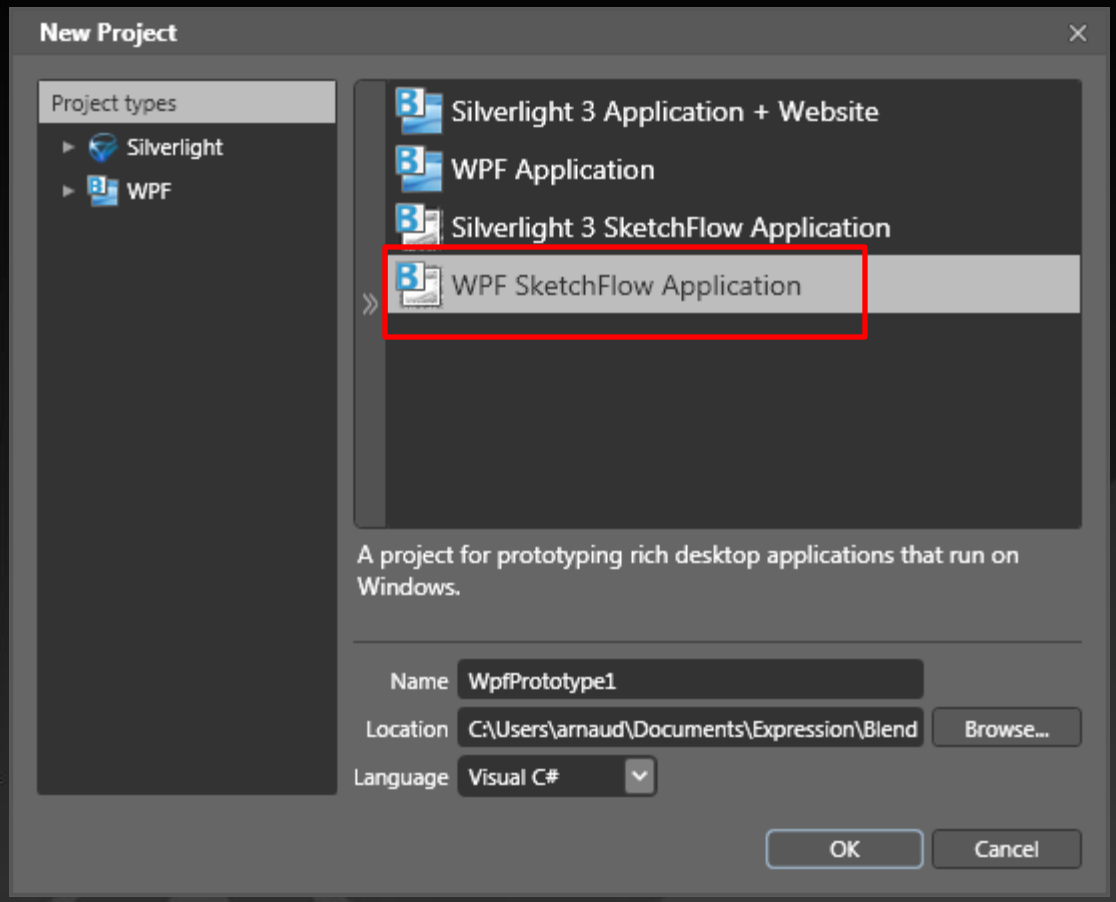
- Ce logiciel permet de réaliser des storyboards de manière simple et efficace.
- Un de ses principaux avantages réside dans le fait qu'il est tourné vers le travail collaboratif.
- Ainsi on peut faire un package de notre storyboard pour l'envoyer aux clients par exemple et se permettre d'y reporter des annotations et des commentaires.

SketchFlow

Animation :

- On peut également réaliser des animations, sans aucun code ainsi que des enchaînements de fenêtres.

SketchFlow



- Créer un nouveau projet sous blend 3.0

SketchFlow

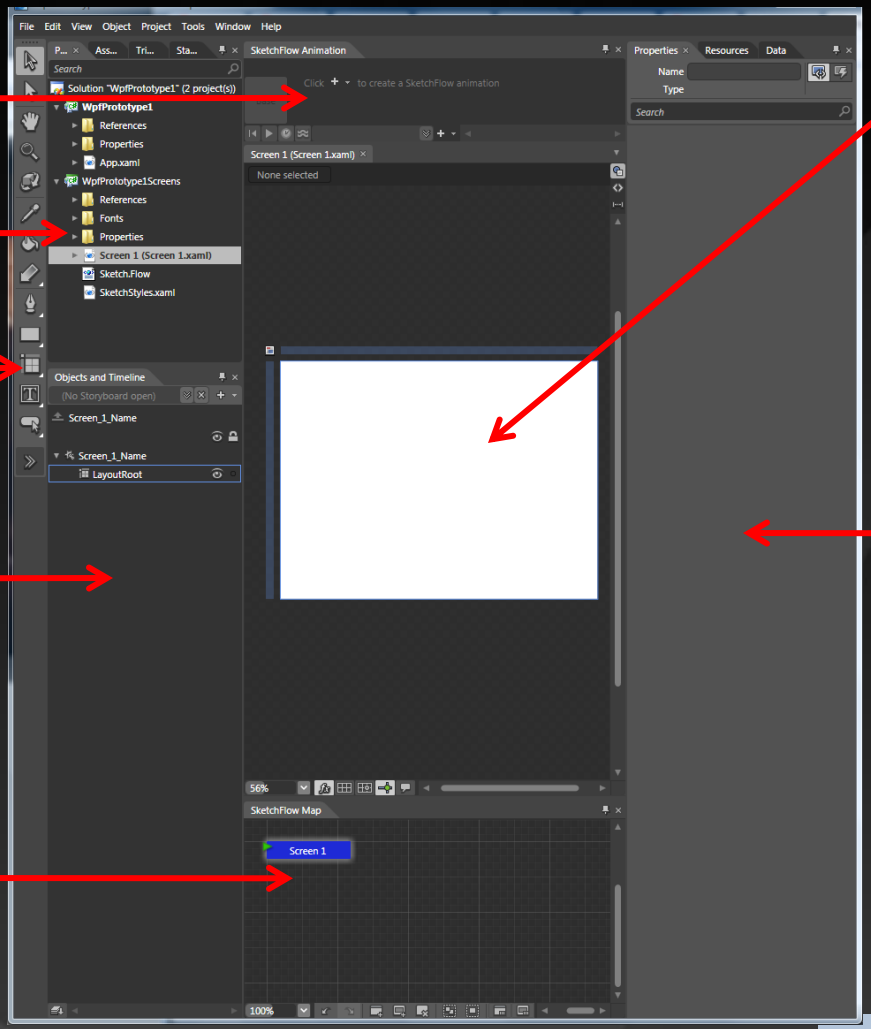
Animation entre les écrans

Liste des projets

La barre d'outils

Liste des objets dans l'écran

Enchaînement des écrans



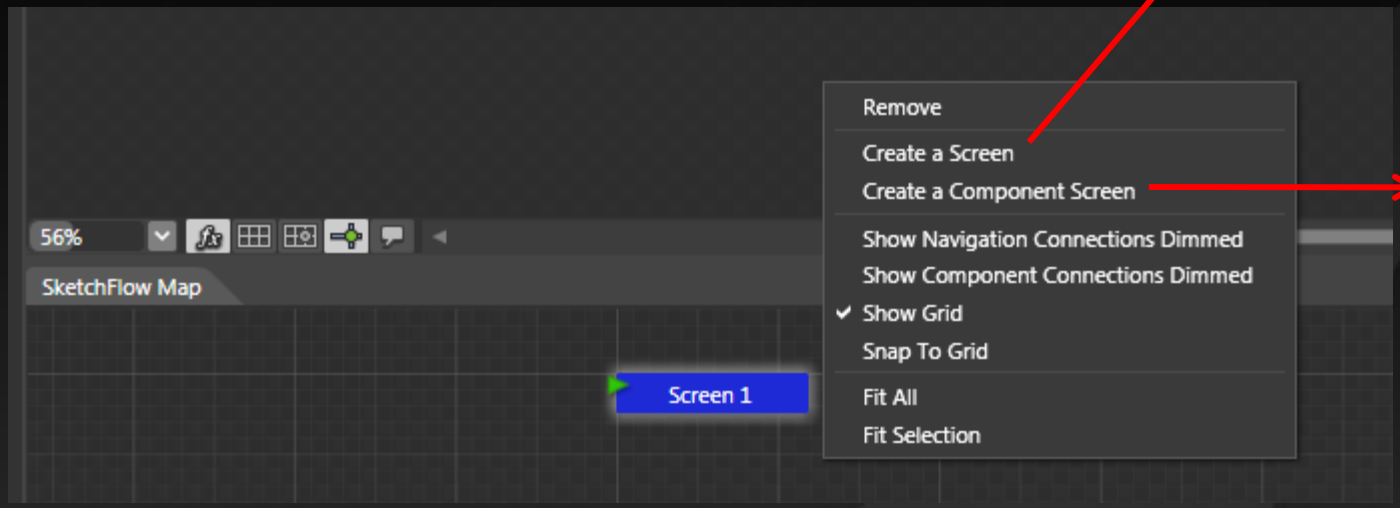
Ecran principal

Les propriétés

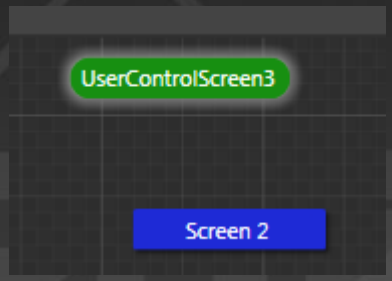
SketchFlow

Ajouter des nouvelles interfaces :

Rajoute un nouvel écran



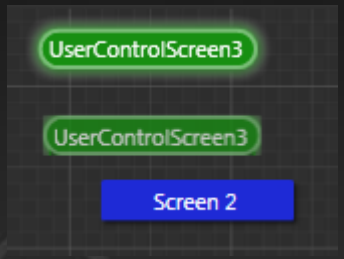
Rajoute un nouveau user control

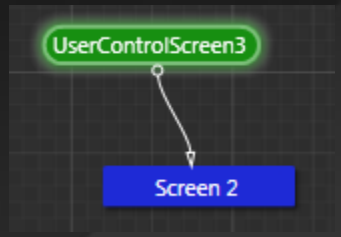


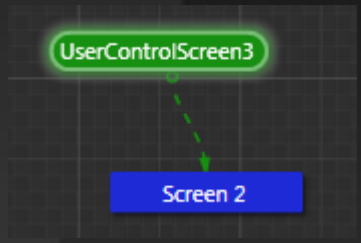
Selon l'insertion les couleurs des éléments ne sont pas les mêmes

SketchFlow

- Dans le user control rajouter un bouton.
- Nous allons l'inclure dans le screen 2.
  - Pour cela, prendre dans le sketchflow Map le usercontrol avec le clic gauche sans relacher et le glisser jusque screen 2



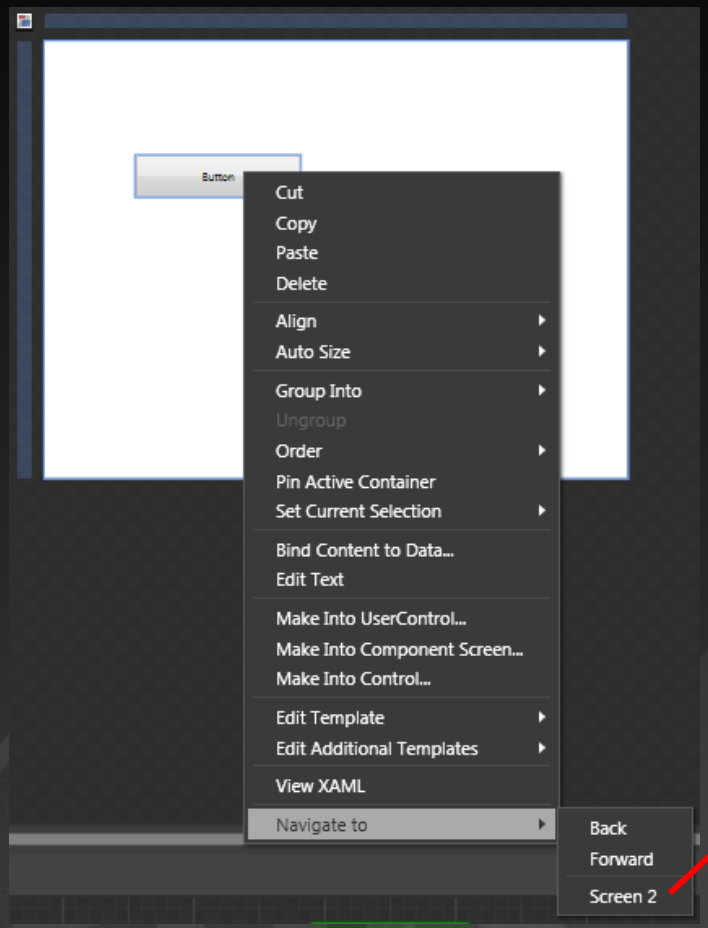




Maintenant dans screen 2 nous avons le user control qui apparait.

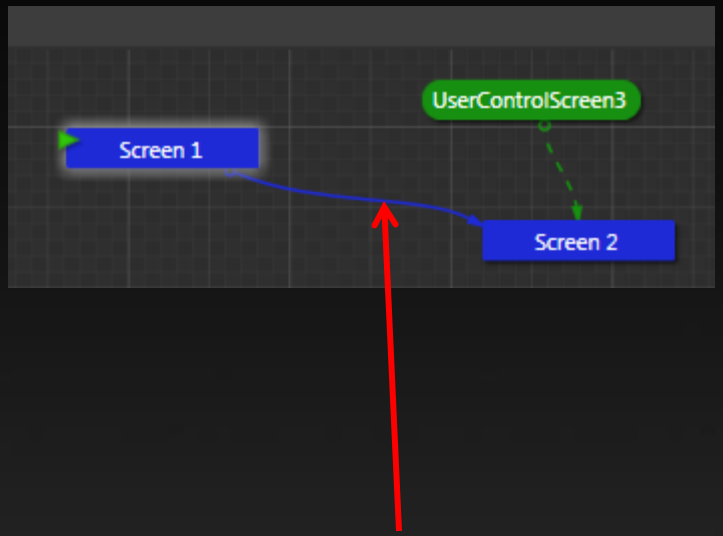
SketchFlow

- Sur le screen1 positionner un bouton. Puis un clic droit dessus et navigate to :



Sélectionner l'écran sur lequel on veut aller

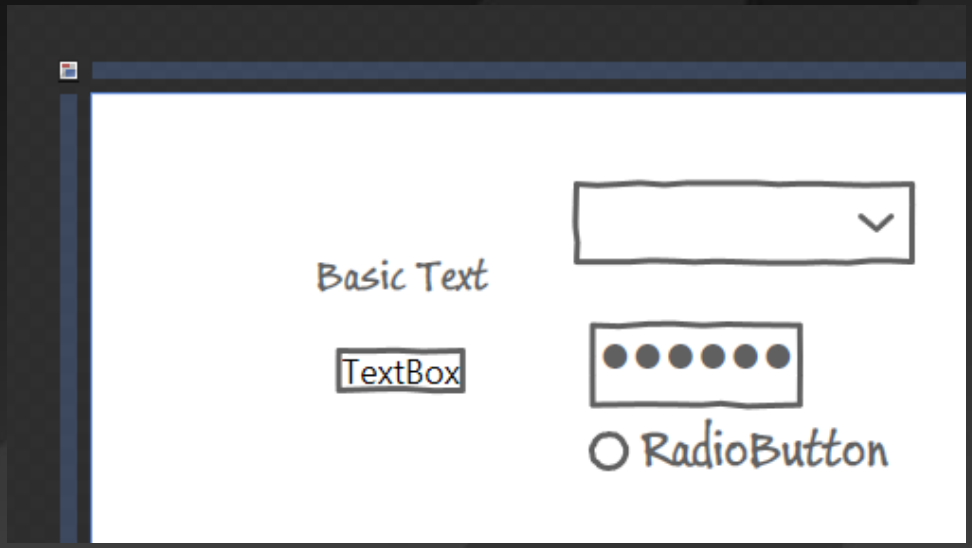
SketchFlow



Lien de navigation

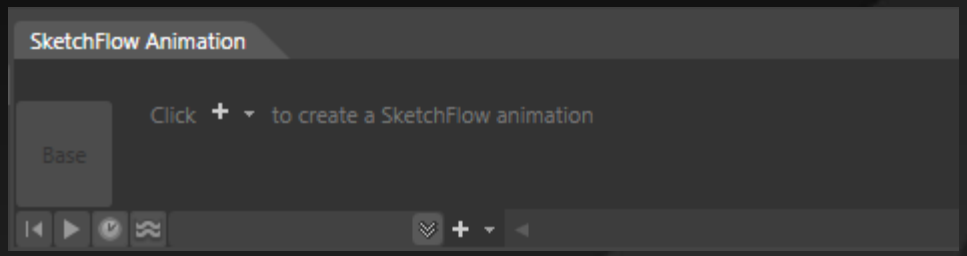
SketchFlow

- On peut avoir également des styles « brouillon »
- => aller dans Asset puis sélectionner :
  - Style
    - SketchStyle



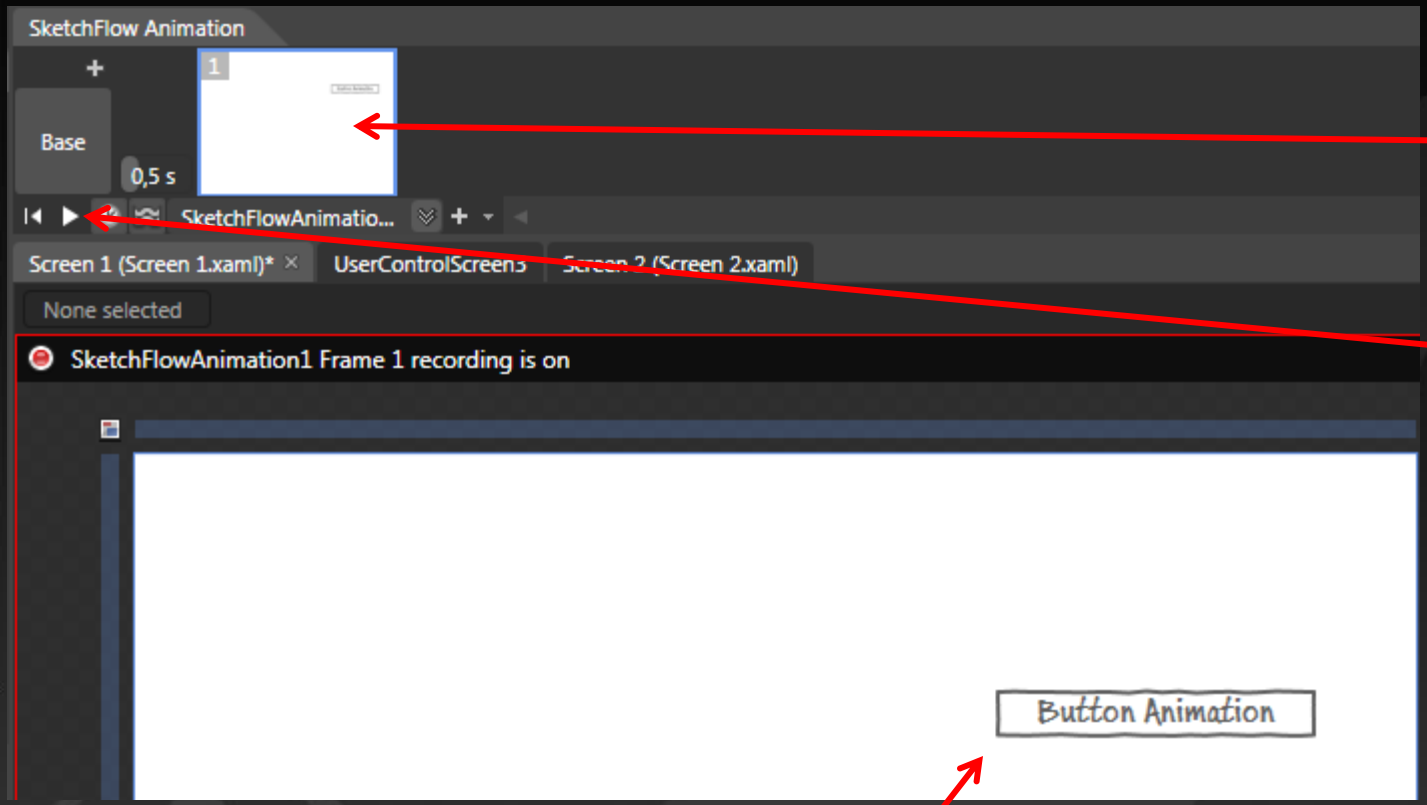
SketchFlow

- Bien que cela soit un prototype on peut quand même utiliser les animations.



SketchFlow

- Pour l'utiliser faire un click sur le « + »



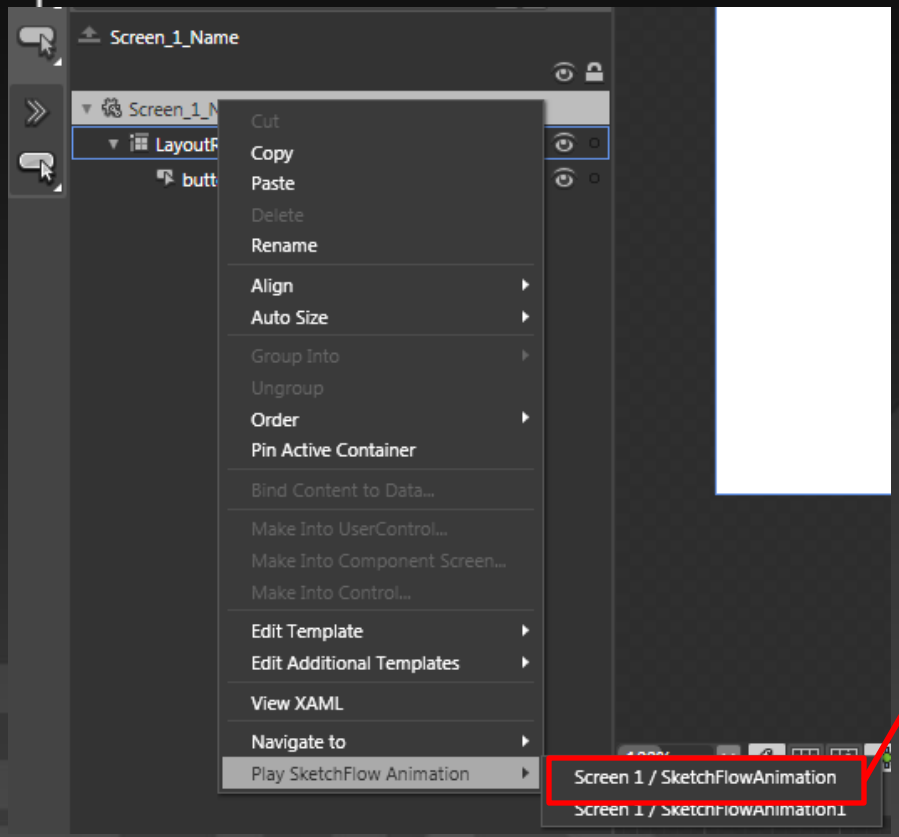
Un nouvel écran apparait

Permet de lire l'animation

Déplacement du bouton pour l'animation

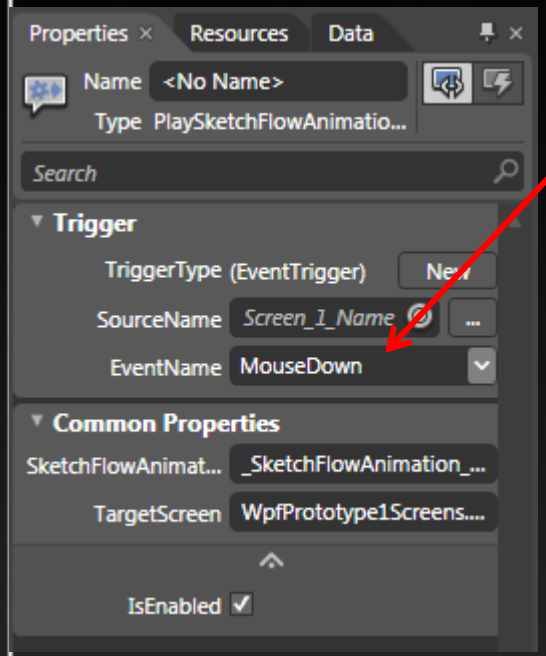
SketchFlow

- Maintenant pour déclencher l'animation :
  - Il faut cliquer sur l'élément de la fenêtre qui doit déclencher l'animation dans le panneau « object & timeline »



Sélection de l'animation
Ici dans notre cas, on va la déclencher au chargement de la fenêtre

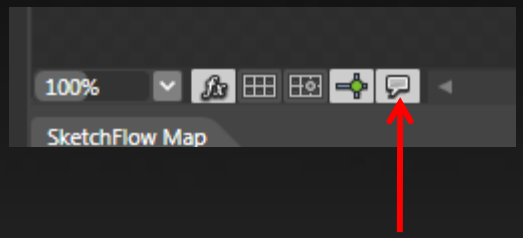
SketchFlow



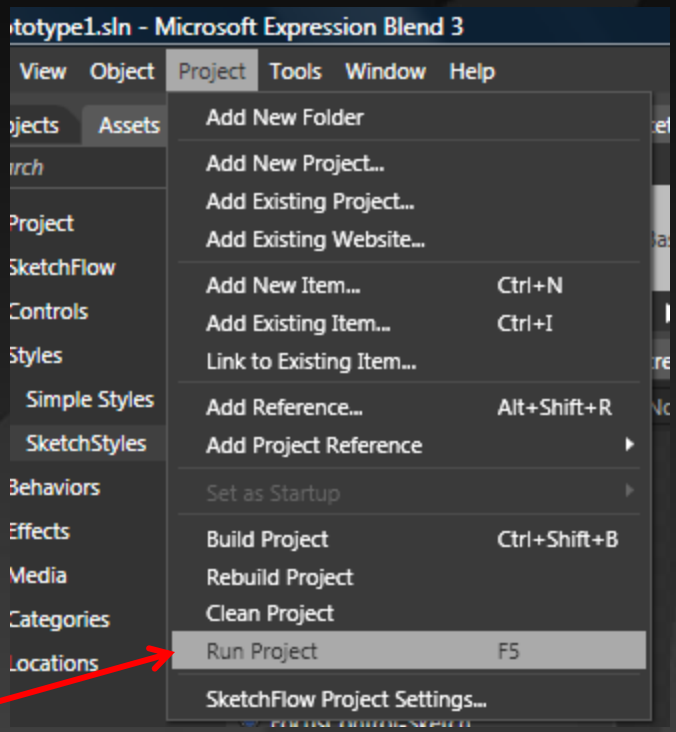
- Changement de la cause de déclenchement

SketchFlow

- Il ne reste plus qu'à exporter notre prototype
  => ne pas oublier d'activer les annotations :



Annotations



Lancement du projet

AVANCÉ

DataConverter

- Permet de transformer un type de données bindé sur une propriété qui ne convient pas au 1er abord.
- Exemple :
  - Un boolean bindé sur la couleur d'un background. Il n'y a aucune correspondance pour cela il faut utiliser un dataConverter.
    - True == Vert
    - False == Rouge
- La classe essentielle est :IValueConverter

DataConverter

```
public class BoolConverter : IValueConverter
{
    #region IValueConverter Members

    public object Convert(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)
    {
        if (value != null)
        {
            bool tmpValue = System.Convert.ToBoolean(value);

            if (tmpValue)
            {
                return Brushes.Green;
            }
            else
            {
                return Brushes.Red;
            }
        }
        return Brushes.Red;
    }

    public object ConvertBack(object value, Type targetType, object parameter, System.Globalization.CultureInfo
     culture)
    {
       //utile lors des bindings twoWay
        throw new NotImplementedException();
    }

    #endregion
}
```

DataConverter

```
<Window x:Class="WpfDataConverter.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="clr-namespace:WpfDataConverter"
    Title="Window1" Height="300" Width="300">
    <Window.Resources>
        <local:BoolConverter x:Key="boolConverter"/>
    </Window.Resources>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <Ellipse x:Name="ellipse1" Width="10" Height="10" Stroke="Black"
   Fill="{Binding Path=BoolFalse, Converter={StaticResource
   boolConverter}}"/>
        <Ellipse Width="10" Height="10" Stroke="Black" Grid.Column="1"
   Fill="{Binding Path=BoolTrue, Converter={StaticResource
   boolConverter}}"/>
     </Grid>
</Window>
```

DataConverter

- Code dans le fichier .cs associé à l'interface

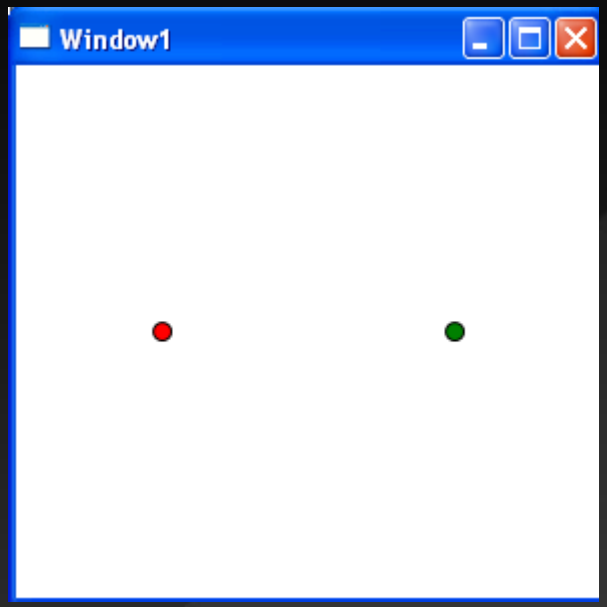
```
private bool _boolFalse ;
private bool _boolTrue ;

public bool BoolTrue...
public bool BoolFalse...

public Window1()
{
    InitializeComponent();
    _boolFalse = false;
    _boolTrue = true;
    DataContext = this;
}
```

DataConverter

- Résultat :



Questions ?