Les animations avec Silverlight

par Benjamin Roux (Retour aux articles)

Date de publication : 05/05/2008

Dernière mise à jour : 17/06/2008

Dans cet article, vous allez découvrir les différents types d'animation proposés dans Silverlight

1 - Introduction

Les animations doivent être placées dans le fichier .xaml. Elles doivent se trouver dans la section **x.Resources** (où x est au choix un Canvas, Grid ou StackPanel), et dans une section **Storyboard**.

```
<Canvas.Resources>
  <Storyboard x:Name="Animate">
    <!-- Animations here -->
  </Storyboard>
</Canvas.Resources>
```

L'attribut *x:Name* permet de donner un nom à votre **Storyboard** pour pouvoir y accéder à partir du javascript ou du code-behind.

On pourra notamment faire

```
Animate.Begin();
```

On pourra également accéder et changer les valeurs des propriétés de nos storyboard via leur nom.

```
Animate.Duration = new Duration(TimeSpan.FromSeconds(5));
```

On peut également depuis la version 2 de Silverlight rajouter des animations et des storyboard dynamiquement en code.

```
DoubleAnimation SimpleAnimation = new DoubleAnimation();
[...]
Animate.Children.Add(SimpleAnimation);
```

```
StoryBoard storyBoard = new StoryBoard();
[...]
MonCanvas.Resources.Add(storyBoard);
```

Un **Storyboard** peut bien entendu contenir plusieurs animations.

Il est également possible de lancer une animation à l'évènement **Loaded**, pour cela on peut placer le **Storyboard** dans les balises du contrôle (ici un **Rectangle**) de cette façon.

```
<Rectangle>
    <Rectangle.Triggers>
        <EventTrigger>
            <BeginStoryboard>
                <Storyboard>
                    <!-- Animations here -->
                </Storyboard>
            </BeginStoryboard>
        </EventTrigger>
    </Rectangle.Triggers>
</Rectangle>
```

Ici l'animation sera lancée après le chargement de l'objet **Rectangle**.

On peut aussi imbriquer les **Storyboard**, vous en verrez l'utilité en regardant les exemples présentés plus bas.

2 - From - To Animations

L'animation dite *From - To* est l'animation de base, elle modifie la propriété d'un objet de la valeur **From** à la valeur **To**.

Le type de la propriété pouvant changer, nous avons à notre disposition plusieurs type d'animation *From - To*.

- DoubleAnimation
- ColorAnimation
- PointAnimation

La **DoubleAnimation** permet d'animer une propriété de type double (Height, Width, Canvas.Top, Canvas.Left...).

La **ColorAnimation** permet d'animer une propriété de type couleur (Backgroud, Foreground...).

La **PointAnimation** permet d'animer une propriété de type Point (RenderTransformOrigin...).

Commençons par spécifier la durée de l'animation, pour cela utilisons la propriété **Duration** qui a la syntaxe suivante : hh:mm:ss (où hh désigne le nombre d'heures, mm de minutes et ss de secondes).

```
<Canvas.Resources>
    <Storyboard x:Name="FromToAnimation">
        <DoubleAnimation Duration="00:00:05" />
    </Storyboard>
</Canvas.Resources>
```

Ici mon animation durera donc 5 secondes.

Pour spécifier des millisecondes (ici 50) : 00:00:00.050 !

Ensuite spécifions les valeurs de début et de fin de l'animation, il faut comme vous vous en doutez, utiliser les propriétés **From** et **To**.

```
<Canvas.Resources>
    <Storyboard x:Name="FromToAnimation">
        <DoubleAnimation Duration="00:00:05" From="10" To="100" />
    </Storyboard>
</Canvas.Resources>
```

Ici la propriété (que nous n'avons pas définie), variera de 10 à 100 en 5 secondes.

Une autre propritété peut être spécifiée en rapport avec **From** et **To** : **By**.

Cette propriété sert à animer la valeur **par** (**by**) une certaine valeur, il ne s'agit en aucun cas de définir par quoi va passer l'animation.

Voici un petit tableau expliquant le fonctionnement des propriétés **From**, **To** et **By**. La valeur initiale est la valeur spécifiée à la création de l'objet qu'on veut animer.

Propriétés spécifiées	Résultat
From	La propriété changera en partant de la valeur **From** jusqu'à la valeur initiale
From et **To**	La propriété changera en partant de la valeur **From** jusqu'à la valeur **To**
To	La propriété changera en partant de la valeur initiale jusqu'à la valeur **To**
By	La propriété changera en partant de la valeur initiale jusqu'à la valeur **By** + la valeur initiale
From et **By**	La propriété changera en partant de la valeur **From** jusqu'à la valeur **From** + **By**

*Si les propriétés **To** et **By** sont spécifiées en même temps, la valeur **To** est prioritaire.*

Exemples :

DoubleAnimation
```
<Canvas.Resources>
    <Storyboard x:Name="FromToAnimation">
        <DoubleAnimation Duration="00:00:05" From="10" To="100" />
    </Storyboard>
</Canvas.Resources>
```

ColorAnimation
```
<Canvas.Resources>
    <Storyboard x:Name="FromToAnimation">
        <ColorAnimation Duration="00:00:05" From="Red" By="Blue" />
    </Storyboard>
</Canvas.Resources>
```

PointAnimation
```
<Canvas.Resources>
    <Storyboard x:Name="FromToAnimation">
        <PointAnimation Duration="00:00:05" To="1,1"/>
    </Storyboard>
</Canvas.Resources>
```

Alors bien évidemment ici, nous ne spécifions nulle part quelle propriété animer, et surtout quel objet animer. Pour cela nous devons utiliser les propriétés **Storyboard.TargetName** et **Storyboard.TargetProperty**.

```
<Canvas.Resources>
    <Storyboard x:Name="FromToAnimation">
        <DoubleAnimation Storyboard.TargetName="GreenSquare" Storyboard.TargetProperty="(Canvas.Left)"
                         Duration="00:00:05" From="10" To="100" />
    </Storyboard>
</Canvas.Resources>
```

Ici nous avons spéficié d'animer la propriété **Canvas.Left** de l'objet **GreenSquare**.
GreenSquare est simplement un carré défini dans le XAML.

GreenSquare
```
<Rectangle x:Name="GreenSquare" Width="100" Height="100" Fill="Green" />
```

Selon la propriété à animer la syntaxe de **Storyboard.TargetProperty** diffère :

http://msdn2.microsoft.com/en-us/library/system.windows.media.animation.storyboard.targetproperty(VS.95).aspx

Voici donc quelques exemples d'utilisation de **Storyboard.TargetProperty**.

Canvas.Left
```
<Canvas.Resources>
    <Storyboard x:Name="FromToAnimation">
        <DoubleAnimation Storyboard.TargetName="GreenSquare" Storyboard.TargetProperty="(Canvas.Left)"
                         Duration="00:00:05" From="10" To="100" />
    </Storyboard>
</Canvas.Resources>
```

Fill.Color
```
<Canvas.Resources>
    <Storyboard x:Name="FromToAnimation">
        <ColorAnimation Storyboard.TargetName="GreenSquare" Storyboard.TargetProperty="(Fill).(Color)"
                        Duration="00:00:05" From="Black" To="Green" />
    </Storyboard>
</Canvas.Resources>
```

Nous allons maintenant passer en revue les autres propriétés d'une animation *From - To*.

Propriété	Valeurs possibles	Résultat
AutoReverse	True / False	**True** : après avoir terminé, l'animation fait marche arrière **False** par défaut
BeginTime	hh:mm:ss	Une fois la méthode Begin appellée, la **TimeLine** commencera après le temps spécifié.
FillBehavior	HoldEnd / Stop	Spécifie comment se comporte la propriété de l'objet une fois l'animation terminée. **HoldEnd** (*défaut*) : la propriété garde sa valeur de fin de l'animation. **Stop** : la propriété retourne à sa valeur initiale.
RepeatBehavior	nx hh:mm:ss Forever	**nx** : l'animation sera repétée *n* fois (3x par exemple). **hh:mm:ss** : l'animation sera repétée le temps spécifié, si le temps spécifié est inférieur à la propriété **Duration**, l'animation s'arrêtera au temps spécifié. **Forever** : l'animation sera répétée à l'infini.
SpeedRatio	n	Spécifie le ratio d'accélération de votre animation. Par exemple, si je spécifie 2, mon animation ira 2 fois plus vite.

Certaines propriétés peuvent être définies au niveau du **Storyboard**, comme **Duration** ou **Storyboard.TargetName**. Du coup ces propriétés sont valables pour toutes les animations définies dans le **Storyboard** (si elles ne sont pas redéfinies bien entendu).

Je rappelle également que l'on peut accéder à toutes ces propriétés via le code behind, il suffit d'ajouter l'attribut x:Name avec une valeur à nos animations.

```
<Canvas.Resources>
    <Storyboard x:Name="FromToAnimation">
        <DoubleAnimation x:Name="SimpleAnimation"/>
    </Storyboard>
</Canvas.Resources>
```

Et un code C# possible (Silverlight 2).

```
SimpleAnimation.Duration = new Duration(TimeSpan.FromSeconds(5));
SimpleAnimation.From = 10;
SimpleAnimation.To = 540;
SimpleAnimation.SetValue(Storyboard.TargetNameProperty, "GreenSquare");
SimpleAnimation.SetValue(Storyboard.TargetPropertyProperty, "(Canvas.Left)");
```

3 - Key Frame Animations

3-1 - Introduction

Contrairement aux animations de type *From - To*, dans lesquelles la valeur variait de **From** à **To**, les *Key Frame* animations nous donnent un meilleur contrôle sur l'animation, en effet nous pouvons spécifier la valeur de la propriété à un moment *T*.

Pour ces animations nous devons utiliser la balise <XAnimationUsingKeyFrames> où **X** peut être le mot **Double**, **Point** ou **Color**.

Nous avons donc, selon le type de la propriété :

- DoubleAnimationUsingKeyFrames
- ColorAnimationUsingKeyFrames
- PointAnimationUsingKeyFrames

Il existe 3 types d'animations de type *Key Frame* :

- Linear
- Discrete
- Spline

Il faut savoir que l'on peut utiliser en même temps des **Linear**, **Discrete** et **Spline** animation dans la même **XAnimationUsingKeyFrames**.

3-2 - Linear

La *Linear* animation est celle qui se rapproche le plus de la *From - To*. Pour l'utiliser nous devons ajouter des balises de type <LinearXKeyFrame> où **X** est encore soit **Double**, **Point** ou **Color** selon la propriété (bien entendu si vous avez utilisé une **DoubleAnimationUsingKeyFrames**, il vous faut utiliser des **LinearDoubleKeyFrame**).

Voici l'utilisation de base d'une **LinearXKeyFrame**.

```
<Canvas.Resources>
    <Storyboard x:Name="LinearAnim">

  <DoubleAnimationUsingKeyFrames Storyboard.TargetName="GreenSquare" Storyboard.TargetProperty="(Canvas.Left)">
            <LinearDoubleKeyFrame Value="10" KeyTime="00:00:00"/>
            <LinearDoubleKeyFrame Value="540" KeyTime="00:00:04"/>
            <LinearDoubleKeyFrame Value="500" KeyTime="00:00:05"/>
        </DoubleAnimationUsingKeyFrames>
    </Storyboard>
</Canvas.Resources>
```

L'utilisation est simplissime, nous définissions les valeurs de nos propriétés à un moment T.

Ici nous voyons donc qu'au début de notre animation notre propriété sera à 10, 4 secondes plus tard elle sera à 540, 1 seconde plus tard, elle sera à 500.
Autrement dit notre animation durera 5 secondes, et durant ces 5 secondes la propriété **Canvas.Left** de notre **GreenSquare** (l'animation provoquera un déplacement) passera à 540 en 4 secondes (déplacement plutôt rapide), puis passera à 500 une seconde plus tard (déplacement lent).

L'utilisation est la même avec une **Color** ou une **Point** animation, il suffit juste de mettre une valeur correcte dans la propriété **Value** (Black, Red, Pink... pour une **Color**, 0.5,1 par exemple pour une **Point**).

Vous pouvez trouver une démo **ici**.

3-3 - Discrete

L'utilisation d'une **Discrete** animation est la même que pour une **Linear** animation, la seule différence se fait sur le résultat visuel.

En effet nous ne verrons pas les différentes valeurs par laquelle passe le propriété, lorsque le temps spécifié dans la propriété **KeyTime** est arrivé, le changement se fait d'un seul coup.

Reprenons l'exemple de la **Linear** animation (on remplace simplement **LinearDoubleKeyFrame** par **DiscreteDoubleKeyFrame**).

```
<Canvas.Resources>
    <Storyboard x:Name="DiscreteAnim">

  <DoubleAnimationUsingKeyFrames Storyboard.TargetName="GreenSquare" Storyboard.TargetProperty="(Canvas.Left)">
            <DiscreteDoubleKeyFrame Value="10" KeyTime="00:00:00"/>
            <DiscreteDoubleKeyFrame Value="540" KeyTime="00:00:04"/>
            <DiscreteDoubleKeyFrame Value="500" KeyTime="00:00:05"/>
        </DoubleAnimationUsingKeyFrames>
    </Storyboard>
</Canvas.Resources>
```

Dans ce type d'animation, la valeur ne change pas d'une frame à l'autre.
Autrement dit, de 0 à 4 secondes (à partir du début de l'animation) on ne verra aucun mouvement de la part de notre **GreenSquare**, c'est seulement à 4 secondes que sa propriété **Canvas.Left** passera à 540. Une seconde plus tard, sa propriété passera à 500.

Vous pouvez trouver une démo **ici**.

Ce type d'animation fait apparaitre un autre type d'objet à animer, il s'agit des objets de type **object**.

Exemple avec un **LinearGradientBrush** :

```
<Canvas.Resources>
    <Storyboard x:Name="DiscreteAnim">

  <ObjectAnimationUsingKeyFrames Storyboard.TargetName="GreenSquare" Storyboard.TargetProperty="Fill">
            <DiscreteObjectKeyFrame KeyTime="00:00:00">
                <DiscreteObjectKeyFrame.Value>
                    <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                        <GradientStop Color="#FFC8DBEE" Offset="0.260"/>
                        <GradientStop Color="#FF84AFE6" Offset="0.530"/>
                        <GradientStop Color="#FFC8DBEE" Offset="0.80"/>
                    </LinearGradientBrush>
                </DiscreteObjectKeyFrame.Value>
            </DiscreteObjectKeyFrame>
        </ObjectAnimationUsingKeyFrames>
    </Storyboard>
</Canvas.Resources>
```

Vous pouvez trouver une démo **ici**.

*Ce type d'animation n'est disponible qu'en mode **Discrete**.*

3-4 - Spline

La **Spline** animation est la plus difficile à comprendre. En effet, elle est basée sur une **courbe de Bézier**.

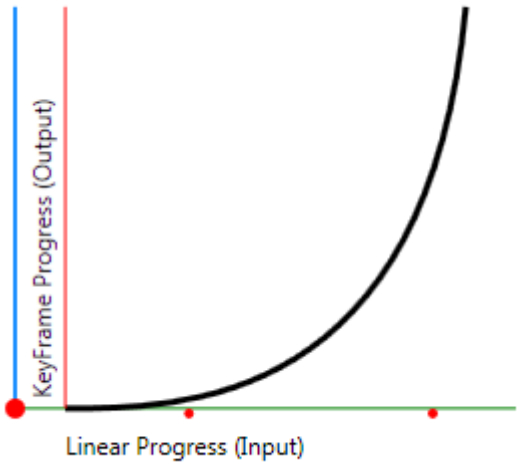
Comme toujours nous avons le choix entre **SplineDoubleKeyFrame**, **SplineColorKeyFrame** ou **SplinePointKeyFrame**.

La propriété importante ici est **KeySpline** qui définit les 2 points de contrôle de notre *courbe de Bézier*. Cette fameuse courbe contrôle le taux de changement de la vitesse de l'animation.

Voici un exemple, avec une explication :

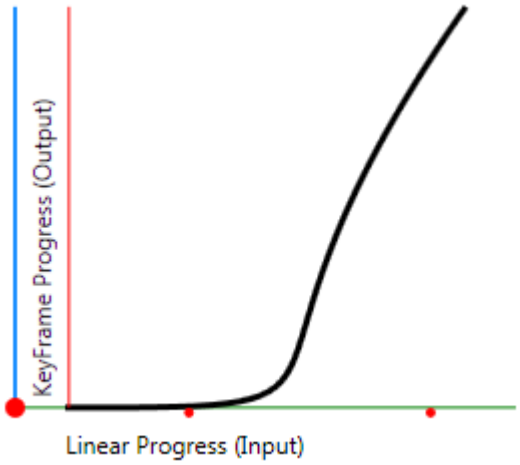
```
<Canvas.Resources>
    <Storyboard x:Name="SplineAnim">

  <DoubleAnimationUsingKeyFrames Storyboard.TargetName="GreenSquare" Storyboard.TargetProperty="(Canvas.Left)">
            <SplineDoubleKeyFrame KeyTime="00:00:00" Value="10"/>
            <SplineDoubleKeyFrame KeyTime="00:00:02" KeySpline="0.3,0 0.9,0" Value="500"/>
            <SplineDoubleKeyFrame KeyTime="00:00:04" KeySpline="0.9,0 0.3,0" Value="10"/>
        </DoubleAnimationUsingKeyFrames>
    </Storyboard>
</Canvas.Resources>
```

Tout d'abord, voici la représentation d'une *courbe de Bézier* ayant pour points de contrôle (0,3;0) et (0,9;0).



Courbe 1

Et celle ayant pour points de contrôle (0,9;0) et (0,3;0).



Courbe 2

Explication :
Pendant 2 secondes, la vitesse de l'animation suivra la courbe numéro 1. Autrement dit la propriété **Canvas.Left** du **GreenSquare**, passera de 10 à 500 tout en suivant la courbe pour définir la vitesse (axe X = temps, axe Y = valeur).

Ce type d'animation nous permet de contrôler l'accélération et la décélération de nos animations pour donner une impression plus réaliste.

Essayez donc vous verrez que cette accélération est un peu plus jolie et réaliste qu'en utilisant des **Linear** animations.

Vous pouvez trouver une démo **ici**.

4 - Exemples

Voici quelques exemples d'animations.

4-1 - Agrandissement d'un contrôle

Agrandissement contrôle
```
<Canvas.Resources>
    <Storyboard x:Name="ScaleAnimation" Storyboard.TargetName="GreenSquare" Duration="00:00:02">
        <DoubleAnimation Storyboard.TargetProperty="(RenderTransform).(ScaleX)" From="0" To="1"/>
        <DoubleAnimation Storyboard.TargetProperty="(RenderTransform).(ScaleY)" From="0" To="1"/>
    </Storyboard>
</Canvas.Resources>

<Rectangle x:Name="GreenSquare" Width="300" Height="200" Canvas.Top="190" Canvas.Left="10" Fill="Green">
    <Rectangle.RenderTransform>
        <ScaleTransform/>
    </Rectangle.RenderTransform>
</Rectangle>
```

Ici nous utilisons donc 2 animations qui se lancent en même temps, et qui durent la même durée.

La première anime la propriété **ScaleX** de notre **GreenSquare** de 0 à 1.
La seconde anime la propriété **ScaleY** elle aussi de 0 à 1.

Résultat visible **ici**.

On peut par exemple s'en servir pour faire apparaître un menu.

4-2 - Rotation d'une image

Voici un petit code pour faire tourner une image dans le sens de aiguilles d'une montre, à l'infini.

Rotation image
```
<Canvas.Resources>
    <Storyboard x:Name="RotateAnimation">
        <DoubleAnimation Storyboard.TargetName="logoDVP" Storyboard.TargetProperty="(RenderTransform).
(Angle)"
            From="0" To="360" Duration="00:00:02" RepeatBehavior="Forever"/>
    </Storyboard>
</Canvas.Resources>

<Image x:Name="logoDVP" Source="logo_DVP.png" Canvas.Left="50" Canvas.Top="200">
    <Image.RenderTransform>
        <RotateTransform CenterX="188" CenterY="37"/>
        <!-- CenterX and CenterY for center the rotation (image does 376*74px) -->
    </Image.RenderTransform>
</Image>
```

Résultat visible **ici**.

4-3 - Animation des items d'un menu

Voici un code pour animer les items d'un menu

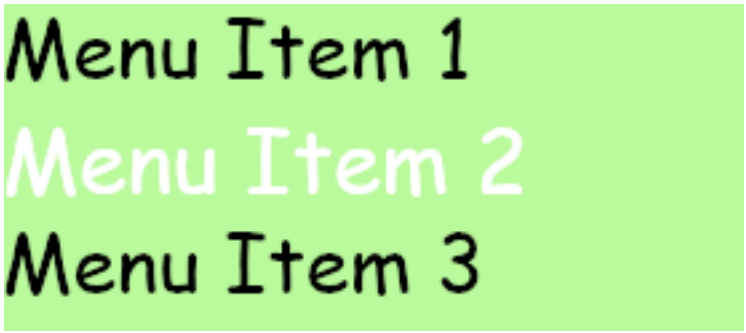
```
<Canvas.Resources>
    <Storyboard x:Name="ItemAnimation" RepeatBehavior="Forever" AutoReverse="True" Duration="00:00:01">
        <DoubleAnimation Storyboard.TargetProperty="(RenderTransform).(ScaleX)" From="1" To="1.1"/>
        <DoubleAnimation Storyboard.TargetProperty="(RenderTransform).(ScaleY)" From="1" To="1.1"/>
```

```
    </Storyboard>
</Canvas.Resources>

<!-- Item 1 -->
<TextBlock x:Name="item1" Canvas.Top="0" FontFamily="Comic Sans MS" FontSize="30" Text="Menu Item 1"
           MouseEnter="SelectItem" MouseLeave="UnselectItem">
    <TextBlock.RenderTransform>
        <ScaleTransform/>
    </TextBlock.RenderTransform>
</TextBlock>

<!-- Item 2 -->
<TextBlock x:Name="item2" Canvas.Top="40" FontFamily="Comic Sans MS" FontSize="30" Text="Menu Item 2"
           MouseEnter="SelectItem" MouseLeave="UnselectItem">
    <TextBlock.RenderTransform>
        <ScaleTransform/>
    </TextBlock.RenderTransform>
</TextBlock>

<!-- Item 3 -->
<TextBlock x:Name="item3" Canvas.Top="80" FontFamily="Comic Sans MS" FontSize="30" Text="Menu Item 3"
           MouseEnter="SelectItem" MouseLeave="UnselectItem">
    <TextBlock.RenderTransform>
        <ScaleTransform/>
    </TextBlock.RenderTransform>
</TextBlock>
```

Et le code behind qui va avec

```
public void SelectItem(object sender, MouseEventArgs e)
{
    TextBlock item = sender as TextBlock;
    item.Foreground = new SolidColorBrush(Color.FromArgb(255, 255, 255, 255)); /* white */
    ItemAnimation.SetValue(Storyboard.TargetNameProperty, item.Name);
    ItemAnimation.Begin();
}

public void UnselectItem(object sender, EventArgs e)
{
    TextBlock item = sender as TextBlock;
    item.Foreground = new SolidColorBrush(Color.FromArgb(255, 0, 0, 0)); /* black */
    ItemAnimation.Stop();
}
```

Alors bien évidemment le code pour chaque Item se trouverait normalement dans un UserControl, puis nous aurions un 2ème UserControl chargé de contenir tout plein d'items et de les placer comme il faut.



Animation des items d'un menu

Résultat visible **ici**.

4-4 - Pluie d'images

Voici un petit exemple, dont j'ai pris l'idée sur le site officiel de **Silverlight**.
Je l'ai trouvé sympa donc je l'ai refait (en moins joli :) et je vous le fais partager.

```
<Canvas.Resources>
    <Storyboard x:Name="ImageRain">

  <DoubleAnimation Storyboard.TargetName="Rotate1" Storyboard.TargetProperty="Angle" Duration="00:00:03" To="360"
                   RepeatBehavior="Forever" />

  <DoubleAnimation Storyboard.TargetName="Translate1" Storyboard.TargetProperty="Y" Duration="00:00:06" To="500"
                   RepeatBehavior="Forever" />
      <Storyboard BeginTime="00:00:01" RepeatBehavior="Forever">

  <DoubleAnimation Storyboard.TargetName="Rotate2" Storyboard.TargetProperty="Angle" Duration="00:00:03" To="-360

  <DoubleAnimation Storyboard.TargetName="Translate2" Storyboard.TargetProperty="Y" Duration="00:00:03" To="500"
      </Storyboard>

  <DoubleAnimation Storyboard.TargetName="Rotate3" Storyboard.TargetProperty="Angle" Duration="00:00:04" To="360"
                   RepeatBehavior="Forever" />

  <DoubleAnimation Storyboard.TargetName="Translate3" Storyboard.TargetProperty="Y" Duration="00:00:07" To="500"
                   RepeatBehavior="Forever" />

  <DoubleAnimation Storyboard.TargetName="Rotate4" Storyboard.TargetProperty="Angle" Duration="00:00:05" To="-360
                   RepeatBehavior="Forever" />

  <DoubleAnimation Storyboard.TargetName="Translate4" Storyboard.TargetProperty="Y" Duration="00:00:03" To="500"
                   RepeatBehavior="Forever" />

  <DoubleAnimation Storyboard.TargetName="Rotate6" Storyboard.TargetProperty="Angle" Duration="00:00:06" To="-360
                   RepeatBehavior="Forever" />

  <DoubleAnimation Storyboard.TargetName="Translate6" Storyboard.TargetProperty="Y" Duration="00:00:12" To="500"
                   RepeatBehavior="Forever" />
      <Storyboard BeginTime="00:00:00.050" RepeatBehavior="Forever">

  <DoubleAnimation Storyboard.TargetName="Rotate5" Storyboard.TargetProperty="Angle" Duration="00:00:05" To="-360

  <DoubleAnimation Storyboard.TargetName="Translate5" Storyboard.TargetProperty="Y" Duration="00:00:05" To="500"
      </Storyboard>
    </Storyboard>
</Canvas.Resources>
<Image Source="images/img1.jpg" Canvas.Left="100" Width="50" Height="37.5">
    <Image.RenderTransform>
        <TransformGroup>
            <RotateTransform x:Name="Rotate1" Angle="0" />
            <TranslateTransform x:Name="Translate1" Y="-50" />
        </TransformGroup>
    </Image.RenderTransform>
</Image>
<Image Source="images/img2.jpg" Canvas.Left="350" Width="100" Height="75">
    <Image.RenderTransform>
        <TransformGroup>
            <RotateTransform x:Name="Rotate2" Angle="0" />
            <TranslateTransform x:Name="Translate2" Y="-100" />
        </TransformGroup>
    </Image.RenderTransform>
</Image>
<Image Source="images/img3.jpg" Canvas.Left="200" Width="75" Height="57">
    <Image.RenderTransform>
        <TransformGroup>
            <RotateTransform x:Name="Rotate3" Angle="0" />
            <TranslateTransform x:Name="Translate3" Y="-60" />
        </TransformGroup>
    </Image.RenderTransform>
</Image>
```

```
<Image Source="images/img4.jpg" Canvas.Left="350" Width="50" Height="37.5">
    <Image.RenderTransform>
        <TransformGroup>
            <RotateTransform x:Name="Rotate4" Angle="0" />
            <TranslateTransform x:Name="Translate4" Y="-200" />
        </TransformGroup>
    </Image.RenderTransform>
</Image>
<Image Source="images/img5.jpg" Canvas.Left="125" Width="125" Height="83">
    <Image.RenderTransform>
        <TransformGroup>
            <RotateTransform x:Name="Rotate5" Angle="0" />
            <TranslateTransform x:Name="Translate5" Y="-200" />
        </TransformGroup>
    </Image.RenderTransform>
</Image>
<Image Source="images/img6.jpg" Canvas.Left="400" Width="100" Height="75">
    <Image.RenderTransform>
        <TransformGroup>
            <RotateTransform x:Name="Rotate6" Angle="0" />
            <TranslateTransform x:Name="Translate6" Y="-200" />
        </TransformGroup>
    </Image.RenderTransform>
</Image>
```

Pluie d'images

Résultat visible **ici**.

5 - Conclusion

Nous venons de découvrir les différentes animations proposées dans Silverlight. Ces dernières vous permettent donc de rendre vos applications plus agréables, plus dynamiques... pour une meilleure expérience utilisateur.

6 - Remerciements

Je tiens à remercier toute l'équipe .NET, ainsi que **caro95470** pour sa correction orthographique.