# Image scrolling synced with large text scrolling

This article explains how to auto scroll a large image in sync with large text.

## Introduction

Sometimes you have a page that contains both a large image and a lot of text. The best UI would be to display both, but there simply isn't enough room on the screen. The temptation is to crop the image height in order to keep enough text displayed, but this doesn't always work effectively, particularly with larger images.

This article outlines another solution to this problem: display part of the image at the top of the page with the text below - then scroll the part of the image that is displayed at the same time as the readable text (the position of the area where the image is displayed does not change).

Image and text top

Image and text bottom

## Implementation

To allow the user to view the full image, we track the `ScrollViewer` of the text part and slide the image into view accordingly.

The layout of such a page is very simple, you'll only need an image and some text and we put this inside the `ContentPanel`.

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"
LayoutUpdated="ContentPanel_OnLayoutUpdated">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <ItemsControl Name="ImageControl" Height="175" Grid.Row="0">
            <Image Source="{Binding Image}" Name="Image">
                <Image.RenderTransform>
                    <CompositeTransform x:Name="ImageCompositeTransform"
TranslateY="{Binding ElementName=ScrollViewer, Path=VerticalOffset,
Converter={StaticResource ScrollViewerOffsetConverter}, ConverterParameter='inverse'}"
/>
                </Image.RenderTransform>
            </Image>
        </ItemsControl>
```


(C) Copyright Nokia 2014. All rights reserved.

```
        <ScrollViewer Name="ScrollViewer"
                      Grid.Row="1"
                      Margin="0,12,0,12"
                      VerticalAlignment="Top"
                      VerticalScrollBarVisibility="Visible"
                      ManipulationMode="Control"
                      Style="{StaticResource ScrollViewerStyle}">
            <RichTextBox Name="Article">
                <Paragraph>
                    <Run FontWeight="Bold" FontSize="22" Text="{Binding Title}" />
                </Paragraph>
                <Paragraph>
                    <LineBreak />
                    <Run Text="{Binding Body}" />
                </Paragraph>
            </RichTextBox>
        </ScrollViewer>
    </Grid>
</Grid>
```

With this XAML in place, you've already added the needed image animation. It's done through use of a `CompositeTransform` that will change the Y value representing the vertical axis image offset.

We are hooking up `TranslateY` value to the `ScrollViewer` of the text, so that the image animation will be kept in sync with the text scroll movement of the user.

Only thing left to do, is figure out how much we need to slide the image in reference to the scroll offset of the `ScrollViewer`. Because with a very large text, the image will slide slowly upwards, but with a small text, the animation will be faster.

This calculation is done when the detail page is loaded and passed along to the `ScrollViewerOffsetConverter`, that will set the offset `TranslateY` value.

In the code behind of the detail page we will do the actual calculation using the `OnLayoutUpdated` event of the `ContentPanel`. This is needed to be sure that the `ScrollableHeight` of the `ScrollViewer` is populated, because it will only be available when the full text has been initialized by the page.

```
private bool _sliderConvertValueSet = false;
private const double ZeroDouble = 0;

private void ContentPanel_OnLayoutUpdated(object sender, EventArgs e)
{
    if (!_sliderConvertValueSet && !Double.IsInfinity(ScrollViewer.ScrollableHeight) &&
!ScrollViewer.ScrollableHeight.Equals(ZeroDouble))
    {
        var slideValue = this.Image.ActualHeight - this.ImageControl.ActualHeight;
        var slideConvert = slideValue/this.ScrollViewer.ScrollableHeight;

        ScrollViewerOffsetConverter.ConvertValue = slideConvert;

        _sliderConvertValueSet = true;
    }
}
```

This calculation will seek out the slide offset value that is needed inside the `ScrollViewerOffsetConverter`. It's done by looking at the real `ActualHeight` of the image and subtracting the viewable part - represented by the `ActualHeight` of the `ImageControl`, giving us the **Height** that is missing/not visible.


(C) Copyright Nokia 2014. All rights reserved.

This value is than divided by the `ScrollableHeight` of the `ScrollViewer` giving us the multiplication value for the image `ScrollViewerOffsetConverter`. The code for the **Converter** looks like this

```
public object Convert(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
{
    if (!Double.IsInfinity(ConvertValue) && !ConvertValue.Equals(ZeroDouble))
    {
        double targetValue = ((double) value)*ConvertValue;

        if (((string) parameter).Equals("inverse", StringComparison.OrdinalIgnoreCase))
            return 0 - targetValue;

        return targetValue;
    }
    else
    {
        return value;
    }
}
```

You can indicate in what direction your image has to scroll - along with your text or the other way around.

So with that in place you'll get a nice animation while scrolling through the content of the article.

## Video demo

The media player is loading...

## Downloads

A complete working solution can be downloaded here: File:ImageParallaxList.zip


(C) Copyright Nokia 2014. All rights reserved.