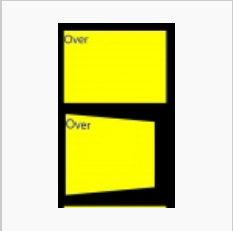
Double Sides Flipping Control - Windows Phone

This article explains how to create a *Double Sides Flipping Control* which flips across its center based on user's horizontal gesture.

22 Dec 2013

Introduction

If you have a list of items and you want to make some actions on each item, or just display more content. To achieve this, you can use a Double Sides Flipping control which allows you to put actions or related content at the back of items. It also allows the user to easily flip between the front and back sides with his finger. Some examples applications that use a similar interaction are 6Vine or 6Tag apps by Rudy Huyn (Amazing Developer).



I've implemented a custom control available here in codeplex which makes it easy to use such a scenario. The custom control has a Front Template and a Back Templates which you can define easily in the design.

In this article, we will see how to use this control in Windows Phone projects and also briefly cover how the control was implemented.

How to use the control

First of all, you need to reference the library in your project. You can install the Nuget package wpflipping or you can build the library from the source code available at Codeplex.

Note: The library target is Windows Phone 8 but almost the same code should work on Windows Phone 7. you can make one more step by recreate the project with windows phone 7.5 target.

- Define `wpflipping` namespace in XAML

```
xmlns:wpflipping="clr-namespace:WPFlipping;assembly=WPFlipping"
```

- Then add an instance of `wpflipping:DoubleSidesControl` in DataTemplate of the ListBox and define the Front and Back templates as show in the below snippet.

```
 <ListBox x:Name="FlipList" CacheMode="BitmapCache" ItemsSource="{Binding Lists}">
              <ListBox.ItemTemplate>
                  <DataTemplate>
                      <StackPanel Margin="12,32" >
                          <wpflipping:DoubleSidesControl>
                              <wpflipping:DoubleSidesControl.FrontTemplate>
                                  <DataTemplate>
                                      <StackPanel Background="Yellow">
                                          <TextBlock Text="Over" Foreground="Blue"
FontSize="48"/>
                                      </StackPanel>
                                  </DataTemplate>
                              </wpflipping:DoubleSidesControl.FrontTemplate>
                              <wpflipping:DoubleSidesControl.BackTemplate>
                                  <DataTemplate>
```


(C) Copyright Nokia 2014. All rights reserved.

```
                                        <StackPanel Background="White">
                                            <TextBlock Text="Under" Foreground="Blue"
FontSize="48"/>
                                        </StackPanel>
                                    </DataTemplate>
                                </wpflipping:DoubleSidesControl.BackTemplate>
                            </wpflipping:DoubleSidesControl>
                        </StackPanel>
                    </DataTemplate>
                </ListBox.ItemTemplate>
            </ListBox>
```

Tip: use CacheMode="BitmapCache" for smooth animation. It is useful when you need to animate, translate, or scale a UIElement as quickly as possible

- `ItemsSource="{Binding Lists}"` can be any collection in code behind or in `viewmodel`.

For example,

```
public MainPage()
        {
            InitializeComponent();
            FlipList.ItemsSource = new string[] { "First", "Second", "Third" };
        }
```

For the complete example, download the source code from codeplex here.

Inside the control

This section briefly describes how the control works.

Note: If you just want to use the control then you may not need to read this section.

- The structure of control consists of Grid with `x:Name="PlaceHolder"` which contains two grids "Over" and "Under".
- set `Canvas.ZIndex="1"` on Over grid and `Canvas.ZIndex="0"` on under. So Over will be on top initially.
- Then set `CenterOfRotationX`, `CenterOfRotationY` and `CenterOfRotationZ` value to 0.5 for `PlaceHolder`, Over and Under grids so they rotate around the center. Give the PlaceHolder's projection `x:Name="AnimatedPlaneProjection"`.

```
<ResourceDictionary
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:wpflipping="clr-namespace:WPFlipping">
    <Style TargetType="wpflipping:DoubleSidesControl">
        <Setter Property="Width" Value="420"/>
        <Setter Property="Height" Value="300"/>
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="wpflipping:DoubleSidesControl">
                    <Grid x:Name="PlaceHolder" CacheMode="BitmapCache">
                        <Grid.Projection>
                            <PlaneProjection x:Name="AnimatedPlaneProjection"
```


(C) Copyright Nokia 2014. All rights reserved.

```
RotationY="0" CenterOfRotationZ="0.5" CenterOfRotationX="0.5" CenterOfRotationY="0.5" />
                        </Grid.Projection>
                        <Grid x:Name="Under" Canvas.ZIndex="0" >
                            <ContentPresenter Content="{Binding}"
ContentTemplate="{TemplateBinding BackTemplate}"/>
                            <Grid.Projection>
                                <PlaneProjection CenterOfRotationZ="0.5"
CenterOfRotationX="0.5" CenterOfRotationY="0.5"/>
                            </Grid.Projection>
                        </Grid>
                        <Grid x:Name="Over" Canvas.ZIndex="1" >
                            <ContentPresenter Content="{Binding}"
ContentTemplate="{TemplateBinding FrontTemplate}"  />
                            <Grid.Projection>
                                <PlaneProjection CenterOfRotationZ="0.5"
CenterOfRotationX="0.5" CenterOfRotationY="0.5"/>
                            </Grid.Projection>
                        </Grid>
                    </Grid>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>
</ResourceDictionary>
```

- In `OnApplyTemplate` method of `DoubleSidesControl`, Rotate "Under" Grid 180 degree across Y axis.

```
 if (DesignerProperties.IsInDesignTool)
            {
                //Apply it only in DesignTime
                Under.Visibility = System.Windows.Visibility.Collapsed;
                Over.Visibility = System.Windows.Visibility.Visible;
            }
            else
            {
                //Apply It only in runtime, to make it easy to design Back Template in
Design Time(not Flipped)
                if (Under != null)
                {
                    if (Under.Projection != null && Under.Projection is PlaneProjection)
                    {
                        var underProjection = Under.Projection as PlaneProjection;
                        underProjection.RotationY = 180;
                    }
                    Under.Opacity = 0;
                }
            }
```

- In `PlaceHolder_ManipulationStarted` event, we will assign start rotation to know the rotate value of `PlaceHolder` grid before manipulation started .

```
startRotation = AnimatedPlaneProjection.RotationY;
```


(C) Copyright Nokia 2014. All rights reserved.

- In `PlaceHolder_ManipulationDelta` event, we will calculate the rotate value since the manipulation started based on the `e.CumulativeManipulation.Translation.X` and the `desiredFlipWidth`.

```
  private void PlaceHolder_ManipulationDelta(object sender,
System.Windows.Input.ManipulationDeltaEventArgs e)
        {
            //if (Math.Abs(e.CumulativeManipulation.Translation.Y) > 5) return;
            //if (Math.Abs(e.DeltaManipulation.Translation.X) < 2) return;
            var container = sender as FrameworkElement;
            double earlyFlipFactor = container.ActualWidth / 3;// to make it faster to
reach the middle
            double desiredFlipWidth = container.ActualWidth - earlyFlipFactor;
            double cumulative = e.CumulativeManipulation.Translation.X;
            double rotateValue = (cumulative / desiredFlipWidth) * -180;
            if (Math.Abs(e.CumulativeManipulation.Translation.X) < 2)
            {
                //If user is clicking
                return;
            }
            var PlaceHolderprojection = (PlaceHolder.Projection as PlaneProjection);
            double NextValue = (360 + startRotation + rotateValue) % 360;

            //Assign New Rotation Value
            PlaceHolderprojection.RotationY = NextValue;
            //Determine which part will be on top
            SetZIndex(NextValue);
            SetOpacity(NextValue);

        }
```

Two key points here are:

1. `earlyFlipFactor` which makes the grid flip early(before reaching middle). You can change it's value based on you scenario.
2. `SetZIndex` and `SetOpacity` methods will determine which grid (Over or Under Grid) will be on top and which one will be hidden based on the `rotateY` value of `PlaceHolder` grid.

```
      //Over Condition is true when Over Part should be on top, False if Under Part
should be on top
            bool OverCondition = (value >= 0 && value <= 90) || (value >= 270 && value
<= 360);
            Canvas.SetZIndex(Over, OverCondition ? 1 : 0);
            Canvas.SetZIndex(Under, OverCondition ? 0 : 1);
```

- Now the grid is rotating around the center axis and showing the Over or Under grid based on the rotate value. What is missing? If the user released the control then the control should continue continue the flipping to the nearest side.

`ContinueSwipeFlip` Method called from `PlaceHolder_ManipulationCompleted` event will continue the animation.

```
  private void PlaceHolder_ManipulationCompleted(object sender,
```


(C) Copyright Nokia 2014. All rights reserved.

```
System.Windows.Input.ManipulationCompletedEventArgs e)
        {
            if (Math.Abs(e.TotalManipulation.Translation.X) < 2)
            {
                //If user is clicking
                return;
            }
            // Continue the flipping animation
            ContinueSwipeFlip();
        }

        /// <summary>
        /// After you release the place holder and no longer flipping it,
        /// This method will fire Animation to continue the flipping to the nearest side
        /// </summary>
        private void ContinueSwipeFlip()
        {
            double LastValue = AnimatedPlaneProjection.RotationY;
            DoubleAnimation doubleAnimation = new DoubleAnimation();
            doubleAnimation.From = LastValue;
            double FullTime = 400;
            if (LastValue < 90 && LastValue >= 0)
            {
                doubleAnimation.To = 0;
                doubleAnimation.Duration = TimeSpan.FromMilliseconds(FullTime *
(LastValue) * 1e-2);
            }
            else if (LastValue >= 90 && LastValue < 180)
            {
                doubleAnimation.To = 180;
                doubleAnimation.Duration = TimeSpan.FromMilliseconds(FullTime * (180 -
LastValue) * 1e-2);
            }
            else if (LastValue >= 180 && LastValue < 270)
            {
                doubleAnimation.To = 180;
                doubleAnimation.Duration = TimeSpan.FromMilliseconds(FullTime *
(LastValue - 180) * 1e-2);
            }
            else
            {
                doubleAnimation.To = 360;
                doubleAnimation.Duration = new
Duration(TimeSpan.FromMilliseconds(FullTime * (360 - LastValue) * 1e-2));
            }
            SetZIndex(doubleAnimation.To.Value);
            SetOpacity(doubleAnimation.To.Value);
            doubleAnimation.AutoReverse = false;
            Storyboard.SetTarget(doubleAnimation, AnimatedPlaneProjection);
            Storyboard.SetTargetProperty(doubleAnimation, new
PropertyPath("RotationY"));
            Storyboard sb = new Storyboard();
            sb.Children.Add(doubleAnimation);
            sb.Begin();
        }
```


(C) Copyright Nokia 2014. All rights reserved.

Summary

This article showed how to implement a flip-able user control, and briefly covered how it was implemented. If you wish to use this control, it is available here.

(C) Copyright Nokia 2014. All rights reserved.