How to Create More Transitions

# Overview

In this release, we provide a basic extension model that allows you to add new transition effects. You’re required to add the code directly inside the projects and recompile the solution. In the next release, we may consider to add a better extension model so the recompilation can be eliminated.

In order to add a new transition, you need to work on both the Windows Phone project (VideoStoryCreator), and the native component project (NativeVideoEncoder).

# Add a transition to the Windows Phone project:

To add a transition to the Windows Phone project, first create a transition class. To do this, under the Transitions folder, create a new class. This class must either inherit the TransitionBase class, or implement the ITransition interface directly. TransitionBase provides some default implementations, such as serialize/deserialize common properties to/from xml.

## Implement the transition

At a minimum, a transition class derived from TransitionBase must implement abstract methods Begin and Stop. Those two methods begin and stop the transition animation. The built-in FadeTransition and FlyInTransition use storyboards to implement the animation. But you’re free to choose whatever mechanism you like.

The TransitionBase class also provides two properties: BackgroundElement and ForegroundElement. You can override them if you need to add additional information in getter/setter. The background element is the next image that will be transited to. The foreground element is the image currently being displayed. Both properties are of type FrameworkElement. In this release, you cannot access the raw bitmap pixel data inside a transition class. We know this limits the kinds of transitions you can create. We may consider enabling this scenario in the future. As for now, you can implement transitions that are not bitmap specific. For example, use Silverlight’s perspective 3D to create a 3D flip effect.

If your transition provides additional properties (properties not defined in ITransition), you should define the properties inside the transition class as well. In addition, you must override the Save and LoadChild methods to serialize/deserialize the properties. Both methods accept a parameter of type XElement, which is the parent element for your transition. You can add new attributes or child elements to the parent element, but please do not modify any existing attributes/child elements. You can refer to the FlyInTransition class for a sample.

## Create a transition designer

If your transition provides additional properties (properties not defined in ITransition) that require user configuration, you must create a transition designer. To do so, add a new UserControl to the Transitions folder. For example, the FadeTransition doesn’t need any additional properties. So no designer is required. On the other hand, the FlyInTransition requires an additional property Direction. So we must provide a designer class which will be displayed on the compose page when the user chooses this transition.

A designer must be a UserControl. To make it fit the parent compose page’s layout, it is recommended to use a Grid as the root element, and set the Grid’s Width and Height to Auto. Then create two columns inside the Grid, whose widths are 0.4\* and 0.6\*, respectively. In the left column, put TextBlocks that label the property. In the right column, put input controls. The input controls should be data bound to your property. Since the data context passed to your designer is a view model class, you can access your own property in the form of Transition.YourProperty. For example:

{Binding Transition.Direction, Mode=TwoWay}

You may need to create a converter if your property’s type is quite complex.

It is recommended not to modify anything not related to your transition, such as an image’s display time.

## Register your transition

Now that your transition and its designer (if needed) are ready, you need to register them. To do so, you have to modify an existing code file. Open the TransitionFactory class, and register your transition and its designer in the constructor. For example:

\_transitionNameTypes.Add("Fly In Transition", typeof(FlyInTransition));

\_transitionNameDesigners.Add("Fly In Transition", typeof(FlyInTransition\_Design));

\_transitionNames.Add("Fly In Transition");

If your transition doesn’t have a designer, simply remove the second line of code.

This concludes the work you need to do to add a new transition to the Windows Phone project.

# Add a transition to the native project:

To add a transition to the native project, first create a transition class. You’re recommended to provide both the .h file and the .cpp file. This class must inherit the TransitionBase class.

## Implement the transition

While the sample solution uses WIC and Media Foundation to encode the video, we do not pass the details of WIC and Media Foundation to your transition. We simply provide several byte arrays (BYTE\*) that represent the pixel data of the input/output frames. So even if you don’t know WIC/Media Foundation, you can still write your own transition. The only requirement is basic C++ knowledge and an understanding of your transition’s algorithm.

At a minimum, a transition class must implement the abstract method GetOutputFrame. This method accepts a parameter ranges from 0 to 1. This parameter indicates the current time for the transition. For example, if a transition lasts 10 seconds, and this parameter’s value is 0.3, it means the current frame is for second 3. In general, you don’t need to care about how long a transition is to be displayed. You only need to translate this value ranged from 0 to 1 to something that is meaningful to your transition.

The return value of the GetOutputFrame method is a BYTE pointer, which is the pixel data of the image representing the current video frame. The TransitionBase class provides a protected member m\_outputFrame, which can be used. But this member is not initialized by TransitionBase. You must initialize it yourself. To make the encoded video correct, it is recommended to initialize it using the following code:

this->m\_outputFrame = new BYTE[this->GetFrameSize()];

Inside the implementation of the GetOutputFrame method, you may want to use the following protected members:

* m\_frameWidth/ m\_frameHeight: The width/height of the current frame.
* m\_backgroundFrame: The next image to be transited to. The type is BYTE\*, which represents the pixels of the image.
* m\_foregroundFrame: The current image being displayed. The type is BYTE\*, which represents the pixels of the image.

The pixel data of the source images passed to you use the RGBA format.

If your transition provides additional properties (properties not defined in TransitionBase), you must override the ParseXml method to deserialize the properties. This method accepts a parameter of type XmlParser\*, which is a homebrew xml parser. The parser implements a basic SAX API. When it is passed to you, the current element being read is the parent element representing the transition. You can use the GetAttributes to obtain all attributes of the current element, whose return type is a map<wstring, wstring>. For example, the following code obtains the value of an attribute named Direction:

wstring direction = pParser->GetAttributes()[L"Direction"];

If your transition adds sub elements to the parent transition element, you must call ReadStartElement to move to the sub element. Once you’ve finished, do not forget to call ReadEndElement, similar to how you use other SAX APIs.

## Register your transition

Now that your transition is ready, you need to register it. To do so, you have to modify an existing code file. Open the TransitionFactory.cpp class, and register your transition in the CreateTransition method. Since C++ doesn’t support reflection, we have to use a switch case branch. For example:

else if (transitionName == L"Fly In Transition")

{

\*output = new FlyinTransition();

}

This concludes the work you need to do to add a new transition to the native project. Since the native component is used inside the Worker Role to automatically encode the video, no designer class is needed.