Animation Basics Silverlight vs. QML

This article provides a comparative overview of the Qt and Windows Phone 7 animation frameworks.

Overview

XAML

QML animation elements can be found in QML Animation and Transitions, and Silverlight animation elements from System.Windows.Media.Animation Namespace.

In both development frameworks the animations are generally handled by applying animation to property change values, with this article only some general differences and similarities with basic animation handling is discussed.

Following code shows a simple QML animation definition

```
NumberAnimation {
  id: animateOpacity
  target: flashingblob
  properties: "opacity"
  from: 0.09
  to: 1.0
  duration: 1000
  loops: Animation.Infinite
}
```

This animation could then be started by calling `animateOpacity.start()` from some command handler. With Silverlight XAML(Extensible Application Markup Language) the same animation could be defined for example as:

```
<Storyboard x:Name="myStoryboard">
  <DoubleAnimation
  Storyboard.TargetName=" flashingblob"
  Storyboard.TargetProperty="Opacity"
  From="0.09" To="1.0" Duration="0:0:1"
  AutoReverse="True" RepeatBehavior="Forever" />
 </Storyboard>
```

And again this animation could be started by calling `myStoryboard.Begin();` in some command handler.

This simple snipped illustrates the one major difference, basically where in QML animation are defined as own instances, in Silverlight XAML they are defined as Storyboards.

With both methods you define the target and target property, you can define to and from values, loops etc. The basic difference is the format of the language used. There are of course minor differences, for example with QML if you have multiple animations, you need to group the animations either with SequentialAnimation or ParallelAnimation, with Silverlight you simply define them inside the same storyboard, and use the BeginTime to define when within the storyboard each animation should start.

So for example, if you would want to do two animations parallel that both have 3 animation sequences, you could define something like this with QML:

```
ParallelAnimation {
  SequentialAnimation {
  NumberAnimation { ...}
  NumberAnimation { ...}
  NumberAnimation { .. }
  }
  SequentialAnimation {
  ColorAnimation { ...}
  ColorAnimation { ... }
  ColorAnimation { .. }
  }
}
```

And then you could try doing the same with Silverlight like this:

```
<Storyboard ...>
  <DoubleAnimation BeginTime="00:00:00" .../>
  <DoubleAnimation BeginTime="00:00:02" .../>
  <DoubleAnimation BeginTime="00:00:04" .../>
  <ColorAnimation BeginTime="00:00:00" .../>
  <ColorAnimation BeginTime="00:00:02" .../>
  <ColorAnimation BeginTime="00:00:04" .../>
</Storyboard>
```

Another option with Silverlight would be to use the KeyFrames-animations instead, then then you could split it in easier looking structure, and do something like this:

```
<Storyboard x:Name="Grow">
  <DoubleAnimationUsingKeyFrames BeginTime="00:00:00">
  <LinearDoubleKeyFrame .../>
  <DiscreteDoubleKeyFrame .../>
  <SplineDoubleKeyFrame .. />
  </DoubleAnimationUsingKeyFrames>
  <ColorAnimationUsingKeyFrames BeginTime="00:00:00">
  <LinearColorKeyFrame .../>
  <DiscreteColorKeyFrame .../>
```

```
  <SplineColorKeyFrame .../>
  </ColorAnimationUsingKeyFrames>
</Storyboard>
```

Now basically the two groups are set to start same time, and inside the groups the different animations are run in sequence.

Note that if you are not using easing on QML, you would be using the Linear-Frames with Silverlight, since it would match the usage pretty well. The Discrete-frame could be used to replace PauseAnimation, basically since it is simple waiting the time and then doing the change.

The Spline-frame would be a way on doing custom easing's within the animation, with it you can define two control points which specify the cubic Bezier curve which defines the progress of the key frame. Note that you can use the EasingFunction to define easing with Silverlight just as you would define them with QML. The Spline-frame just offers additional way on doing it.

Standalone Animations

With QML if you want animation that you can start and stop from code, you would define the animation separately, give it an id tag and then use the start()/stop()/pause() etc. functions to utilize it. For example if you would want to animate the opacity change for a rectangle when it is clicked, you could do something like this:

```
Rectangle {
  id: flashingblob
  width: 75; height: 75
  color: "blue"
  opacity: 1.0

  MouseArea {
  anchors.fill: parent
  onClicked: {
  animateOpacity.start()
  }
}

  NumberAnimation {
  id: animateOpacity
  target: flashingblob
  properties: "opacity"
  from: 0.09
  to: 1.0
  duration: 1000
  loops: Animation.Infinite
  }
}
```

And with Silverlight, if you would be using a canvas the XAML code could look something like that

```
<Canvas x:Name="LayoutRoot" Background="Transparent">
  <Canvas.Resources>
  <Storyboard x:Name="myStoryboard">
  <DoubleAnimation
  Storyboard.TargetName="myrectanggle"
  Storyboard.TargetProperty="Opacity"
  From="0.09" To="1.0" Duration="0:0:1"
  AutoReverse="True" RepeatBehavior="Forever" />
  </Storyboard>
  </Canvas.Resources>

  <Rectangle Name="myrectanggle" Width="75" Height="75" Fill="Blue" MouseLeftButtonDown="myrectanggle_MouseLeftButtonDown"/>

</Canvas>
```

Basically meaning that instead of using the animation as variable you would define a storyboard, and define the animations inside that storyboard. Then to start this animation you would call `Begin();` inside your C# code for the class, for example you could start the animation in the mouse down handler for the rectangle as follows:

```
private void myrectanggle_MouseLeftButtonDown(object sender, MouseButtonEventArgs e){
  myStoryboard.Begin();
}
```

Automatic animations

With QML you can get animations executed automatically for signal handlers, for example mouse click animation could be handled as follows:

```
MouseArea {
  anchors.fill: parent
  onClicked: PropertyAnimation {…}
}
```

Code doing the similar behavior with Silverlight, could be defined using the event triggers, for example like this:

```
<EventTrigger SourceName="stopButton" RoutedEvent="Button.Click">
  <BeginStoryboard>
  <Storyboard>
  <DoubleAnimation … />
  </Storyboard>
  </BeginStoryboard>
</EventTrigger>
```

When defining the animation outside the object itself, you need to define the source object for the event as shown above, and if you define the trigger inside the object, you don't need to define the source, since you are inside it already. For example you could use following code for starting animation

during the canvas loaded event, from inside the canvas definition:

```
<Canvas.Triggers>
  <EventTrigger RoutedEvent="Canvas.Loaded">
  <BeginStoryboard>
  <Storyboard>
  ...
  </Storyboard>
  </BeginStoryboard>
  </EventTrigger>
</Canvas.Triggers>
```

As an addition with Silverlight XAML you could also define the animation to be executed within some events, and also you could define the triggers inside the style definition, then the code would look something like this:

```
<Style.Triggers>
  <Trigger Property="IsMouseOver" Value="True">
  <Trigger.EnterActions>
  <BeginStoryboard>
  <Storyboard>
  ...
  </Storyboard>
  </BeginStoryboard>
  </Trigger.EnterActions>
  <Trigger.ExitActions>
  <BeginStoryboard>
  <Storyboard>
  ...
  </Storyboard>
  </BeginStoryboard>
  </Trigger.ExitActions>
  </Trigger>
</Style.Triggers>
```

QML also has the nice Behavior defined for animation, this is used for starting animation on value changes with properties. With Silverlight XAML the similar animations can be handled with `DataTrigger`. Following example shows an example implementation within a style definition:

```
<Style.Triggers>
  <DataTrigger Binding="{Binding MyValue[0].UpVisibility}" Value="1.0">
  <DataTrigger.EnterActions>
  <BeginStoryboard>
  <Storyboard>
  ...
  </Storyboard>
  </BeginStoryboard>
  </DataTrigger.EnterActions>
  </DataTrigger>
</Style.Triggers>
```

examples

Simple Silverlite (WP 7.0) example can be found from: File:Silver SimpleAnimation1.zip