Archived:Animation for Games in Flash Lite

Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

We do not recommend Flash Lite development on current Nokia devices, and all Flash Lite articles on this wiki have been archived. Flash Lite has been removed from all Nokia Asha and recent Series 40 devices and has limited support on Symbian. Specific information for Nokia Belle is available in Flash Lite on Nokia Browser for Symbian. Specific information for OLD Series 40 and Symbian devices is available in the Flash Lite Developers Library.

Introduction

Flash is an ideal authoring environment for creating rich-media digital content like games and other applications that leverage sophisticated graphics. There are various forms of interacting with such digital assets, the prominent of them being animating them and manipulating the same. Animations encompasses moving of the item around the screen whereas manipulating deals with modifying the asset itself (like changing brightness or contrast). Flash allows its users to both animate and modify the digital assets in his library.

Flash allows animation both via design and through programming constructs. The focus of this document is to achieve them programmatically in Actionscript 2.

The Tween class

Tween class is the class that is used to animate objects on stage dynamically. The associated classes are resident in

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
```

Hence, it is necessary to import them. To animate the movieclip on stage, simply write

```
someTween = new mx.transitions.Tween( object, property, function,begin, end, duration,
useSeconds )
```

This is the constructor of this Tween class where - *object = Name of the movieclip* property = MovieClip property to change (like X-coord or y-coord) function = Type of Tween (smooth, regular, strong etc). begin, end = The values of the property at beginning and end of the animation duration = the no. of Frames/seconds useSeconds = Boolean value that is used to control animation either by time or frame-wise.

```
mcTween:Tween = new Tween( mc, "_x", Strong.easeOut, 20, 100, 5, true );
```

This statement animates the movieclip mc from x-coord 20 to 100 in 5 seconds.

Importance in Games

As we know, animations or tweens are central to games. When applied in right logistics they help a lot in storytelling and improve the game experience. Creating simple movements and animations are common in games done with Flash and these are quite commonplace (and are not discussed here). Few animations and their associated scenarios are discussed here.

Flip/inversion This is a very common animation employed in games. The simplest use case can be the coin-toss before a game. This can be achieved by -

```
new Tween(coinpic_mc, "_xscale", Regular.easeIn, 100, -100, 13, false);
// Horizontal flip
```

Remember that the registration point of the movieclip must be the centre of this coin graphic.Only then the coin toss would appear immaculate. However for a coin toss, it is more appropriate to do a _yscale than _xscale.

Hero Introduction In a lot of games, the entrance of the game's protagonist is grand. And a common technique used is to magnify him or to increase the alpha value. This gives the actor/hero more prominence or attention.

```
new Tween(hero_mc, "_xscale", Regular.easeIn, 100, 200, 12, false);
new Tween(hero_mc, "_yscale", Regular.easeIn, 100, 200, 12, false);
// Magnify effect

new Tween(hero_mc, "_alpha", Regular.easeIn, 20, 100, 12, false);
// Alpha or Prominence effect
```

Death Death characterizes the loss of energy and this can once again be directly associated with _alpha property of the character. This effect has been excessively used in Nintendo style video games, where a dead/defeated character fades out of the screen upon death.

```
death = new Tween(hero_mc, "_alpha", Regular.easeIn, 100, 5, 12, false);
// Remarks death of character
```

Since the character is dead, it is important that he is cleaned from the memory. So it is important to perform removeMovieClip() after the completion of the above tween.

```
death.onMotionFinished = function() {
 // Do a removeMovieClip or deal with it accordingly
};
```

Conclusion

This article discusses some common scenarios in games where animation can be effectively used to convey story/message. Its implementation in Flash Lite is also shown in the article.