Pour réaliser des interpolations de mouvements, il existe de nombreuses techniques, à base de code ou à travers la manipulation de la Timeline (Scénario).

En matière de programmation, l'ActionScript 3 propose une classe par défaut (intitulée Tween()), qui permet d'interpoler la valeur d'une propriété, mais elle s'avère un peu lourde à gérer lorsque vous devez réaliser des interpolations "élaborées".

Pour réaliser des interpolations à base code, il existe sur le web de nombreuses classes écrites par des développeurs indépendants, mais nous allons principalement parler dans cette ressource de la classe TweenMax() qui reste l'une des plus connues et des plus efficaces.

Lien de téléchargement de la classe : Classe TweenMax

Comparaison des différents tests entre classes : http://blog.greensock.com/tweening-speed-test/

Remarque : Comme vous pouvez le constater, la classe TweenMax est beaucoup plus efficace que les autres, la classe Adobe Tween() étant la moins efficace.

Synthèse

Si vous avez déjà lu le contenu de cette ressource (paragraphes ci-dessous jusqu'à la fin de cette page), voici une synthèses des lignes d'instructions à utiliser avec la classe TweenMax :

```
import com.greensock.*;

import com.greensock.easing.*;

TweenMax.to(nomOccurrence, 2, {x:350,delay:1, ease:Bounce.easeOut});
```

Classe TweenMax

Cette classe présente l'avantage de pouvoir réaliser des interpolations sur de nombreuses propriétés (y compris des interpolations de couleurs et filtres), elle inclut également la possibilité d'utiliser une trajectoire définie par une courbe (de Bezier).

Interpolation à partir de la classe TweenMax()

1. Commencez par télécharger la classe TweenMax .
2. Placez le dossier intitulé com dans le même dossier que celui qui contient votre .fla.
3. Placez une occurrence sur la scène et nommez la **barre**.
4. Insérez les deux lignes d'instruction ci-dessous dans votre programme.

```
import com.greensock.*;

TweenMax.to(barre, 3, {x:350});
```

Voilà, la programmation de l'interpolation est terminée, une seule ligne suffit, car la syntaxe est extrêmement simplifiée en comparaison de la classe Tween() de l'AS2/AS3. Essayons à présent de comprendre la structure de cette ligne d'instruction.

Remarque : Si vous souhaitez ajouter un effet de ralentissement ou d'accélération au début ou à la fin de l'interpolation, vous devrez ajouter une ligne d'instruction supplémentaire pour importer une autre classe. Consultez l'exemple qui suit les explications ci-dessous.

Explications

Pour commencer, vous ne devez pas instancier la classe TweenMax(), vous y faites simplement référence en la combinant avec la méthode to().

```
TweenMax.to()
```

Vous devez ensuite spécifier le nom d'une occurrence suivi d'une durée d'interpolation et d'un objet (délimité et défini par les accolades) qui contient les propriétés à utiliser lors de l'interpolation.

```
barre, 3, {x:350}
```

Cela doit vous donner :

```
TweenMax.to(barre, 3, {x:350});
```

Effets de mouvements applicables à une interpolation

Pour appliquer un effet au début ou à la fin de l'interpolation, utiliser le paramètre **ease** comme dans l'exemple ci-dessous :

```
TweenMax.to(barre, 2, {rotation:45, ease:Bounce.easeOut});
```

Vous pouvez remplacer easeOut par easeIn ou easeInOut et Bounce par :

- Linear, Elastic, Back, Strong, Sine...

Exécuter une instruction lorsqu'une interpolation est terminée

Parmi les propriétés que vous spécifiez entre les accolades, vous allez devoir ajouter **onComplete** suivi du nom de la fonction à exécuter.

```
import com.greensock.*;

 import com.greensock.easing.*;

TweenMax.to(barre, 2, {x:350,delay:1, ease:Bounce.easeOut,onComplete:finInterpolation});
```

```
function finInterpolation() {

    nomOccurrence.scaleX = 2;

    nomOccurrence.scaleY = 2;

}
```

Retarder l'exécution d'une interpolation

A la lecture du titre de cette explication, on pourrait s'interroger sur l'intérêt de cette fonctionnalité car il semble plus simple d'exécuter une instruction au moment on la ligne ci-dessous est exécutée…

```
TweenMax.to(barre, 2, {x:350,delay:1});
```

N'oubliez pas que la propriété **delay** retarde l'exécution d'une interpolation, c'est pourquoi, il sera parfois utile de faire appel au script ci-dessous :

```
import com.greensock.*;

TweenMax.to(nomOccurrence, 2, {x:350,delay:2,onStart:initialiserTaille});

function initialiserTaille() {

    nomOccurrence.scaleX = 0.5;
```

```
    nomOccurrence.scaleY = 0.5;


}
```

La fonction ci-dessus ne va s'exécuter qu'à partir du moment où l'interpolation va démarrer et non au moment ou la ligne d'instruction TweenMax.to() est exécutée.

Exécuter une instruction durant une interpolation

Durant tout le déroulement d'une interpolation, vous aurez peut-être besoin d'exécuter une ligne d'instruction (ex. : diminuer ou augmenter une valeur, changer l'opacité et l'angle de rotation d'une occurrence, etc.).

```
import com.greensock.*;


 import com.greensock.easing.*;


TweenMax.to(nomOccurrence, 4, {x:350,delay:2,onUpdate:faireTourner});


function faireTourner() {


    nomOccurrence.rotation++;


}
```

Exécuter une interpolation en suivant une courbe de Bezier

Si vous maîtrisez bien les courbes de Bezier, utilisez alors la syntaxe ci-dessous qui vous propose une propriété supplémentaire

```
var passagesPoints:Array = [{x:180, y:20}, {x:60, y:200}]
```

```
TweenMax.to(zoneDemo, 3, {x:350, y:250, bezier:passagesPoints});
```

Si vous souhaitez intégrer directement les coordonnées des points directeurs que doit atteindre une occurrence, utilisez cette syntaxe :

```
TweenMax.to(zoneDemo, 3, {x:350, y:250, bezier:[{x:180, y:20}, {x:60, y:200}]});
```

Attention : L'occurrence ne vas pas "passer par" les points dont vous spécifiez les coordonnées, ce sont des coordonnées de points directeurs.

Interpoler plusieurs occurrences en même temps

La technique est extrêmement simple, il vous suffit d'utiliser la méthode **allTo()** en lieu et place de to(). Ainsi, vous n'écrirez pas...

```
TweenMax.to(bt1,0.6,{rotation:45});
TweenMax.to(bt2,0.6,{rotation:45});
TweenMax.to(bt3,0.6,{rotation:45});
```

... mais plutôt ...

```
TweenMax.allTo([bt1,bt2,bt3],0.6,{rotation:45});
```

Comme vous pouvez le constater, le premier paramètre (le nom de l'occurrence que vous spécifiez lorsqu'il n'y en a qu'une à interpoler) est un tableau comprenant l'ensemble des noms d'occurrences à interpoler).

Retour d'interpolation

Pour effectuer une interpolation, puis la refaire dans l'autre sens, il vous suffit d'utiliser la propriété **yoyo** et de la régler à **true** et de définir la propriété associée **repeat** à 1. Cela devrez vous donner :

```
TweenMax.to(bt1,0.6,{rotation:45,yoyo:true,repeat:1});
```

N.B. : Si vous spécifiez une valeur supérieure à 1, vous obtiendrez une boucle reproduisant

l'interpolation autant de fois que vous l'avez spécifié.

Pour définir un intervalle entre le retour d'interpolation, spécifiez le temps à l'aide de la propriété **repeatDelay**.

```
TweenMax.to(bt1,0.6,{rotation:45,yoyo:true,repeat:1,repeatDelay:5});
```

Dans cet exemple, une interpolation se produit, puis, 5 secondes plus tard, le retour se déclenche.

Référez-vous aux scripts proposés dans la palette Yazo 6 pour des exemples complémentaires.