Cours pratiques en langage actionScript 3.

Ver : 10012010.
Niveau 1
public graphistes web.

Partie 3

Les évènements dans flash.

Un nombre important d'objets composites dans flash émettent des signaux permettant de savoir si la souris interagit avec lui, si un objet externe est correctement chargé, si une interpolation est en cours ou si elle commence ou se termine. Certains autres permettent de mettre en œuvre des intervalles de temps pour créer du mouvement ou des décomptes. Les objets liés aux médias diffusés comme le son et la vidéo émettent eux aussi des signaux permettant de connaître leur état de chargement ou de lecture. Cette liste, nous permet de deviner l'importance que prend le système de diffusion d'évènement dans flash si l'on souhaite enrichir ses animations.
Maitriser les interactions dans flash nécessite de savoir quel objet va diffuser quel évènement. Pour cela il faudra le plus souvent se renseigner dans la documentation pour savoir quels évènements sont décrits.
En revanche il sera bon de se rappeler que les objets suivants sont en mesure d'émettre tous les évènements souris :

- MovieClip.
- Sprite.
- Textfield.
- SimpleButton.

Etant donné le nombre considérable des évènements disponibles, nous n'en détaillerons ici que les plus courants ceux concernant la classe MouseEvent (classe décrivant les différents évènements souris possibles), ou Event (classe générale décrivant des évènements très divers).

Un évènement actionScript 3 c'est quoi ?

Un évènement est un objet composite tout comme peut l'être le type MovieClip ou Sprite, cet objet est automatiquement instancié si l'action qu'il décrit est déclenchée. Puis, il est diffusé au travers d'un diffuseur d'évènement EventDispatcher qui est un autre objet composite au travers de sa méthode dispatchEvent (événement). Pour plus de commodité l'objet EventDispatcher est directement intégré aux objets interactifs que nous avons cités plus haut, nous en verrons la raison par la suite.

Pour utiliser ce système il suffit de placer un écouteur d'évènement sur l'objet diffuseur au travers de la méthode :
objet_a_ecouter.addEventListener(Evt_a_ecouter , fonction_a_declencher)

Pour créer une interaction dans flash, il faut donc effectuer ces trois choses :

- Sélectionner un objet qui diffuse l'événement recherché.
- Abonner un écouteur sur cet objet.
- Créer une fonction qui traitera l'évènement s'il est déclenché.

Tous les systèmes permettant les interactions ou diffusant des informations dans flash sont basés sur cette même formule. Il existe bien une ou deux exceptions basés sur les processus en vigueur dans les précédentes versions (diffusion vidéo, interpolations externes), que nous évoquerons en temps utiles.
Ce qu'il faut retenir ici c'est simplement le trio : écouteur, diffuseur, fonction à déclencher.

Créer une interaction souris

Nous avons sur la scène un MovieClip auquel nous avons donné « mc_lien » comme nom d'occurrence. On souhaite le rendre cliquable et lancer un lien vers une autre page du site.

mc_lien.addEventListener(MouseEvent.CLICK,envoiLien)

```
function envoiLien ( even : MouseEvent) : void
{
      var requete : URLRequest = new URLRequest("http://www.google.com");
      navigateToURL(requite, "_blank");
}
```

Voici donc notre première interaction mise en place. Regardons maintenant notre fonction évènementielle un peu plus en détail, car elle répond elle aussi à un schéma très strict.

Une fonction liée à un évènement doit TOUJOURS avoir un paramètre en parenthèse : Ce paramètre est l'instance de l'objet évènementiel diffusé par l'élément écouté (ici l'objet est de type MouseEvent). Il contient des propriétés et méthodes qui permettent d'obtenir des informations concernant l'objet écouté et la nature de l'évènement (comme par exemple connaître les coordonnées locales de la souris sur l'objet lors du clic). Le paramètre évènementiel fait aussi toujours référence à l'objet qui l'a émis au travers de la propriété target.

Une fonction évènementielle ne peut avoir qu'UN SEUL paramètre. En l'occurrence celui que nous venons de voir. Pour utiliser des fonctions ou variables non déclarées dans la fonction il faudra que celle-ci soit déclarée de manière à pouvoir être vue par l'ensemble des fonctions. Sachez qu'il existe aussi la possibilité de créer ses propres évènements permettant de faire transiter des données lorsque la première solution n'est pas applicable.

Une fonction évènementielle ne peut renvoyer aucune valeur. L'utilisation du mot clé « return » en fin de fonction provoquera une erreur. Cela est toutefois logique puisque l'appel de la fonction est effectué automatiquement et qu'il

n'existe de ce fait aucune façon de récupérer une éventuelle valeur issue de l'exécution du code.

Si vous vous posiez encore la question de savoir ou et comment placer variables et autres fragments de codes, les quelques règles que nous venons de voir sont déterminantes. Ce sont celles-ci, en fait, qui décideront si une variables doit être localisée dans une fonction ou à la racine du code, si tels objet doit être accessible depuis la fonction évènementielle. C'est le point d'entrée vers l'architecture de votre application.

Stopper l'écoute d'un évènement.

Nous avons besoin de stopper l'écoute d'un évènement lorsque nous ne voulons plus que la fonction soit déclenchée ou lorsqu'il devient nécessaire de supprimer l'objet qui diffuse l'évènement. Pour ce dernier cas il est nécessaire de connaître un peu mieux le fonctionnement du lecteur flash : Pour ce programme, un élément ne peut être supprimé de la mémoire que si plus aucun évènement n'y est attaché, sinon la trace de l'objet reste en mémoire et si il est lourd graphiquement, il peut ralentir voire compromettre le bon fonctionnement de l'application.
Dans le principe tout écouteur placé doit pouvoir être retiré avant suppression de l'objet sur lequel il est placé. Nous utilisons pour cela la méthode removeEventListener(type_evenement, fonction_evenementielle).

```
var mc_lien :MovieClip=new MovieClip();
mc_lien.graphics.beginFill(1,1);
mc_lien.graphics.drawRect(0,0,200,40);
mc_lien.addEventListener(MouseEvent.CLICK, envoiLien);
mc_lien.buttonMode = true ;
addChild(mc_lien);

function envoiLien (even :MouseEvent):void
{
      var requete : URLRequest = new URLRequest("http://www.google.com");
      navigateToURL(requete, "_blank");
      mc_lien.removeEventListener( MouseEvent.CLICK , envoiLien);
      mc_lien.buttonMode = false ;
}
```

L'exemple ici montre comment rendre un bouton cliquable une seule fois. Le clic déclenche la fonction qui renferme l'instruction de désabonnement et la désactivation du pointeur « main » sur le clip. En outre si nous supprimons ce clip il sera effectivement éligible à la suppression totale du cache de l'animation.

Savoir se servir de propriétés et méthodes du paramètre de la fonction évènementielle.

Il reste un élément du système évènementiel que nous n'avons pas abordé : Il s'agit de la variable « even » inscrite entre les parenthèses de notre fonction. Nous savons donc qu'il s'agit de l'objet instancié lorsque l'action à été déclenché. Cet objet contient des propriétés et méthodes communes à l'ensemble des objets

évènementiels. Par exemple target désigne l'objet qui à déclenché l'évènement. Nous avons réécrit la fonction de l'exemple précédent en nous servant du paramètre target :

```
function envoiLien (even :MouseEvent):void
{
      var requete : URLRequest = new URLRequest("http://www.google.com");
      navigateToURL(requite, "_blank");
      even.target.removeEventListener( MouseEvent.CLICK , envoiLien);
      even.target.buttonMode = false ;
}
```

D'autres propriétés en revanche sont propres au type d'évènement en cours. L'objet MouseEvent contient de nombreuses informations, telles que les coordonnées de la souris sur l'objet ou la scène au moment du clic. Pour permettre les combinaisons de touches, il est possible encore de savoir si la touche « ctrl »(ou « command »), « alt » ou « maj » sont enfoncées et permet. L'ensemble des objets de type évènement auront toujours des propriétés liées à leur nature évènementielle. Les événements diffusés par une interpolation permettent d'en contrôler très finement le fonctionnement et même de les combiner en diffusant des signaux au début, fin, et à chaque changement de valeur de l'objet interpolé.

Exercice : Mettez en place un menu de trois boutons avec un trace dans chaque fonction.

Charger une image ou un swf externe.

Le chargement d'une image ou d'un fichier swf externe est une opération très courante. Grace à ce système vous pourrez mettre en place des sites entiers en alliant un système de chargement externes aux menus que vous devriez savoir mettre en place arrivé à ce point du cours.

Tout d'abord il est important de souligner que le chargement d'un media est susceptible de diffuser un nombre important d'évènements tels que le début, la progression, le signalement d'une erreur de chargement, la fin du chargement et quelques autres encore. Nous allons ici juste appréhender la base du chargement externe au travers du chargement simple d'une image et d'un swf. Cette base utilise uniquement la classe d'évènement Event. Sachez toutefois que le signalement d'erreur de chargement est géré par le type d'objet IOErrorEvent, et que les signaux de progression qui nous permettront de mettre en place une barre de chargement est assuré par la classe ProgressEvent.
Nous faisons la connaissance de l'objet Loader qui fait partie des objets affichables dans flash. C'est l'objet qui nous sert à charger le media. celui-ci est équipé d'un diffuseur d'évènement (loader.contentLoaderInfo) qui va servir à émettre tout les signaux du téléchargement. En nous interessant à la documentation nous nous apercevons que deux évènements peuvent correspondre à notre attente :
complete
Dispatched when data has loaded successfully.LoaderInfo

init
Dispatched when the properties and methods of a loaded SWF file are accessible.LoaderInfo

La différence entre les deux réside dans le fait que l'évènement init assure qu'un swf ou une image permettront toutes les propriétés et méthodes dont dispose un movieClip ou un bitmap. Nous choisirons donc d'écouter cet évènement.
Nous allons abonner un écouteur qui réagira lorsque l'évènement INIT sera émis par l'objet contentLoaderInfo. Il sera alors possible de le redimensionner, si c'est un bitmap de le lisser en cas de redimensionnement, de le centrer dans un container puis *in fine* de l'afficher sur la scène.

```
var container_principal:MovieClip=new container();
addChild(container_principal);
var requete:URLRequest=new URLRequest("images/wizld.jpg");
var chargeur:Loader=new Loader();
chargeur.contentLoaderInfo.addEventListener(Event.INIT,chargementTermine);

function chargementTermine(even:Event)
{
      container_principal.addChild(chargeur);
      //centrage du media
      chargeur.x= container_principal.width / 2 – chargeur.width / 2;
      chargeur.y= container_principal.height / 2 – chargeur.height / 2;

      chargeur.contentLoaderInfo.removeEventListener(Event.INIT,chargementT
ermine);

}
```

Gérer le chargement une image ou un swf externe.

Comme je l'ai évoqué, le chargement d'une image émet plus d'évènements que ceux que nous avons vus. Ces évènement peuvent nous permettre en particulier d'eviter des erreurs qui bloqueraient notre application si le chargement ne peut s'effectuer ou d'afficher une animation de chargement. Tout les évènements sont diffusés au travers de l'objet contentLoaderInfo contenu par l'instance de notre loader. En observant l'objet LoaderInfo dont hérite ContentLoaderInfo.

A / Gestion des erreurs.
Un problème de chargement (connexion coupée, mauvais lien) peut mettre en péril la stabilité et le bon déroulement d'une image. Aussi prévoir ces types de comportement peut nous permettre d'afficher quand même un symbole par défaut, de la même manière qu'en html. Pour ce faire il est utile de prévoir dans son animation un clip que nous pourrons mettre en place par le script en l'exportant au travers de son panneau de liaison. Nous le ferons passer pour l'image chargée au cas où une erreur est émise. Dans ce cas deux objet peuvent apparemment nous servir. Le premier « flash.events.HTTPStatusEvent » informe le lecteur flash sur la présence ou l'absence de connexion internet. Il est émis dans l'un ou l'autre cas. Il contient la propriété status qui envoie le code de connexion http correspondant. Nous n'avons pas un réel besoin d'écouter cette évènement dans le cadre de notre application.

Le lecteur en cas de lien brise ou d'image corrompue envoie également un évènbement qui spécifie l'erreur : `flash.events.IOErrorEvent.IO_ERROR`
En écoutant cet évènement il nous est possible d'ajouter une gestion des erreurs :
Nous créons un clip dans la bibliothèque avec « container » comme nom de liaison. Il contient eventuellement un decor.

```
var container_principal:MovieClip=new container();
addChild(container_principal);
var requete:URLRequest=new URLRequest("images/wizld.jpg");
var chargeur:Loader=new Loader();
chargeur.contentLoaderInfo.addEventListener(Event.INIT,chargementTermine);
chargeur.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR,
erreurChargement);

function erreurChargement (even: IOErrorEvent):void
{
      // retrait des ecouteurs
chargeur.contentLoaderInfo.removeEventListener(Event.INIT,chargementTermine);
chargeur.contentLoaderInfo.removeEventListener(IOErrorEvent.IO_ERROR,erreurChargeme
nt);

      // instanciation d'un clip destiné à remplacer l'image manquante
      var img_defaut : MovieClip = new ImageDefaut();
      container_principal.addChild(img_defaut);
      // j'envoie un signal de fin de chargement
      chargeur.contentLoaderInfo.dispatchEvent(new Event(Event.INIT));

}

function chargementTermine(even:Event):void
{
      container_principal.addChild(chargeur);

      //centrage du media
      chargeur.x= container_principal.width / 2 – chargeur.width / 2;
      chargeur.y= container_principal.height / 2 – chargeur.height / 2;

      chargeur.contentLoaderInfo.removeEventListener(Event.INIT,chargementTermine);
      chargeur.contentLoaderInfo.removeEventListener(IOErrorEvent.IO_ERROR,erreur);

}
```

Note : Dans la fonction « erreurChargement » nous diffusons un évènement INIT destiné à enclencher un nouveau chargement dans le cas d'un chargement par lot, comme celle d'une galerie. Dans ce cas nous pourrions écouter l'évènement init afin de déclencher le chargement suivant.

A Création et gestion d'une animation de chargement.

Le chargement d'une image lourde (le lecteur flash permet l'affichage d'un bitmap de 2880 pixels de cotés) ou d'un swf contenant des élément bitmap peut prendre un certains temps. Il peut être bienvenu dans ce cas d'afficher une animation de chargement ou d'attente afin de ménager la patience du visiteur. Pour ce faire nous aurons besoin d'écouter deux évènements supplémentaires, et d'ajouter une instruction (disparition de l'animation) à notre fonction de fin. Event.OPEN nous informe du debut du chargement de l'image. Nous insererons donc un ecouteur qui se chargera dans la fonction d'afficher et de centrer notre animation.

ProgressEvent.PROGRESS quand à lui est une classe d'évènement qui permet en consultant ses propriétés de connaître le nombre de bytes chargés. Nous connaissons également à travers lui le nombre de bytes à charger. En réalisant une simple opération nous pouvons obtenir un decimal entre 0 et 1 que nous pouvons appliquer à l'échelle horizontale (scaleX) d'un clip contenant un graphique en forme de barre. L'évènement est déclenché à chaque chargement d'un nouveau paquet de donnée. Il peut donc créer l'illusion d'une barre animée.

// un clip contenant un rectangle montrant la barre de chargement dans son état final est créé dans la bibliothèque (contient un champ texte : « champ_texte » et le clip du graphique : « barre ». On lui donne « BarreChargement » comme identifiant de liaison.

```
var barre_load : MovieClip= new BarreChargement();

var container_principal:MovieClip=new container();
addChild(container_principal);
var requete:URLRequest=new URLRequest("images/wizld.jpg");
var chargeur:Loader=new Loader();
chargeur.contentLoaderInfo.addEventListener(Event.INIT,chargementTermine);
chargeur.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR,
erreurChargement);
// ajout d'une ecoute sur le commencement et la progression du chargement
chargeur.contentLoaderInfo.addEventListener(Event.OPEN,departChargement);
chargeur.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS,chargementEnCour
s);

function departChargement(even:Event):void
{
      //centrage de la barre de chargement par rapport à son état final
      barre_load.x= container_principal.width / 2 – barre_load.width / 2;
      barre_load.y= container_principal.height / 2 – barre_load.height / 2;

      // reduction de l'echelle de la barre à son état minimal
      barre_load.barre.scaleX=0;

      //ajout à la scène
      container_principal.addChild(barre_load);

}

function chargementEnCours(even:ProgressEvent):void
{
      var ratio_chargement:Number= even.bytesLoaded / even.bytesTotal;
      var pourcent:Number= Math.round(ratio_chargement * 100);
      var texte:String = pourcent +" %";
      barre_load.champ_texte.text = texte;
      barre_load.barre.scaleX = ratio_chargement;
}

function erreurChargement (even: IOErrorEvent):void
{
      // retrait des ecouteurs
chargeur.contentLoaderInfo.removeEventListener(Event.INIT,chargementTermine);
chargeur.contentLoaderInfo.removeEventListener(IOErrorEvent.IO_ERROR,erreurChargeme
nt);
chargeur.contentLoaderInfo.removeEventListener (Event.OPEN,departChargement);
chargeur.contentLoaderInfo.removeEventListener
(ProgressEvent.PROGRESS,chargementEnCours);


      // instanciation d'un clip destiné à remplacer l'image manquante
      var img_defaut : MovieClip = new ImageDefaut();
      container_principal.addChild(img_defaut);
```

```
        // j'envoie un signal de fin de chargement
        chargeur.contentLoaderInfo.dispatchEvent(new Event(Event.INIT));

}

function chargementTermine(even:Event):void
{
        // retrait de la barre de chargement :
        container_principal.removeChild(barre_load);

                //centrage du media
        chargeur.x= container_principal.width / 2 – chargeur.width / 2;
        chargeur.y= container_principal.height / 2 – chargeur.height / 2;

        container_principal.addChild(chargeur);
        chargeur.contentLoaderInfo.removeEventListener(Event.INIT,chargementTermine);
        chargeur.contentLoaderInfo.removeEventListener(IOErrorEvent.IO_ERROR,erreur);
chargeur.contentLoaderInfo.removeEventListener (Event.OPEN,departChargement);
chargeur.contentLoaderInfo.removeEventListener
(ProgressEvent.PROGRESS,chargementEnCours);

}
```

C/ optimisation du swf chargé (scaling)ou de l'image (lissage).

Nous allons compléter notre code par un processus qui va redimensionner notre média et lui appliquer un lissage si nous détectons qu'il s'agit d'un bitmap.

Le redimensionnement doit tenir compte du format vertical ou horizontal du media chargé. Nous devrons donc tenir compte de la hauteur et largeur.
Le redimensionnement ne s'applique que si le medias est PLUS grand que le conteneur. Aussi nous aurons besoin de savoir si la hauteur de l'image depasse la hauteur du cadre et ainsi de suite.
Nous nous servons pour cela de la classe math qui contient plusieurs méthodes de comparaison de valeur numérique. L'une Math.max(valeur_1, valeur_2) renvoie la valeur la plus grande. Math.min(valeur_1, valeur_2) renvoie la valeur la plus petite. C'est celle-ci dont nous avons besoin pour notre comportement de réduction. Nous emploierons des ratios de manière à modifier l'echelle plus simplement.

```
var ratio_largeur:Number= (container_principal.width-30 )/ chargeur.width;
var ratio_hauteur:Number= (container_principal.height-30) /
chargeur.height;
var ratio_final:Number= Math.min(ratio_largeur,ratio_hauteur);
if(ratio_final<1)
chargeur.scaleX=chargeur.scaleY=ratio_final;
```

Le lissage est uniquement utilisable sur les objets de type Bitmap. Aussi, si nous voulons garder la capacité de charger aussi des swf, nous aurons besoin d'utiliser une condition en nous servant du mot clé du mot clé « is » permettant de déterminer si l'instance est de tel ou tel type.
Pour effectuer ce test nous aurons besoin de la propriété content de notre instance loader qui contient effectivement l'objet chargé pour la comparer au type bitmap.

```
if (chargeur.content is Bitmap)
```

Si tel est le cas nous aurons besoin de transformer le contenu en un objet bitmap à par entière.

```
if (chargeur.content is Bitmap)
{
      var bmp:Bitmap = Bitmap(chargeur.content);
      bmp.smoothing = true;
}
```

Ce procédé est nécessaire lorsque nous manipulons un objet que nous savons être d'un type, mais que flash traite pour raison pratique sous une forme plus générique dont dépend l'objet en question. Par exemple tout objet imbriqué dans un clip peut être récupéré avec la fonction getChildByName. L'objet renvoyé l'est sous la forme DisplayObject dont dépendent l'ensemble des objets affichable. Pour accéder à la propriété text d'un champ texte par ce biais nous aurions également besoin de le transtyper.

Voici le code intégré à notre fonction de fin de chargement :

```
function chargementTermine(even:Event):void
{
      // Retrait de la barre de chargement :
      container_principal.removeChild(barre_load);
      // extraction du contenu :
      var contenu : DisplayObject = chargeur.content;

      // Redimensionnement
      var ratio_largeur:Number= (container_principal.width-30 )/
      contenu.width;
      var ratio_hauteur:Number= (container_principal.height-30) / contenu
      .height;
      var ratio_final:Number= Math.min(ratio_largeur,ratio_hauteur);
      if(ratio_final<1)
      contenu.scaleX = contenu.scaleY = ratio_final;

      // Lissage si bitmap
      if (contenu is Bitmap)
      {
            var bmp:Bitmap = Bitmap(contenu);
            bmp.smoothing = true;
      }

      //centrage du media
      contenu.x= container_principal.width / 2 – contenu.width / 2;
      contenu.y= container_principal.height / 2 – contenu.height / 2;

      container_principal.addChild(contenu);

      chargeur.contentLoaderInfo.removeEventListener(Event.INIT,chargementTermine);
      chargeur.contentLoaderInfo.removeEventListener(IOErrorEvent.IO_ERROR,erreur);
      chargeur.contentLoaderInfo.removeEventListener (Event.OPEN,departChargement);
      chargeur.contentLoaderInfo.removeEventListener
      (ProgressEvent.PROGRESS,chargementEnCours);

}
```

Transformer l'image chargée en objet cliquable

Il est fréquent également de charger une vignette qui servira de lien vers une image plus grande. Abonnons quelques évènement à notre objet afin d'améliorer la convivialité de notre application.
L'objet Loader ne convient en revanche pas pour créer ces interactions car il ne permet pas d'utiliser. Nous ne pouvons pas appliquer l'écouteur à l'objet content , car le type bitmap ne permet pas cela non plus. Nous pourrions l'appliquer a container_principal. Mais le comportement pourrait nous réserver de mauvaise surprise. Aussi, devrions-nous insérer le contenu dans un conteneur de type clip avant son centrage dans la fonction.

Voici le code intégré à notre fonction de fin de chargement :

```
function chargementTermine(even:Event):void
{
	// Retrait de la barre de chargement :
	container_principal.removeChild(barre_load);
	// extraction du contenu :
	var contenu : DisplayObject = chargeur.content;

	// Redimensionnement
	var ratio_largeur:Number= (container_principal.width-30 )/
	contenu.width;
	var ratio_hauteur:Number= (container_principal.height-30) / contenu
	.height;
	var ratio_final:Number= Math.min(ratio_largeur,ratio_hauteur);
	if(ratio_final<1)
	contenu.scaleX = contenu.scaleY = ratio_final;

	// Lissage si bitmap
	if (contenu is Bitmap)
	{
		var bmp:Bitmap = Bitmap(contenu);
		bmp.smoothing = true;
	}

	// insertion dans une enveloppe diffusant les évènement souris
	var enveloppe : MovieClip = new MovieClip ;
	enveloppe.addChild(contenu);

	//centrage de l'enveloppe (contenu peut être remplacé par enveloppe
puissqu'il est imbrique dans celui-ci
	enveloppe.x= container_principal.width / 2 – contenu.width / 2;
	enveloppe.y= container_principal.height / 2 – contenu.height / 2;

	// Modification de l'imbriquation (« contenu » pour « enveloppe »)
	container_principal.addChild(enveloppe);

	chargeur.contentLoaderInfo.removeEventListener(Event.INIT,chargementTermine);
	chargeur.contentLoaderInfo.removeEventListener(IOErrorEvent.IO_ERROR,erreur);
	chargeur.contentLoaderInfo.removeEventListener (Event.OPEN,departChargement);
	chargeur.contentLoaderInfo.removeEventListener
	(ProgressEvent.PROGRESS,chargementEnCours);

	// ajout des comportements interactifs

	enveloppe.addEventListener(MouseEvent.CLICK,clicVignette);
	enveloppe.addEventListener(MouseEvent.ROLL_OVER,entreeVignette);
	enveloppe.addEventListener(MouseEvent.ROLL_OUT,sortieVignette);
```

```
	enveloppe.buttonMode = true;
}


function clicVignette (even:MouseEvent)
{
	//on placera ici le comportment de chargement de l'image principale
	var vignette:DisplayObject= even.target as DisplayObject;
	vignette.filters=[new GlowFilter(),new BevelFilter()]

}

function entreeVignette (even:MouseEvent)
{
	var vignette:DisplayObject= even.target as DisplayObject;
	vignette.filters=[new GlowFilter(),new BevelFilter(),new
DropShadowFilter(10,45,0x234567,0.5,4,4)];

}

function sortieVignette (even:MouseEvent)
{
	var vignette:DisplayObject= even.target as DisplayObject;
	vignette.filters=[];

}
```

Nous avons l'occasion de noter ici l'autre méthode de transtypage permise par flash. Certaines expression ne permettront pas le mot clé « as » ou au contraire d'utiliser l'expression TYPE(OBJET_A_TRANSTYPER), le panneau d'erreur est très explicite dans le cas d'une erreur.

Combiner les écouteurs pour créer des applications interactives.

L'art de créer des applications riches en interactivité et en mouvement tient principalement à la maitrise des processus évènementiels en œuvre dans le logiciel. Dans de nombreux cas il faudra mettre en œuvre une système qui traitera un interaction souris qui mettra en place l'écoute d'un autre évènement jusqu'à ce que la souris soit déplacée.

Créer un mouvement à l'aide du modèle évènementiel.

Certains évènements, tels qu'un clic souris, le signal de fin de chargement d'une image externe sont envoyés une seule fois à chaque fois que l'utilisateur actionne le bouton gauche de sa souris. D'autres sont diffusés plusieurs fois et déclenchent à chaque fois la fonction qui lui est associée. Certains d'entre eux sont même diffusés de façon régulière. Ces évènements s'avèrent très utiles pour crées du mouvement, ou un déroulement dans le temps. Nous allons en découvrir un des plus courants : L'événement ENTER_FRAME :

La diffusion de l'évènement ENTER_FRAME est assurée par les MovieClip et les objets héritant de MovieClip à savoir la scène de notre animation. Le comportement de cet évènement est d'envoyer un signal à chaque fois que la tête de lecture du clip entre dans une nouvelle image. Attention, ce comportement n'est pas dépendant de l'état de lecture du clip. Il envoie donc cet évènement même si le clip est stoppé.
L'évènement ENTER_FRAME est dépendant de la cadence de l'animation. Une animation cadencée à 12 images par secondes déclenchera la fonction associée 12 fois en une seconde.
Ce comportement nous permet donc de créer un défilement dans le temps à cadence rapide… Donc, du mouvement bien sur, mais aussi des vérifications redondantes pour des jeux ou d'autres applications à haute interactivité. Grace à cette fonction on entre de plein pied dans ce qu'il n'est pas possible de réaliser en s'appuyant sur la ligne de scénario de flash.

Une première utilisation : Remplacer le curseur de la souris par un clip

Sur un nouveau fichier nous importons une image que nous insérons dans un clip. Nous le posons sur la scène avec « image » comme nom d'occurrence.
Nous créons un second clip contenant une forme vectorielle créée dans l'environnement auteur un cercle rempli avec un dégradé radial. La couleur est sans importance puisque la forme nous servira de masque. En revanche, il faut que la couleur soit à alpha = 100 en son centre et 0 sur les bords. Nous déposons également ce clip sur la scène avec « lorgnette » comme nom d'occurrence.

Nous créons un calque « actions » dans lequel nous écrivons ce code.

```
// Ces deux lignes sont indispensables pour que le dégradé alpha soit pris
en compte. cacheAsBitmap sert à ce que le lecteur transforme la forme
vectorielle en tableau de pixels.

lorgnette.cacheAsBitmap = true;
```

```
image.cacheAsBitmap = true;

// Masquage de l'image par la forme
image.mask=lorgnette;
// nous rendons le pointeur de la souris invisible
Mouse.hide();

this.addEventListener(Event.ENTER_FRAME,pointer)

function pointer (even:Event)
{
      lorgnette.x=mouseX;
      lorgnette.y=mouseY;
}
```

Support : masque.

Se servir d'algorithmes pour créer des effets réalistes

L'exemple suivant est un petit script qui permet un remplacement du pointeur par un objet graphique. Nous avons crée un Clip contenant le bitmap d'un helicoptère ce clip à été placé sur la scène avec le nom d'occurrence « helico »

```
            helico.mouseEnabled=false;
            helico.addEventListener(Event.ENTER_FRAME, frame);
            addChild(helico);
            helico.cacheAsBitmap= true;
            Mouse.hide();

            function frame(even:Event):void
            {
//ces formules permettent d'obtenir un effet d'acceleration /
ralentissement progressifs.
                  helico.x=0.8*helico.x + 0.2*stage.mouseX;
                  helico.y=0.8*helico.y + 0.2*stage.mouseY;
                  helico.rotation = -0.15 * (helico.x - stage.mouseX);

// L'inversion de l'echelle permet de simuler l'orientation du clip (-1
crée une symetrie).
                  var sens:Number = 1;
                  if(helico.rotation < 0)
                  sens = -1
                  helico.scaleX = sens;
            }
```