Département
Education et
Technologie

- Environnement
- Animation
- Interactivité

CRÉATION D'ANIMATIONS INTERACTIVES EN FLASH

(Notes provisoires)

Monique Colinet

5.85
Mai 2004

Centre pour la formation à
l'Informatique dans le Secondaire

Ces notes sont destinées aux personnes suivant la formation "Création d'animations interactives avec Flash"
Elles sont provisoires et incomplètes. Elles seront remaniées en fonction de vos remarques, corrections, commentaires...
Vous êtes invités à communiquer toutes vos observations à

monique.colinet@fundp.ac.be

Introduction

Qu'est-ce que Flash ? Que peut-on faire avec Flash ?

Flash est présenté comme étant un outil de conception d'animations vectorielles pour le Web. Il génère des fichiers de petite taille, ce qui représente un avantage important au vu des vitesses de connexion encore relativement lentes qu'utilisent encore beaucoup d'internautes. Il permet de dynamiser un site Web pour le rendre plus attrayant.

Si son utilisation est souvent confinée à Internet, elle ne s'y limite pas. Les possibilités de présentations sont infinies: création de boutons flash intégrables dans une page *HTML*, réalisation de dessins animés, ...

De plus, avec ses fonctions avancées, ce logiciel permet également d'intégrer la dimension de l'interactivité. Cette interactivité peut se faire à l'intérieur même du produit qui est alors utilisé comme un logiciel auteur de création multimédia.

Grâce à ses fonctions avancées de programmation, l'interactivité peut également se faire en connexion avec un serveur de bases de données *MySQL* par l'intermédiaire de l'interface *PHP*.

Avec ce programme, il est donc possible de faire dans le plus simple mais aussi dans le plus complexe.

Avant de commencer

Puisqu'il est question d'animations vectorielles, il est peut-être bon de rappeler ce que sont les images vectorielles à l'inverse des images bitmap (celles que l'on rencontre le plus souvent sur le Web).

Qu'est-ce qu'une image vectorielle ?

L'image vectorielle est une représentation conceptuelle de formes calculées par des formules mathématiques (exemple: un carré n'est pas déterminé par des pixels mais par une formule mathématique qui détermine la longueur d'un côté, son emplacement, sa couleur, son orientation,...)

Le format vectoriel ne convient pas pour les images complexes, comme des photos par exemple, mais est surtout utilisé pour des représentations schématiques (*DAO*, *3D*, ...). Il peut se montrer très simple d'utilisation pour les modifications des images, comme par exemple pour changer une partie de la couleur de l'objet dessiné ou encore les dimensions de cet objet. Dans chacun de ces deux exemples, il s'agira de modifier une partie de la formule et non pas une partie des pixels. Un autre avantage de ce format est le poids des fichiers qu'il génère. (Un carré repéré par la coordonnée de ses 4 sommets pèsera toujours le même poids quelle que soit sa dimension.)

l'image **bitmap** d'un trait est faite d'un ensemble de pixels:

- 1ère ligne: point blanc, point blanc,...,point blanc, point noir, point blanc, ...point blanc
- 2ème ligne: point blanc, point blanc,...,point blanc, point noir, point blanc, ...point blanc

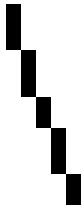
- 3ème ligne: point blanc, point blanc,...,point blanc, point noir, point blanc, ...point blanc
- etc pour chaque ligne

L'image **vectorielle** est, par contre, un ensemble de fonctions mathématiques:

- on donne les coordonnées de départ d'un point
- on fait tracer le trait
    - avec la longueur (a)
    - avec une épaisseur de trait (b)
    - avec une couleur (c)
    - suivant un angle (d)

Donc, si on agrandi l'image bitmap, la taille du fichier sera elle aussi agrandie tandis qu'une image vectorielle agrandie ne change pas de taille: seul un paramètre sera modifié (dans ce cas, le paramètre (a).
D'autre part, une image vectorielle agrandie restera aussi nette que l'originale tandis qu'une image bitmap agrandie présentera une pixelisation.
Un exemple concret:

Trait taille normale: 

Trait grossi 10 fois: 

Trait taille normale: 

Trait grossi 10 fois: 

Par contre, une image vectorielle est beaucoup plus exigeante, en terme de processeur, qu'une image bitmap puisque les points de l'image qui devront être affichés ne sont pas enregistrés dans le fichier et devront être calculés à l'affichage sur l'écran.

Découverte de l'environnement

C'est par l'intermédiaire de l'outil de conception que nous allons découvrir ce qu'est *FLASH*. Il s'agit d'un logiciel de création d'animations. Pour utiliser efficacement ce logiciel, il faut tenir compte de deux paramètres fondamentaux: **l'espace** et le **temps**. Il répond aux mêmes grands principes qu'un logiciel de montage vidéo.

Les créations d'animations doivent être envisagées dans deux directions: en fonction du temps (puisqu'il s'agit d'une suite d'images qui défilent) et en fonction de l'espace (sur deux images différentes les objets ne sont pas situés au même endroit "puisqu'il existe une animation"). Nous allons donc découvrir ces deux éléments différents mais qui ne peuvent être dissociés.

Le temps (partie scénario dans l'environnement)

L'animation est une suite d'images qui défilent à un certain rythme. Ces images peuvent être particulières: les images-clés qui seront enregistrées dans leur totalité avec toutes leurs caractéristiques. C'est également sur ces images-clés que le concepteur de l'animation insère les objets de la séquence. Les autres images ("normales") seront calculées par le logiciel en fonction de leurs différences avec la ou les précédentes et la ou les suivantes. Nous y reviendrons plus en détails lors des *interpolations*. D'autres images encore peuvent être "vides" (ne contenant pas d'objet "graphique") mais seulement des informations d'action tel qu'un arrêt par exemple. En bref, les images-clés permettent les modifications, les images simples servent essentiellement à modifier la durée de l'animation.

Pour une question d'organisation, de structure, plusieurs images peuvent être superposées. Dans ce cas, elles sont placées sur des *calques* (sortes de transparents sur lesquels on place des objets). Cependant, certains calques peuvent ne contenir que des images vides (comme précisé plus haut) mais seulement des informations d'actions.

On pourrait comparer les différents calques à différentes pellicules de films qui sont superposées et qui sont lues simultanément.

Ces calques peuvent être de types différents (normal, guide, masque, dossier,...) et avoir des propriétés différentes: masqué ou verrouillé. L'utilisation de ces calques est un passage obligé si on veut un travail clair et structuré.

L'espace (partie séquence ou scène dans l'environnement)

C'est, en réalité, l'agrandissement d'une image du scénario. C'est donc la zone de travail par excellence. C'est aussi la feuille de papier (en réalité: l'image)sur laquelle on dessine, on insère du texte ou du son et qui sera vue lors de la lecture de l'animation. C'est dans cette zone que se fera la majeure partie du travail de composition, le scénario étant réservé à l'organisation des informations insérées dans la séquence (que l'on appelle aussi scène).

Le contenu visible de cette séquence peut être variable suivant que le scénario contient un ou plusieurs calques et que ceux-ci sont visibles ou masqués. Il faut donc être bien conscient que le travail que l'on effectue sur la scène se fait en réalité sur une sorte de transparent et qu'il est

possible de travailler sur plusieurs transparents d'une même image. Il faut pour cela se situer avec précision dans le scénario. On voit déjà que les deux environnements ne sont pas indissociables. De plus, les scènes de travail doivent toujours être positionnées sur des images-clés du scénario, sans quoi, les modifications réalisées prendront cours dès l'image-clé précédente du même calque.

Une manipulation pratique vous permettra de comprendre ce fonctionnement.

Les autres panneaux disponibles

Le logiciel contient plus des fenêtres volantes plutôt que des barres d'outils. Ces fenêtres mobiles sont appelées panneaux et peuvent être cachés ou se dévoiler en fonction des besoins de l'utilisateur. Ce qui rend l'interface personnalisable.

Les barres d'outils

Ce sont les barres d'outils de base semblables à celles que l'on retrouve dans tous les programmes actuellement et leurs contenus sont bien entendu adaptés au logiciel.

Les invariants de ces barres d'outils sont toujours situés dans les mêmes menus, ce qui ne doit pas perturber le nouvel utilisateur. Citons les incontournables *ouvrir*, *enregistrer sous*, *couper*, *copier*, *coller*, ...Un item spécifique à l'utilisation des différentes fenêtres (ou panneaux) est accessible depuis la barre d'outils principale.

Les panneaux

Le but n'est pas de rédiger des notes sur les différents panneaux possibles car ils sont nombreux mais de reprendre les principaux. Chacun pourra bien sûr en fonction de son type de produit, utiliser plutôt l'un que l'autre. Citons quand même:

les outils de dessin

ceux-ci permettent, comme le nom l'indique, de dessiner des formes particulières ou à main levée, de sélectionner des objets dessinés, de les modifier (forme, couleur, orientation, déformation,...). C'est également parmi ces outils de dessin que l'on retrouve l'item *texte* qui permet d'insérer des zones de texte dans l'image.

Le mélangeur de couleurs et le nuancier

Ce mélangeur de couleurs permet de créer et de modifier des couleurs unies, dégradées, texturées ou transparentes. Il est possible de sélectionner une teinte selon trois modes de définition des couleurs: le mode RVB (rouge-vert-bleu), le mode TSL (teinte-saturation-luminosité) et le mode hexadécimal.

Le nuancier, quant à lui, affiche les 216 couleurs unies de la palette par défaut de *Flash*. Il s'agit des 216 couleurs communes aux systèmes ***Windows*** et ***Macintosh***. Cette palette pourra être personnalisée. Ce qui entraîne que chaque fichier Flash possède sa propre palette de couleurs. Celle-ci est disponible à chaque ouverture de fichier.

Les propriétés

Cette fenêtre affichera automatiquement les propriétés de l'objet sélectionné sur la scène ou de l'image sélectionnée sur le scénario, c'est-à-dire qu'elle permet la modification de cet objet. S'il s'agit d'un objet de type trait, il est possible de modifier sa couleur, son épaisseur, son type de trait, sa dimension, sa position,...

Les actions

Un module de programmation est intégré dans ce logiciel. Il utilise un langage propriétaire: *l'ActionScript*. Cette fenêtre permettra d'insérer du code soit en utilisant le mode **Normal**, et dans ce cas, il suffira de choisir l'action à faire exécuter (une sorte d'assistant à la programmation), soit en utilisant le mode **Expert** et, dans ce cas, il faudra que le concepteur encode les lignes de programmation. A ce panneau d'actions, on peut associer d'autres panneaux comme le panneau de **Référence** qui donne un lexique des différents termes du langage *ActionScript* ou encore le **Débogueur** qui permet de tester les actions insérées dans l'animation.

Les bibliothèques

La bibliothèque sert à stocker certains éléments de l'animation, notamment des symboles, des éléments multimédia. Elle permet la réutilisation d'un même élément sur une ou plusieurs scènes. L'intégralité de cet objet ne sera enregistré qu'une seule fois. Pour toutes les copies de cet élément ou objet, seules les modifications par rapport à l'original seront enregistrées. Le terme générique utilisé dans ce logiciel pour les copies d'objets est: *occurrence* de symboles.

Chaque fichier *Flash* possède sa propre et unique bibliothèque. Elle est accessible lorsque le fichier est ouvert et se ferme lors de la fermeture d'un fichier. Elle reste associée au fichier en cas de transfert.

L'explorateur d'animations

La puissance de l'outil permet une animation constituée de plusieurs scénarios contenant eux-mêmes des animations. A partir d'un scénario principal, il est possible d'imbriquer d'autres scénarios contenant eux aussi des animations. A partir de là, l'animation peut se complexifier avec des imbrications sur plusieurs couches et des interactions, voire des échanges de données entre celles-ci.

Pour pouvoir visualiser la structure et naviguer dans celle-ci, le module approprié est l'explorateur d'animations qui permet une visualisation plus ou moins développée du travail en cours. Il permet donc de travailler dynamiquement avec la scène.

Le lecteur

Ce module va permettre à tout moment, lors de la création de visualiser le travail que l'on est en train de réaliser.

Les caractéristiques générales des fichiers générés

Les propriétés des documents

Ces propriétés vont avoir une influence sur l'affichage de l'animation au moment de sa lecture. Le concepteur peut décider entre-autres de:

- ses dimensions: il s'agit de la grandeur de la fenêtre qui va s'ouvrir lors du lancement de la lecture du fichier. Ces dimensions peuvent se définir en pixels ou en points, en centimètres, en millimètres, en pouces ou encore en pouces transformés directement en centimètres. Ces différentes dimensions vont permettre de prévoir un affichage correct suivant l'environnement de diffusion de la création.
- la couleur de l'arrière-plan. Il est à noter qu'il n'est pas possible de choisir un dégradé de couleur pour l'arrière-plan d'une animation *Flash*. En effet, les calculs nécessaires lors de l'affichage de ce type d'arrière-plan nécessiteraient des performances matérielles que tout utilisateur ne possède pas. Le poids du fichier serait aussi plus important; ce qui va à l'encontre du but premier de ce type de programme.
- la cadence de l'animation. Il s'agit de la vitesse de défilement des Images lors de la lecture de l'animation. Cette vitesse s'exprime en Images par seconde (ips). Par défaut, la cadence est réglée sur 12 ips. Cette vitesse est suffisance pour une animation provenant d'Internet. D'ailleurs, les autres types d'animations que l'on rencontre sur Internet (*AVI* et *QuickTime*) sont aussi cadencés à 12 ips. Le nombre d'images par seconde a une influence certaine sur la fluidité de l'animation. Il est vrai que les séquences vidéos classiques sont cadencées à une vitesse supérieure. D'ailleurs, pour que l'oeil fonde les images, la cadence minimum est de 15 ips mais dans ce cas, l'oeil n'a pas le temps de voir tous les détails des images et les objectifs d'une animation *Flash* sont différents de ceux d'une vidéo.

L'enregistrement et la publication des fichiers

Les fichiers créés sont enregistrés dans un format spécifique. Ils portent l'extension FLA. Ce sont les fichiers de travail. Il s'agit des fichiers d'animation à "l'état brut". Il est possible de les modifier à tout moment même après leur publication.

Afin de rendre ces fichiers lisibles sur Internet ou encore de pouvoir les lire sans posséder le programme de création, il faut publier ces fichiers. Plusieurs formats sont possibles:

- le format *SWF* (*ShockWave*[1] *Flash*) est le format obtenu lors de la publication du fichier *FLA*. Il sera lisible à partir du programme *FlashPlayer*. Il est également possible de lire ces fichiers *SWF* grâce au lecteur implémenté (il peut s'agir d'un plugin ou d'un Contrôle *ActiveX*) dans les navigateurs. Le fichier *SWF* doit alors être intégré dans une page *HTML*.
- le format *EXE*: en publiant le fichier *FLA* sous ce format, on rend le fichier exécutable, ce qui permet de créer des applications offline ne nécessitant aucun autre programme pour les lire. La taille de ce fichier est considérablement augmentée étant donné que le lecteur Flash

[1] Sh*ockWave* est une technologie développée par Macromedia Inc. destinée à permettre l'insertion d'objets multimédia dans des pages Web.

se trouve intégré dans le fichier.

- le format *HTML* permettra de contrôler la lecture du fichier *SWF*, son affichage et son aspect dans une page *HTML*.
- les formats *GIF*, *JPEG* et *PNG* permettent d'exporter les Images de l'animation dans ces différents formats.

Lors de la publication d'un fichier FLA dans les différents formats décrits ci-dessus, ces publications s'enregistrent par défaut, dans le même dossier que le fichier source FLA.

La scène et ses objets

Les objets de type dessin

Les outils de dessin

Les outils de dessin disponibles sont fort semblables aux outils (de dessin) de n'importe quel logiciel graphique qu'il soit d'ailleurs de type vectoriel ou de type bitmap. Comme chacun le sait, tous les programmes ont des invariants (outil de sélection, crayon, pot de peinture, ...) mais ont aussi leurs particularités. La découverte devra être progressive.

Cependant, pour une prise en main, un premier exercice serait de manipuler ces outils et de construire quelques graphiques pour se rendre compte des performances que l'on peut atteindre et sans doute aussi des limites. Dans ce cas, le talent de chacun sera mis à l'épreuve.

Les éléments graphiques d'une scène sont appelés objets. Il existe 3 principaux types d'objets:

- les Formes
- les groupes
- les symboles

Il existe une gradation dans l'utilisation de ces objets.

Les Formes

Les premiers graphiques que vous composez sur une scène sont essentiellement des formes (ovale, ligne, rectangle,...). En effet, un dessin, un schéma, un croquis, aussi compliqué soit-il, commence toujours par quelques traits ou quelques formes. C'est l'organisation de ces formes simples qui permettra la présentation de formes plus complexes. Mais partons de ces graphiques simples.

*Tout graphisme construit à partir des outils de dessin est considéré comme une **FORME**. La Forme est l'objet le plus élémentaire dans Flash.*

Il s'agira de ne pas confondre l'objet Forme (écrit avec une majuscule) avec le terme générique: forme.

Une première observation est à faire lorsqu'on commence à dessiner sur la scène en utilisant les outils: la Forme peut être de deux types: le **trait** ou le **remplissage**. Des outils de dessin correspondent à ces deux types de Forme. Ces 2 types de Forme ont un comportement différent et des attributions différentes.

Voici un tableau récapitulatif[2] des caractéristiques de ces 2 types de Forme:

[2] Extrait de *Flash MX Animation, interactivité, ActionScript*, Guylaine Monnier, Dunod, Paris, 2002

	Trait	Remplissage
Qui est quoi	La ligne Le crayon Le tracé de la plume L'extérieur des ovales L'extérieur des rectangles	Le pinceau L'intérieur des ovales L'intérieur des rectangles L'intérieur d'un tracé fermé à la plume Le texte séparé L'image bitmap séparée
Déformation Flèche	Ligne droite ou courbe	forme pleine ou déliée
Modification	Pipette Crayon + Encrier	Pipette Pinceau + Pot de peinture
Texture image bitmap	non	oui
Dégradés	non	oui
Flou extérieur	non	oui
Dilater/Rétracter	non	oui
Affecter un contour	non	oui

Remarque particulière: les Formes peuvent avoir des comportements vectoriels ou "bitmap".

Exemple: dessiner un rectangle rouge et un ovale bleu qui ne se chevauchent pas et qui n'ont pas de bord (partie trait). Sélectionnez l'ovale et déplacez-le sur le rectangle. Déplacez-le ensuite sur un autre endroit de la scène. Le rectangle n'a pas changé. Faites la même chose en sélectionnant le rectangle et en le déplaçant sur l'ovale. On observe que l'objet sélectionné vient toujours se positionner au-dessus des autres objets. (Il semble que les deux objets réagissent comme des objets vectoriels. Refaites maintenant une autre manipulation. Sélectionnez le rectangle et placez-le sur l'ovale. Dé-sélectionnez-le, ensuite, re-sélectionnez-le et déplacez-le. L'ovale est maintenant coupé et le morceau situé sous le rectangle à disparu. Il semble maintenant que les objets réagissent comme des objets "bitmap". Il faut en avoir conscience, dès la prise en main du logiciel, lorsqu'on dessine ces Formes car cela peut perturber quelque peu.

Vous pouvez même essayer avec deux Formes de même couleur, elles ne semblent pas être dans le même plan. Cependant si vous dé-sélectionnez la seconde Forme alors qu'elle se superpose sur la première, après, elles n'en forment plus qu'une.

Les Formes se mangent (quand elles sont de couleurs différentes) et s'agglomèrent (lorsqu'elles sont de même couleur).

Les groupes

Comme son nom l'indique, le groupe est un ensemble de Formes. Le groupe contient des Formes. Dès que les Formes sont englobées dans un groupe, elles ne sont plus modifiables.

- La constitution d'un groupe a plusieurs avantages:
- Les Formes sont protégées. Cela évite en cas de déplacement sur la scène par exemple, que certaines Formes ne se "mangent" accidentellement ou ne s'agglomèrent.
- Il y a augmentation de la productivité lorsqu'on travaille dans un groupe: Flash ne recalcule pas les éléments extérieurs au groupe à chaque manipulation.
- Le lecteur Flash n'affiche un groupe que lorsqu'il est complètement chargé. Il y a donc optimisation de l'affichage.
- Les groupes ne seront créés que si l'animation est compliquée. Une animation simple serait rendue plus lourde par la création d'un groupe.

Les symboles

Les symboles sont fort semblables aux groupes. Ils ont, en tout cas, les mêmes avantages. Ils ont en plus, la particularité d'être stockés dans une bibliothèque. Ils sont, de ce fait, réutilisables. Ils permettent donc de gagner du poids dans l'animation. Les symboles sont comparables à des "modèles" dans la bibliothèque, dont sont extraits des "tirages", en local sur la scène, et cela autant de fois que nécessaire.

> Chaque *utilisation* d'un symbole sur la scène est une *occurrence* du symbole.

Les symboles peuvent avoir 3 types de comportement:

- **le symbole graphique** est le plus simple et le plus élémentaire. Il contiendra de simples Formes. Il est utilisé pour gagner du poids dans l'animation. Il doit être créé chaque fois qu'un même dessin doit être répété sur une même image ou dans le temps. Par exemple: la roue d'une voiture.
- **le symbole bouton** est un symbole réactif. Lors de son survol par pointeur de la souris, ce dernier se transforme en main pour signaler que la zone est réactive. Des actions pourront donc être déclenchées en fonction du comportement de l'utilisateur sur ce bouton.
- **Le symbole clip d'animation** est un symbole interactif et programmable. Un symbole clip d'animation peut également être animé. Il possède sa propre temporalité ce qui lui permet de lire son contenu, et cela, indépendamment du scénario principal. Par exemple, une roue de voiture qui tourne sur elle-même.
  Le clip d'animation est la clé de l'interactivité. Celle-ci sera détaillée plus loin.

Certains symboles n'auront pas de contenu graphique sur la scène. En effet, un symbole peut ne contenir qu'un son.

Les objets de type texte

Outre le dessin vectoriel, il est bien entendu possible de présenter du texte sur une scène.

Il est possible de trouver 3 types de blocs de texte:

- **la zone de texte statique**. Il s'agit d'un contenu rédactionnel qui sera simplement affiché et lu par l'utilisateur.
- **le champ de texte dynamique**. Il s'agit d'une zone réservée à l'affichage de texte de façon dynamique. Un nom d'occurrence est associé à ce champ. Cela permet de le manipuler par programmation. Le contenu est affiché par l'intermédiaire d'une variable. Ce champ de texte dynamique est un *champ de lecture* pour l'utilisateur. Cela pourrait être, par exemple, un score qui s'affiche.
- **le champ de texte de saisie**. On peut le comparer à un "champ de formulaire". Un nom d'occurrence est également associé à ce champ. Il est donc aussi possible de le traiter par programmation. Une variable sera également affectée à ce champ afin d'en permettre le traitement. Ce champ de texte de saisie est un *champ d'écriture* pour l'utilisateur. Cela pourrait être, par exemple, l'introduction d'une réponse à une question.

Ces trois types d'objet ont des propriétés différentes. Ces propriétés peuvent être définies par l'intermédiaire de l'inspecteur de propriétés. C'est d'ailleurs par l'intermédiaire de cet inspecteur de propriétés qu'il faut préciser, au moment de la création de l'objet, le type de bloc texte (statique, dynamique ou de saisie) que l'on veut insérer dans la scène.

Outre les objets graphiques classiques (provenant de dessins vectoriels) et les objets de type texte, il est encore possible d'insérer dans une scène des objets multimédia tels que des sons, des clips vidéos, des images bitmap,... Ces objets doivent d'abord être importés dans la bibliothèque pour pouvoir être utilisables.

Le scénario et ses Images

Comme cela a déjà été écrit précédemment, c'est sur le scénario que la notion de temps apparaît.

> Le scénario permet de distribuer le contenu de l'animation dans le temps.

C'est donc la description de l'animation qui se trouve sur le panneau *scénario*. Il est cependant difficile de travailler uniquement sur le scénario sans tenir compte des objets situés sur chacune des scènes de ce scénario. Du moins lorsqu'il s'agit d'animations graphiques.

Les Images

Ce scénario est composé d'**Images** (écrit avec une majuscule! afin de ne pas les confondre avec les images graphiques). Elles représentent des éléments *non graphiques* mais des unités de temps. La lecture du scénario se fait à l'aide d'une tête de lecture que l'on peut déplacer manuellement ou faire déplacer automatiquement en utilisant le lecteur Flash.

Les Images du scénario peuvent avoir 3 états différents:

- Image-clé vide
- Image-clé
- Image virtuelle

A l'ouverture d'un nouveau fichier, il y a nécessairement, en première position sur le scénario, une **Image-clé vide**. Cette Image-clé vide peut être comparée à une boîte vide. C'est dans cette boîte que viendront se placer les éléments graphiques (un dessin vectoriel, une image bitmap, une vidéo,...) ou non graphiques (un son, une action, un commentaire,...) que l'on dépose sur la scène. Dès le moment où un élément est placé sur cette Image-clé vide, cette dernière se transforme en **Image-clé** (sous-entendu non vide).

On peut déjà se rendre compte de la corrélation qui existe entre l'Image-clé du scénario et la scène. Il faut à tout moment pouvoir observer les deux car lorsqu'on insère un objet non graphique (un son ou une action) sur la scène, on peut quand même le visualiser sur l'Image du scénario.

Un troisième type d'Image est l'**Image virtuelle**. Cette Image n'est pas une "boîte". Elle ne contient qu'une duplication du contenu de l'image précédente. Une Image virtuelle est un prolongement visuel d'une Image-clé. Ces Images virtuelles permettent donc de voir un élément plus longtemps. Elles remplacent avantageusement (gain de poids) des Images-clés successives ne comportant aucune modification de graphie. En bref, si l'on veut voir longtemps un objet, il faut insérer autant d'Images virtuelles que nécessaire.

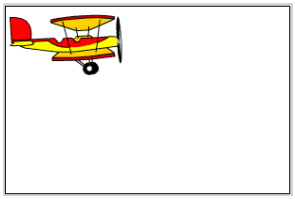
Il est impossible d'insérer un élément dans ce type d'image. Tout élément inséré sur ce type d'Image, le sera automatiquement dans l'Image-clé précédente.

Par contre les Images virtuelles ont toutes leur importance dans les animations et principalement dans les interpolations. Nous en reparlerons un peu plus loin.

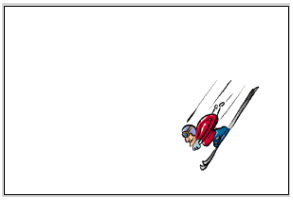
Pour pouvoir insérer du contenu dans une Image, il faut que celle-ci soit une Image-clé (vide ou non suivant les situations).

Les calques

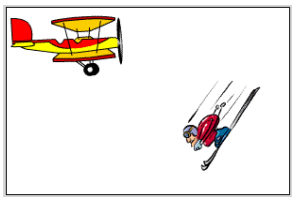
Les calques sont des supports destinés à recevoir des éléments (graphiques ou non). On peut les comparer à des transparents qui se superposent. Ils permettent ainsi d'organiser les objets selon des plans d'affichage différents. Ce système de superposition permet de gérer l'ordre des objets sur la scène.



Calque 1



Calque 2



Superposition des 2 calques

Le calque situé le plus en bas dans la fenêtre du scénario est situé le plus en arrière-plan. C'est sur ce calque que l'on placera les éléments destinés à constituer le *fond de l'animation*.

Il faut encore savoir que sur un même calque, les éléments déposés respectent la même règle d'empilement. Donc le dernier déposé sera situé en avant-plan par rapport à tous les autres. Cependant, il faut remarquer que les objets de type Forme sont systématiquement affectés au plan le plus en arrière du calque. Ce qui signifie que si, sur un même calque on dépose un groupe, puis un symbole puis une Forme, l'ordre de l'arrière vers l'avant sera la Forme, le groupe et en avant-plan le symbole.

Les calques sont indispensables dans la gestion des objets. Ces calques pourront même être organisés en dossiers et sous-dossiers, ce qui permettra une amélioration de l'organisation de l'animation. Il faut cependant savoir que ces dossiers n'ont aucun impact sur le déroulement de scénario. Ils n'ont pour fonction que de garder un espace de travail clair et propre. Il est fortement conseillé d'en user sans réserve. Surtout qu'ils ont la propriété avantageuse de ne pas augmenter le poids du fichier.

Une autre raison pour laquelle le calque devient indispensable dans la création d'animations un peu plus complexes est que pour pouvoir animer un objet, celui-ci doit être isolé sur un calque.

Comment s'organiser efficacement ?

Vous créerez systématiquement des calques différents pour isoler les informations graphiques des informations non graphiques. Ainsi des calques différents seront insérés pour:

- les actions
- les étiquettes
- les commentaires

- le son

En isolant ainsi les différentes informations, il sera beaucoup plus facile d'y avoir accès pour les manipuler.

D'une manière plus pratique, les calques pourront être verrouillés afin de ne pas les modifier malencontreusement par une manipulation involontaire. Ils pourront également être rendus invisibles lors de la création de nouveaux calques afin de faciliter le travail sur d'autres. Ces deux paramètres n'ont cependant aucune répercussion sur le travail final c'est-à-dire lors de la lecture du fichier.

Les calques peuvent être aussi de plusieurs types:

-le calque *Normal*. C'est le plus courant et c'est la propriété de tout calque par défaut.
- le calque *Guide*. Il pourra contenir une trajectoire qui servira de repère pour un objet en mouvement situé sur un autre calque bien entendu. Ce type de calque ne sera pas visible dans l'animation et il ne sera même pas exporté lors de la lecture du fichier.
- le calque *Guidé*. C'est celui sur lequel se trouvera l'objet qui doit suivre une trajectoire dessinée sur le calque précédent. Une opération ne sera possible que si le calque Guide est situé juste au dessus du calque Guidé.
- le calque *Masque*. Il permet de masquer le contenu d'un ou plusieurs calques *Masqués*. Sa place dans la hiérarchie des calques est tout aussi importante. Il doit se situer juste au-dessus d'eux.
- le calque *Masqué* contiendra forcément des informations qui ne seront visibles qu'au travers du calque *Masque*.

Ces différents types de calques pourront être classés dans des dossiers et sous-dossiers afin de rendre le travail organisé et structuré.

L'animation

Maintenant que nous connaissons les fonctions de base du logiciel, nous pouvons entamer la création proprement dite. N'oublions pas qu'il faut en même temps travailler sur la scène et sur le scénario donc sur les Images et sur les Calques.

Quelques propriétés sont à définir avant le commencer la création d'une animation:

- les dimensions de la scène: elles seront déterminées en fonction de la destination finale et surtout du type de résolution d'écran. Il est cependant possible, par programmation, d'adapter ces paramètres pour la configuration personnelle de chaque utilisateur.
- la couleur de la scène: il n'est pas possible d'utiliser des dégradés de couleur car c'est un élément qui sollicite énormément le CPU[3] de la machine et qui ralentirait la lecture de la séquence.
- l'unité de mesure prise en référence: pixels, centimètres, pouces,...
- la cadence de l'animation: c'est le nombre d'Images par seconde. Ce paramètre est défini de manière unique pour toute l'animation.

Il existe plusieurs méthodes de création d'animations. Le choix de l'une ou l'autre méthode va dépendre essentiellement du produit que vous voulez créer.

Les animations simples

L'animation Image par Image

Ce type d'animation ressemble très fort aux animations traditionnelles que l'on connaît déjà. Ne citons que les *gif animés*. Il s'agit dans ce cas de créer chacune des images individuellement et ensuite de les rassembler par l'intermédiaire du logiciel afin que celles-ci soient vues et lues de manière séquentielle.

Cette manière de créer une animation peut très vite devenir lourde en poids si l'on souhaite utiliser cette animation pour Internet. En effet, l'utilisation de ce procédé nécessite que chaque Image du scénario soit une Image-clé.

De plus, deux méthodes sont envisageables pour construire ce type de séquence.

- chaque Image est construite séparément en redessinant l'entièreté du contenu avec les quelques modifications nécessaires. Citons par exemple la marche d'un personnage.

- chaque Image contient un maximum d'objets communs. Si l'on reprend le cas précédent de la marche du personnage, le tronc et la tête de ce personnage sont toujours les mêmes. Il est possible de les placer sur un calque. Les parties qui diffèrent seront, quant à elles, placées sur un(des) autre(s) calque(s). Les modifications seront donc minimisées et par le fait même, le poids du fichier aussi. Cette méthode pourra encore être améliorée si les objets "en mouvement" sont

[3] CPU: ***C**entral **P**rocessing **U**nit*: microprocesseur.

transformés en symboles. Il ne restera plus qu'à placer ces symboles dans les positions convenables pour que la lecture puisse montrer un effet plus naturel. Pour aider le concepteur, un outil est mis à sa disposition: la *pelure d'oignon*[4].

Il s'agit d'un mode d'affichage qui permet de visualiser, sur la scène, le contenu des Images qui précédent et suivent l'Image en cours. L'utilisation de ces pelures d'oignon n'a aucune influence sur le contenu réel de l'animation. Elles ne sont pas, non plus, visibles lors de l'exportation du fichier. Elles n'influencent donc pas le poids du fichier. C'est un outil de visualisation présent dans le logiciel.

Les objets se trouvant sur chacune des Images de cette animation peuvent être aussi des symboles graphiques qui ont été préalablement importés dans la bibliothèque. Il peut s'agir d'images de formats différents (bmp, gif, jpg,...) Dans ce cas, le poids du fichier sera nettement plus important.

L'animation par interpolation

Les interpolations sont des effets générés par le logiciel. Elles permettent d'améliorer nettement le poids du fichier. En effet, le concepteur crée l'Image-clé de départ et l'Image-clé de fin et insère entre les deux autant d'Images virtuelles qu'il souhaite en fonction de la durée de l'animation qu'il a choisi et de la cadence donnée à cette animation. C'est le logiciel qui se charge de créer les Images virtuelles intermédiaires au moment de la lecture du fichier.

Il existe deux types d'interpolation:

L'interpolation de mouvement

Elle se construit, sur la 1ère Image-clé, à partir d'une Forme ou d'un groupe de Formes converti en symbole. Il s'agit bien entendu d'une occurrence de ce symbole.

La 2ème Image-clé contiendra une autre occurrence[5] de ce même symbole. Cette occurrence sera modifiée en position[6] mais aussi et/ou en grandeur et/ou en couleur.

Méthode pour créer une interpolation de mouvement simple.

- Dessiner la forme sur la scène (il est toujours sous-entendu que l'on ait choisi la bonne Image et le bon calque).
- Transformer cette forme en symbole graphique;
- Sur le même calque, insérer une nouvelle Image-clé vide.

[4] La pelure d'oignon est un mode d'affichage des objets sur la scène en environnement auteur. Elle n'a aucune incidence sur le résultat puisqu'il s'agit d'un outil de travail.

[5] Chaque occurrence d'un symbole peut porter un nom différent. Dans certaines situations, ces noms d'occurrences sont souhaitables. L'insertion du nom de l'occurrence se fait par l'intermédiaire du Panneau de Propriétés lorsque l'objet est sélectionné.

[6] pour donner l'effet de mouvement...

- Se positionner sur cette nouvelle Image-clé.
*- Insérer, sur la scène, une occurrence de ce même symbole (par un cliquer-glisser depuis la **bibliothèque** jusque sur la scène).*
- Se placer sur une image virtuelle intermédiaire dans le scénario.
*- Dans le menu **Insertion**, créer une **interpolation de mouvement** ou dans le panneau **Propriétés** de l'Image choisir **mouvement** dans **Interpolation**.*

Il n'est pas interdit de limiter l'interpolation de mouvement au seul déplacement. La seule contrainte est de travailler sur deux occurrences d'un même symbole. Chaque occurrence peut être modifiée en position, en grandeur, en couleur, ...

C'est avec ces interpolations de mouvement que l'on va utiliser les calques guides. L'interpolation sera, quant à elle, placée sur un calque guidé. Nous en parlerons un peu plus loin.

L'interpolation de Forme

Il s'agit dans ce cas, de partir d'une Forme se trouvant sur la 1ère Image-clé pour terminer par une autre Forme sur la dernière Image-clé. C'est donc la transformation d'une Forme en une autre Forme. On appelle généralement ce type de transformation: *morphing*.

Il est bien évident que la Forme de départ peut aussi servir de base de dessin à la Forme d'arrivée.

Certaines options peuvent encore être déterminées pour affiner les effets d'animation:

- l'accélération (positive ou négative) qui montre une modification plus importante (au début ou à la fin de l'animation).
- le mélange: distributif (pour des Formes arrondies) ou angulaire (pour des Formes plus anguleuses).
- les repères de Formes: permettent de guider et d'orienter les transformations. Ce paramètre est surtout utilisé pour les Formes complexes.

Méthode pour créer une interpolation de forme.

- Dessiner la Forme (simple ou complexe) sur la scène (l'Image-clé étant bien sélectionnée).
- Sur le même calque, insérer une nouvelle Image-clé vide.
- Sur cette nouvelle Image-clé, dessiner une autre Forme (qui peut aussi avoir une autre couleur).
- Se positionner sur une Image (virtuelle) intermédiaire.
- Modifier le type d'interpolation en interpolation de Forme.

Les Formes qui subissent une interpolation de forme ne doivent pas être groupées ni transformées en symboles. Cela n'aurait d'ailleurs pas de sens !

Lors de certaines interpolations de formes plus complexes, la transformation ne se fait pas toujours de manière harmonieuse. Dans ces cas compliqués, le logiciel fournit la possibilité de

placer des repères de formes qui permettront de contrôler les modifications. Ces repères identifieront les points qui doivent correspondre dans les Formes de début et de fin. Citons par exemples, la transformation d'une forme pleine en une forme creuse ou encore les yeux d'une même figure qui doivent rester en place lors d'une modification d'expression de ce visage.

Méthode: insertion de repères de forme dans une interpolation de forme

- Créer l'interpolation de forme comme cela est décrit précédemment
- Se placer sur l'Image-clé de début
*- Vérifier que l'option "**Afficher les repères de formes**" dans le menu **Affichage** est bien sélectionnée*
*- Par l'intermédiaire du menu **Modification - Forme** sélectionner l'option "Ajouter des repères de formes".*
- Placer le ou les repère(s) de forme(s)[7] sur les points "sensibles" de la forme.
- Se positionner sur l'Image-clé de fin et disposer aussi correctement les repères de formes.
- Pour se faciliter la tâche lors du placement de ces repères de formes, il est conseillé d'afficher les contours des pelures d'oignon.

Qu'il s'agisse d'une interpolation de forme ou de mouvement, la représentation de cette interpolation sur le scénario est visualisé par une flèche continue entre les deux Images-clés. Cette flèche est placée sur un fond vert (interpolation de forme) ou bleu (interpolation de mouvement). Si la flèche est représentée par une ligne discontinue, c'est que l'interpolation n'est pas correctement définie.

L'animation par programmation

La programmation peut aussi être utilisée pour créer des animations. On utilisera cette méthode dans le cas d'animations complexes. Pensons à certains effets comme l'élasticité ou la pesanteur qui nécessite des formules mathématiques. Celles-ci seront décrites avec le langage de programmation spécifique au logiciel: *ActionScript*.

Dans ce cas, l'animation sera encore plus légère qu'une animation par interpolation de mouvement.

Le choix entre ces trois techniques dépend avant tout du type d'effet que l'on veut mettre en oeuvre, de l'esthétique du résultat et surtout du niveau technique du concepteur. Le facteur temps de mise en oeuvre est aussi un facteur non négligeable. "Le jeu en vaut-il la chandelle?". L'utilisation de *l'ActionScript* sera décrite un chapitre spécifique.

[7] Il est possible de placer jusqu'à 26 repères de formes qui contiendront des lettres (de a à z) permettant d'identifier les points qui correspondent dans les formes de départ et de fin. Ces repères seront de couleur jaune dans l'Image-clé de début, de couleur verte dans l'Image-clé de fin et de couleur rouge s'ils ne se trouvent pas sur une courbe. Il est à remarquer que les repères de formes fonctionnent mieux s'ils sont placés en commençant par le point le plus en haut à gauche et en continuant le placement dans le sens horlogique.

Les animations complexes

Le travail du concepteur ne se limite pas à la création d'animations simples telles que celles décrites ci-dessus. Si l'on veut présenter un travail un peu plus sophistiqué, il va falloir réfléchir aux procédures à mettre en place. Il peut s'agir de combinaisons d'animations simples. Et là, les possibilités du logiciel sont nombreuses. On peut simplement travailler avec l'utilisation de calques différents ou de séquences différentes. Mais il peut aussi s'agir d'utiliser des symboles de types différents. Le plus courant pour créer une animation dans une animation sera le clip.

Toujours sur un seul calque...

Premier exemple: la décomposition d'une animation en plusieurs animations

Une animation peut devenir un peu plus complexe lorsqu'elle est décomposée en plusieurs animations simples. Pensez au va et vient d'un pendule.

Lors de la création d'une interpolation de mouvement, celle-ci peut être décomposée par la suite en une succession d'interpolations de mouvement par insertion d'Images-clés vides intermédiaires sur lesquelles on placera chaque fois une occurrence du symbole mais à des endroits différents ou avec des grandeurs différentes. Cette méthode pourra être comparée à une interpolation de mouvement guidée (voir exemple "le calque guide") où les trajectoires ne sont pas nécessairement rectilignes mais qui peuvent donner, par exemple, des effets dans la 3ème dimension par changement de taille des occurrences sans changer la position du symbole.

On se rend compte qu'il est possible de placer sur un même calque plusieurs interpolations de mouvement pour autant que celles-ci soient successives.

Deuxième exemple: animations successives de plusieurs objets

Des interpolations de mouvement pourront être créées sur des objets différents. Si l'on travaille sur un calque unique, les mouvements seront successifs: un seul mouvement d'objet à la fois sur un calque. Il n'est en effet, pas possible de superposer les interpolations de mouvement sur un même calque.

Sur plusieurs calques

Premier exemple: le calque Guide

Le calque *guide* nous permet de modifier la trajectoire rectiligne (définie comme telle dans le logiciel lors d'une interpolation de mouvement simple) en une trajectoire quelconque. Quelques consignes sont à respecter pour un bon fonctionnement de cette animation:

- le calque *guide* doit être situé au-dessus[8] du calque où se trouve l'objet à déplacer.
- Le calque contenant l'objet en mouvement doit être modifié en calque *guidé*. Il faut donc modifier les propriétés des deux calques considérés.
- Le centre[9] du symbole "objet" en mouvement doit se trouver sur la première Image-clé au

[8] ...dans le scénario...

[9] Le centre, appelé aussi *point d'alignement*, d'un symbole n'est pas nécessairement situé au centre de l'objet. Il est représenté par une petite croix lors de sa construction ou par

point de départ de la trajectoire et le centre de ce même symbole doit coïncider, sur la dernière Image-clé, avec l'extrémité de la trajectoire. Il faut, en quelque sorte que l'objet soit accroché à la trajectoire. Si après avoir testé l'animation, l'objet ne suit pas la trajectoire, c'est vraisemblablement que cet objet n'est pas correctement accroché au point de départ ou au point d'arrivée du guide.
- Suivant le type d'objet en mouvement sur cette trajectoire, il ne faudra pas oublier dans les propriétés de l'objet de sélectionner l'option "Orienter vers la trajectoire".

Méthode pour créer une interpolation de mouvement avec guide

- Réaliser une interpolation de mouvement classique (comme expliqué précédemment) sur un calque 1.
- Insérer un nouveau calque 2 juste au-dessus du calque contenant l'interpolation de mouvement.
- Dessiner une trajectoire quelconque sur ce nouveau calque en utilisant les outils de dessin. Il faut que cette trajectoire possède un point de départ et un point d'arrivée.[10]
- Sur la 1ère Image-clé du calque 1, positionner le symbole correctement (le centre du symbole doit coïncider avec le point de départ de la trajectoire et sur la dernière Image-clé du calque 1, le centre du symbole doit coïncider avec le point d'arrivée de la trajectoire.
- Transformer la propriété du calque 2 en calque guide.
- Transformer la propriété du calque 1 en calque guidé. Ces deux calques peuvent être renommés.
- Tester l'animation
*- Sélectionner éventuellement l'option "**Orienter vers la trajectoire**" sur le calque guidé.*

Remarque: les calques guides ne sont pas visibles lors de l'exportation de l'animation.

Deuxième exemple: deux objets en mouvement simultanément

Le travail créé pour une seule interpolation de mouvement peut se répéter aisément sur plusieurs calques. Cette superposition de calques permet des mouvements d'objets identiques ou différents dans des directions identiques ou différentes et à des vitesses identiques ou différentes. Les combinaisons sont donc multiples. Qu'il s'agisse d'interpolation de mouvement ou de forme.

Cette association de plusieurs calques permet de garder un arrière-plan commun pour toutes les images. Vous aurez peut-être déjà constaté que dès qu'une interpolation de mouvement est placée sur un calque, il devient impossible de placer d'autres objets (formes) sur ce même calque.

Donc, si l'on veut que deux objets se déplacent simultanément dans le temps, il est obligatoire de les placer sur deux calques différents. La complexité peut aussi augmenter si on donne à chaque mouvement un guide. Pensez, dans ce cas, qu'il faut aussi respecter les proportions de

un cercle contenant une croix lorsqu'une occurrence de ce symbole est placé sur la scène. La position de cette croix peut, à tout moment être modifiée à l'intérieur du symbole.

[10] Si la trajectoire est une courbe fermée, il faudra en "gommer" une petite partie pour avoir un point de départ et un point d'arrivée.

chaque objet pour que l'animation qui sera lue en une seule étape soit plausible. Vous pouvez dès à présent faire travailler votre imagination pour créer des animations.

Troisième exemple: des lettres en mouvement

Il ne faut pas s'imaginer, même si cela n'a pas encore été envisagé, que les animations sont réservées aux objets de type "dessin". Il est aussi possible de réaliser des animations d'objets textes. Dans ce cas, le texte inséré sur la scène sera un texte de type *statique*. Il sera converti, toujours selon les mêmes principes en symbole graphique. Et l'entièreté des caractères contenu dans ce symbole se déplacera sur la scène.

Si on souhaite un mouvement différent pour chacun des caractères de l'objet texte, une manipulation un peu plus complexe doit être mise en place. Etant donné que chacune des lettres effectuera un mouvement, ces lettres doivent être réparties sur des calques différents. Il ne faut pas perdre son temps à créer un nouveau calque pour chaque caractère. Cette opération peut être simplifiée. Suivez la méthode.

Méthode: répartition de textes sur plusieurs calques

- Sur un premier calque, placer autant d'objets texte qu'il y a de caractères dans le texte.
- Sélectionner tous les objets texte.
*- Utiliser la fonction "**Répartir vers les calques**".*
- Sur chacun des calques créés, réaliser une interpolation de mouvement avec le caractère présent.
- Ces interpolations peuvent débuter à des moments différents et être de durées différentes.

Les calques créés lors de ce type de manipulation peuvent être classés dans un seul dossier "calque de textes" par exemple.

Quatrième exemple: le calque Masque

Le *masque*[11] permet de voir certaines zones de l'animation et de ne pas en voir d'autres. Une zone de masque se définit sur un calque. Vous pouvez utiliser un calque de masque pour créer un "trou" qui laisse apparaître le contenu d'un ou plusieurs calques situés en-dessous. Ces calques peuvent contenir des objets statiques ou dynamiques. Citons comme exemple, le regard au travers d'un trou de serrure.

Mais vous pouvez également utiliser un masque de calque dynamique car un masque peut lui aussi être animé. L'interpolation de mouvement permet le déplacement de la zone de masque. Dans ce cas, on envisagera, par exemple, l'effet de projecteur sur une scène statique.

Suivant la forme qu'on place sur le masque de calque et l'animation qu'on lui applique, on peut créer des effets de projecteur, de loupe, pour donner l'impression aux spectateurs qu'ils

[11]Masque: calque contenant une ou plusieurs formes (statiques ou dynamiques) qui définiront les zones de transparence de l'Image. Ces formes doivent être de type remplissage et posséder une couleur.

regardent par un trou de serrure ou à l'aide de jumelles. L'imagination de chacun peut encore permettre d'inventer d'autres effets...

Méthode: masque statique devant une scène dynamique

- Sur un premier calque, créer une interpolation de mouvement d'un objet quelconque. Nommer ce calque "masqué" pour le reconnaître par la suite. Cette définition de nom n'est pas absolue. Le calque peut porter n'importe quel nom pourvu qu'il puisse être retrouvé par la suite.
- Sur un second calque, créer un masque, c'est-à-dire définir les zones de transparence de l'écran. Renommer ce calque en "masque". Ce n'est pas obligatoire non plus. Le nom peut être quelconque.
- Modifier la propriété du calque 2 (masque) en type Masque. Automatiquement le calque 1 devient un calque de type Masqué. Ces deux calques doivent être verrouillés pour que leurs nouvelles propriétés soient réellement actives.
- Lire la séquence.

Pour une plus grande visibilité du masque, vous pouvez créer un calque supplémentaire au dessus des précédents qui sera une réplique du masque mais dont les zones de forme seront complémentaires par rapport aux zones de forme du masque. Ces nouvelles zones pourront ainsi avoir une couleur.

Méthode: masque dynamique sur une scène statique

- Créer un calque 1 (masqué) avec des objets (images, dessins ou textes) statiques.
- Créer un calque 2 (masque) avec un objet (symbole) subissant une interpolation de mouvement.
- Sur le calque 1 du scénario, insérer une Image-clé à la même position que la dernière Image du calque 2.
- Transformer les propriétés des calques (masque et masqué) et vérifier que ces deux calques sont verrouillés.
- Lire la séquence.

Nous aurons encore l'occasion de revenir sur cette notion de masque lors de l'utilisation de clips. En effet, on peut aussi utiliser un clip animé pour créer un masque de calque dynamique.

Tous les calques se trouvant en-dessous d'un masque subissent les effets du masque, pour autant qu'ils aient la propriété d'être de type *masqué*. Les calques se trouvant par dessus le calque *masque* risquent, quant à eux, de cacher les effets d'un masque. Il faut donc être prudent avec ce genre de manipulation. Pour obtenir les effets escomptés dans une animation, il sera peut être nécessaire de les construire dans une séquence bien particulière.

La manipulation des masques, des guides, des interpolations va permettre une multitude d'effets spéciaux. Citons:
- un masque animé qui donne un effet de poursuite,
- une interpolation de forme sur le calque Masque,
- une transition qui laisse apparaître puis disparaître un objet et qui est provoquée par une

interpolation de forme ou de mouvement de l'objet sur le calque Masque,
- un effet de loupe sur un objet en utilisant un masque guidé sur une trajectoire,...

Les résultats obtenus dépendent beaucoup de l'imagination du concepteur et de l'organisation mise en oeuvre pour obtenir un résultat. C'est dans ce domaine précisément que chacun devra faire preuve de créativité et de réflexion.

Cinquième exemple: des calques contenant des objets non graphiques

Il n'y a pas que des objets graphiques que l'on peut placer sur les calques. Des objets de type son ont aussi leurs places dans les animations *Flash*. Ils ont cependant besoin d'être importés dans la bibliothèque de l'animation avant de pouvoir être activés. Ils doivent aussi se trouver sur des calques différents des calques contenant des graphies. Pour les placer sur ces calques, il suffit de les faire glisser de la bibliothèque vers la scène. Peu importe d'ailleurs l'endroit de la scène où vous les déposez car ils n'y seront pas visibles. La présence d'un son dans une animation ne se visualise que sur les calques du scénario.

Il est possible que la durée du son soit plus longue que celle de l'animation. La lecture du son se fera jusqu'au bout. Pour éviter ce désagrément et stopper la lecture du fichier audio, il faut introduire une commande *d'ActionScript* dans la dernière image du scénario. Nous envisagerons cette fonction dans un chapitre spécifique.

Des animations sur plusieurs séquences

Tout comme le monteur découpe son film lorsque toutes les prises de vue sont réalisées, le concepteur d'une animation va lui aussi découper son animation en plusieurs séquences. Cette méthode évitera de travailler sur un scénario trop long et trop encombré par les différents calques.

Chaque séquence pourra donc se construire séparément selon les procédures vues jusqu'à présent. Dès qu'une séquence est terminée, le concepteur créera la suivante. Lors de la visualisation de l'animation, le lecteur Flash Player, lira les séquences les unes à la suite des autres dans l'ordre déterminé par la création ou par le concepteur lui-même. En effet, il est possible, à tout moment de modifier l'ordre de lecture des séquences. C'est par l'intermédiaire du panneau "*Séquence*" qu'il est possible de modifier cet ordre. Rien ne vous empêche donc de créer des séquences dans le désordre puis les replacer chronologiquement par la suite.

Des symboles particuliers

Le clip

Le symbole clip est une animation à part entière. Pour construire un clip, il faut d'abord le décider avant de commencer. C'est donc de cette manière qu'il faut commencer la création d'une animation. La majorité des clips seront construits en priorité. Cela n'empêche pas le concepteur de créer de nouveaux clips par la suite. C'est donc aussi avec ce type de symbole qu'il faut envisager une animation d'une animation. Prenons comme exemple les roues d'une voiture qui avance. Il serait préférable que l'on puisse voir le mouvement de rotation de ces roues.

Quelle procédure mettre en place pour réaliser cette animation?

Il est vrai que l'on peut prendre ("perdre") du temps pour dessiner chacune des images en

modifiant la position de l'objet "voiture" ainsi qu'en modifiant le dessin de chacune des roues pour laisser apparaître leur mouvement de rotation. Mais le travail est fastidieux. En procédant de manière systématique, mais aussi pour réduire la taille du fichier, une seconde méthode sera beaucoup plus efficace. Il s'agit de créer un symbole de type clip représentant le mouvement d'une roue. Dans un deuxième temps, il faudra créer la "carrosserie" de la voiture et y ajouter les roues de la voiture. Le tout formera un symbole. Il sera de préférence aussi de type clip. C'est ce dernier symbole qui sera utilisé lors d'une interpolation de mouvement.

On est maintenant au coeur de l'animation. L'imbrication de clips est intéressante pour autant que l'on ait organisé son travail. Sans quoi on risque de se trouver confronté à des situations desquelles il est assez difficile de se sortir.

Il est assez fastidieux de transformer une animation simple en clip car il faut recopier chacune des images de l'animation dans le symbole. Il est donc préférable prévoir le travail à l'avance.

Les clips peuvent aussi être utilisé pour créer des masques. La mise en oeuvre est cependant un peu plus complexe.

Le bouton

Le bouton est un symbole très particulier qui peut être utilisé comme animation ou comme point de départ d'une interactivité. Nous le considérerons, dans ce chapitre, comme le point de départ d'une animation. C'est un symbole réactif. Il transforme le curseur en *main* lors du survol de cet objet par le pointeur.

Sa construction

Envisageons tout d'abord sa construction. Il est souhaitable de le définir comme tel avant de le dessiner. Ce symbole est composé de 4 images. Attention, il ne s'agit pas d'Images du scénario mais de 4 états différents ou 4 apparences. On parle aussi des images du scénario du bouton.

- l'état *Haut* (1ère image): représente l'apparence normale du bouton quand le pointeur n'est pas dessus,
- l'état *Dessus* (2ème image): représente l'apparence du bouton quand le pointeur se trouve dessus,
- l'état *Abaissé* (3ème image): représente l'apparence du bouton quand l'utilisateur clique dessus,
- l'état *Cliquable* (4ème image): définit la zone qui réagit au clic de la souris. Cette zone est invisible dans l'animation. Si cette zone n'est pas définie, elle est équivalente à la zone définie dans l'état *Abaissé*.

Les comportements de la souris

Chaque état d'un bouton correspond à son aspect selon le comportement de la souris. Les informations que l'on peut donner aux différents aspects sont variés:

- le son: il est possible d'associer un son à l'état *Abaissé* du bouton. Il ne faut pas, dans ce cas précis, ajouter de calque mais simplement faire glisser sur la scène un son que l'on a importé dans la bibliothèque.

- le texte: un bouton peut aussi être créé à partir d'un bloc de texte. Cette manipulation peut être utile dans le cas où lors d'un clic, on veut voir apparaître un autre bloc de texte.

Méthode: transformer un bloc de texte en bouton

- Sur la scène, insérer un bloc de texte.
- Transformer ce bloc de texte en symbole bouton.
- Accéder aux états du bouton (par double-clic sur le symbole).
- Dupliquer la zone de texte sur l'état Abaissé.
- Modifier le contenu de la zone de texte.
- Tester l'animation.

On observe que les zones entre les caractères ne sont pas cliquables. Pour remédier à cet inconvénient, on peut définir une zone cliquable artificielle sur le 4ème état du bouton. Cette zone (de forme rectangulaire) couvrira toute la zone de texte.

- un bouton invisible: si vous avez compris le principe des états du bouton, vous ne placerez une Image-clé que sur le 4ème état du bouton (état *Cliquable*).

Le texte

Dans cette première partie, l'animation de texte n'est envisagée que pour du texte de type statique. Ces zones de texte peuvent ainsi être traitées comme des objets graphiques. On peut donc leur faire subir aussi bien des interpolations de mouvement que des interpolations de forme. Cela a déjà été décrit précédemment.

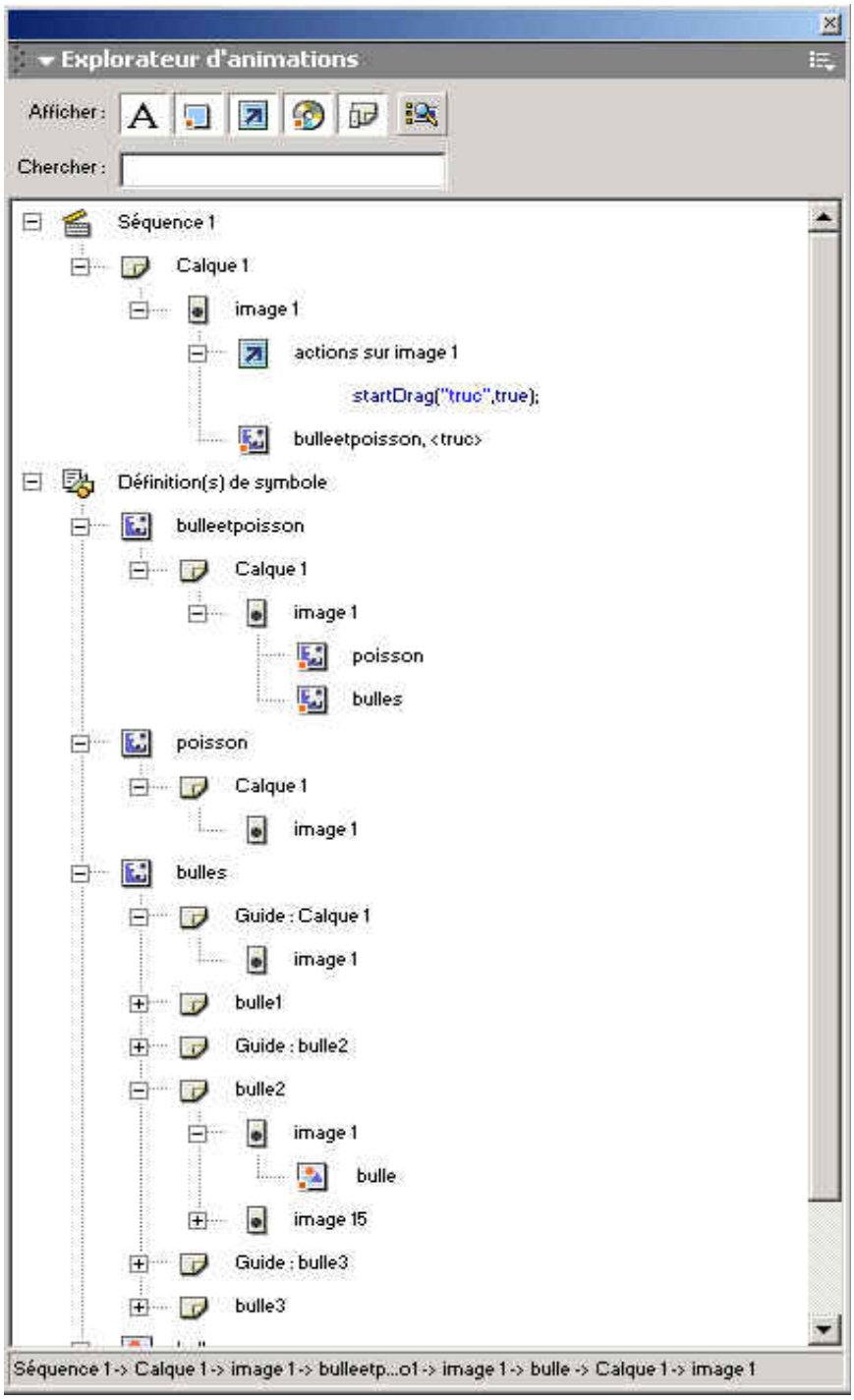
Les traitements les plus intéressants que l'on peut faire avec les zones de texte nécessitent l'utilisation de variables et du langage *ActionScript*. Ils seront donc décrits un peu plus loin.

L'organisation des informations

Il est intéressant, à certains moments, de pouvoir avoir une vue d'ensemble de son travail, qu'il s'agisse de visualiser les séquences, les calques, les clips,.... Une aide précieuse sera l'explorateur d'animations. Ce panneau peut être activé ou désactivé à tout moment. Il peut aussi être affiché de manière concise ou de manière détaillée. Il suffit pour cela de régler les paramètres d'affichage de ce panneau.

Attention cependant, toutes les informations ne seront pas données dans cet explorateur d'animations. Des renseignements tels que les propriétés de chacun des objets se trouvant sur chacune des images ne seront pas affichés. Chacun pourra trouver dans cet explorateur les renseignements qui lui conviennent pour visualiser ses animations.

Il est possible d'avoir accès directement à l'information que l'on recherche (en vue d'une modification) par l'intermédiaire de cet explorateur d'animations. Un double-clic sur l'objet le fait apparaître sur la scène. Il est donc aussi possible de travailler sur le scénario, la scène et sur une séquence par ce biais. Ce moyen permet d'avoir aussi un plan global de son travail. Cet explorateur peut se révéler être un outil de réflexion, de correction, de synthèse. Voici d'ailleurs un exemple :


Explorateur d'animations
Afficher :
Chercher :
Séquence 1
Calque 1
image 1
actions sur image 1
startDrag("truc",true);
bulleetpoisson, <truc>
Définition(s) de symbole
bulleetpoisson
Calque 1
image 1
poisson
bulles
poisson
Calque 1
image 1
bulles
Guide : Calque 1
image 1
bulle1
Guide : bulle2
bulle2
image 1
bulle
image 15
Guide : bulle3
bulle3
Séquence 1 -> Calque 1 -> image 1 -> bulleetp...o1 -> image 1 -> bulle -> Calque 1 -> image 1

ActionScript

Avant d'aller plus loin... un peu de POO

Sans entrer dans la rédaction d'un cours sur la programmation orientée objet[12] (désormais appelée *POO*), il faut quand même donner quelques notions de ce type de langage afin de pouvoir comprendre les mécanismes utilisés par *ActionScript* pour insérer de l'interactivité dans ses animations.

Dans la *POO*, les informations sont organisées en groupes appelés **classes**. Lors de la création d'une classe, il faut aussi définir l'ensemble des **propriétés** (caractéristiques) et des **méthodes** (comportements) d'un **objet** virtuel de cette classe. On pourra par la suite créer autant d'occurrences (objets) de cette classe que l'on veut. Tous les objets d'une même classe auront des caractéristiques communes et des méthodes identiques.

Il est à noter que *ActionScript* contient déjà une série de classes intégrées. On peut d'ailleurs les trouver dans le dossier *Objets* du panneau *Actions*.

Les **objets** peuvent être de simples containers (boîtes) de données ou peuvent être représentés graphiquement sur la scène sous forme de clips, de boutons ou de champs de texte.

Les clips sont des occurrences de la classe intégrée MovieClip. Chaque occurrence d'un clip contient toutes les propriétés et toutes les méthodes de la classe MovieClip.
Les boutons sont des occurrences de la classe intégrée Button.

En plus des classes qui existent dans *ActionScript*, il est possible de créer ses propres classes. Pour définir une classe, il faut utiliser une fonction spéciale appelée *fonction constructeur*.

Dans la *POO*, les classes peuvent s'échanger entre-elles des propriétés et des méthodes selon un ordre spécifique appelé **héritage**. Une classe qui hérite d'une autre classe est appelée sous-classe et une classe qui transmet à une autre classe est appelée super-classe. Une classe peut être à la fois sous-classe et super-classe.

Ces notions vont être décrites un peu plus en détail dans les pages suivantes.

ActionScript

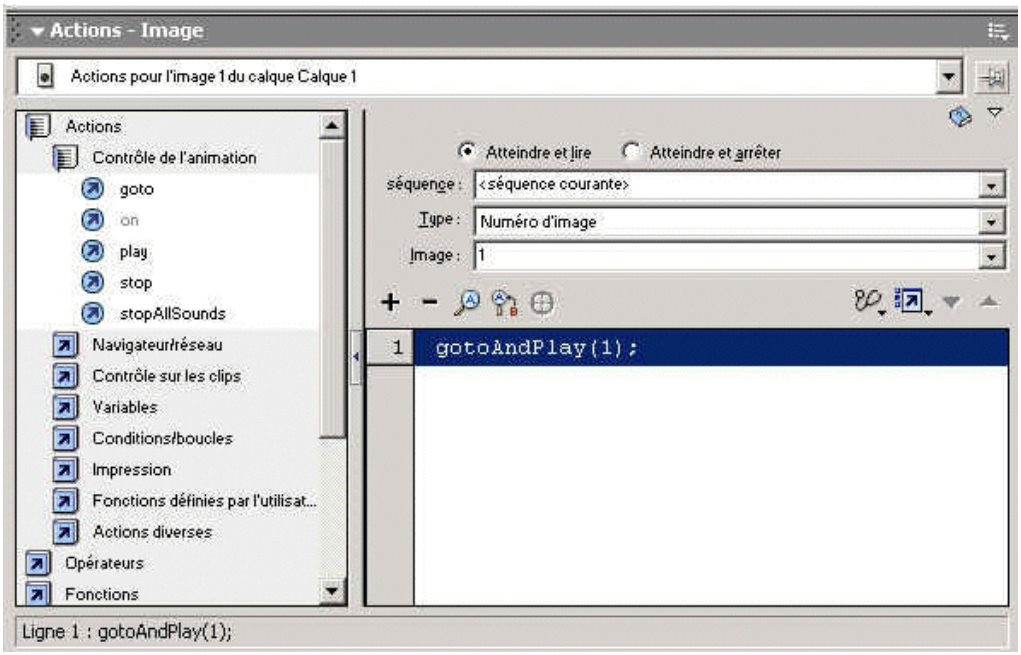
ActionScript, le langage de programmation propriétaire de *Flash*, permet d'ajouter de l'interaction dans une animation.

[12] Pour des informations sur la POO, consultez les notes "Programmer avec des objets: une découverte de la POO à travers le langage Java" d'Etienne Vandeput. Ce document est consultable à l'adresse:
http://www.det.fundp.ac.be/cefis/publications/etienne/poo2002.pdf

Pour introduire ces scripts[13] dans un fichier d'animation, trois méthodes sont mises à disposition:

- le mode Normal

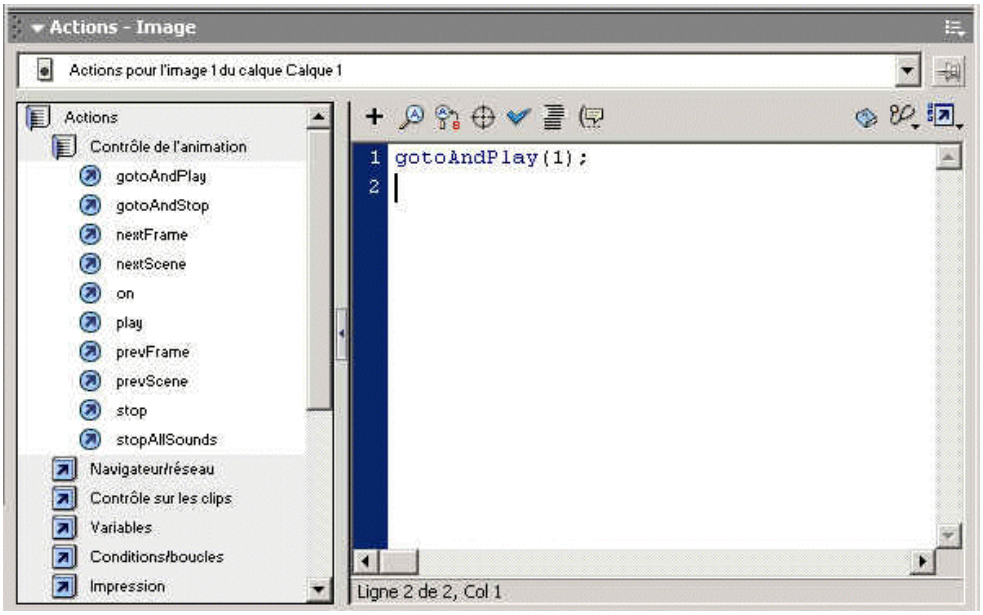
Ce mode permet de construire les scripts en sélectionnant les outils dans la boîte à outils **Actions**. Les actions, les fonctions, ... y sont prédéfinies. Il "reste" au concepteur à compléter les paramètres de chacune de ces actions, fonctions,... Ce mode peut être très avantageux pour les débutants mais ne permet pas toujours d'en comprendre le fonctionnement exact.



- le mode Expert

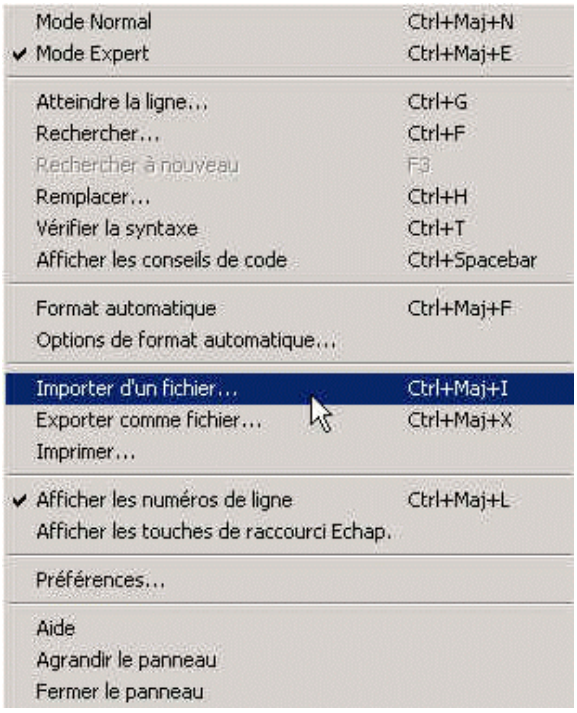
Le mode **Expert** nécessite que le concepteur saisisse, par l'intermédiaire du panneau **Actions** et dans la fenêtre réservée à cet effet, directement l'entièreté du code *ActionScript*. Les différentes options de cette fenêtre sont les mêmes que pour le mode **Normal**.

[13] Un script est un morceau de texte contenant les instructions d'un langage de programmation. C'est donc un programme dont le but est de réaliser une action: calculer, afficher un texte, lire une séquence,...



- par un éditeur externe

Il est aussi possible d'écrire les scripts à l'aide d'un éditeur de texte simple. Les fichiers doivent alors être enregistrés avec l'extension "*.as*" pour pouvoir ensuite être intégrés dans le fichier *FLA*. Ils sont importés dans la fenêtre d'action de *Flash* par l'intermédiaire du menu déroulant du panneau Actions (dans le coin supérieur droit du panneau).



Pour une meilleure organisation de votre travail et aussi par facilité, il est conseillé de créer un dossier **include** dans le dossier **configuration** de *Flash* et d'enregistrer vos fichiers "*.as*" dans ce dossier.

D'autres fenêtres sont également associées

à *ActionScript*:

- la fenêtre de **Sortie** qui permet, lors du test de l'animation, de lister les erreurs de syntaxe.

- la fenêtre de **Référence** qui donne la description de toutes les actions, fonctions, propriétés,... disponibles dans le langage.

- la fenêtre du **Débogueur** qui permet de rechercher les erreurs de script dans une animation en cours d'exécution dans le lecteur *Flash*. Ce débogueur permet d'insérer des points d'arrêt dans le code *ActionScript*. Ceux-ci permettront d'arrêter l'animation et de consulter le code ligne par ligne. La seconde étape sera le retour au script proprement dit et à sa correction.

Codes de couleur des scripts

Afin de faciliter la tâche du concepteur lors de la création de scripts et surtout lors de la recherche d'erreurs à l'intérieur de ces scripts, *Flash* a prévu des signes visuels pour l'aider à repérer ses erreurs.

Flash utilise des codes de couleur pour montrer le déroulement et la structure du script que le concepteur est en train de créer:

noir pour la structure générale du script. Celui-ci contient les signes, les chiffres, des noms d'occurrences, de variables, des paramètres,

bleu pour les instructions, les fonctions et les actions du langage. Les chaînes de caractères sont également colorées en bleu,

bleu marine pour les mots réservés. Si un mot réservé reste en noir, il faut en vérifier l'écriture et notamment la casse (majuscules et minuscules) car ils y sont sensibles,

gris pour les commentaires qui commencent par //. En mode Expert, des commentaires sur plusieurs lignes peuvent être encadrés par / * et */

rouge pour les syntaxes erronées ou incomplètes,

surlignage jaune pour les actions qui ne sont pas compatibles avec la version d'export choisie.

La syntaxe et la structure d'un script

La structure d'un langage est un mode d'organisation des instructions. Au sein de cette structure, il existe une syntaxe qui est au langage de programmation ce que la grammaire et l'orthographe sont aux langues. Cette syntaxe est nécessaire pour que tout le monde puisse se comprendre. Dans le cas de *Flash*, la syntaxe doit permettre à l'interpréteur du langage de comprendre les instructions que vous lui demandez d'exécuter.

Les accolades permettent de regrouper un bloc d'instructions. Elles déterminent la "portée" pour un script ou une instruction. Le début est précisé par l'accolade ouverte et la fin par l'accolade fermée.

Exemple:
```
on (release) {
        play ( );
}
```

Le point est utilisé pour indiquer les propriétés ou les méthodes associées à un objet ou à un clip. Il est aussi utilisé pour identifier le chemin cible d'un clip, d'une fonction, d'une variable ou d'un objet.

Le point-virgule signale la fin d'une instruction dans *ActionScript*.

Les parenthèses permettent de définir un ordre de priorité ou de définir des arguments[14] ou paramètres de certaines actions. Les parenthèses peuvent aussi servir à évaluer une expression située à gauche d'un point dans la syntaxe à point.

Les majuscules et les minuscules. Seuls les mots clés sont sensibles à la casse. Il est cependant préférable de s'en tenir aux conventions afin de retrouver facilement les noms des fonctions et des variables lors de la lecture du code *ActionScript*.

Les commentaires qui se représentent par // permettent d'insérer des remarques dans un script. Un script est plus facile à comprendre lorsqu'il est complété par des commentaires. Il est à remarquer que les commentaires n'alourdissent pas la taille du fichier exporté.

Les guillemets délimitent les chaînes de caractères.

Les objets

Comme cela a déjà été dit précédemment, les objets possèdent des propriétés et des méthodes. Comment reconnaître une propriété d'une méthode ? Voici un exemple de partie de script qui vous permettra de faire la distinction.

```
// positionnement du clip sur l'axe des abscisses à 100 pixels: (c'est une propriété)
monClip._x = 100;

// pilotage du clip: (c'est une méthode)
monClip.stop( );
```

Il s'agit de la position d'une occurrence du symbole clip qui a été renommée en monClip. Sa position est déterminée par la propriété _x qui détermine sa position sur l'axe des abscisses. Certaines propriétés peuvent être des constantes. Par exemple, l'une des propriétés de l'objet Math est PI[15].

> Les propriétés ne possèdent jamais de parenthèses. Elles ne s'exécutent pas mais se manipulent. Dans beaucoup de cas, il est possible de récupérer la valeur d'une propriété ou de la définir.

Dans le deuxième cas, le comportement du clip est modifié pour être arrêté. La fonction stop ()

[14] Argument: nombre ou chaîne de caractères qui suit une commande et qui lui indique comment elle doit travailler ou sur quoi elle doit s'appliquer. Il peut aussi s'agir d'une variable indépendante associée à une procédure ou à une fonction. Grand dictionnaire terminologique

[15] Ces constantes sont toujours écrites en majuscules.

pourrait aussi être utilisée comme action sur une Image[16].

> Les fonctions, actions et méthodes se terminent par des parenthèses, qui contiennent ou non des paramètres, et qui permettent leur exécution.

Les principales instructions du langage

Ces instructions peuvent être regroupées en 4 grandes familles.

Les objets

Ce sont des groupes de propriétés et de méthodes. Chaque objet ayant son propre nom et étant une occurrence d'une classe particulière. Il existe des objets prédéfinis dans *ActionScript*.

Exemple: Date : qui fournit des informations provenant de l'horloge système.

Les actions

Les actions sont des instructions qui demandent à une animation de faire quelque chose au cours de la lecture. On peut également utiliser le terme *instruction* pour parler d'une action.

Exemple: stop( );

Les fonctions

Ce sont des blocs de code réutilisables qui peuvent recevoir des paramètres et renvoyer une valeur.

Exemple: getVersion( ) : qui renvoie la version de Flash Player qui est en train de lire l'animation.

Les éléments et les composants d'interface

Un élément d'interface est à la fois un objet (un bouton), une action (en cliquant sur le bouton, le logiciel change d'image) et un événement.

Outre ces familles d'instructions, on peut encore retrouver d'autres types d'instructions parmi lesquels:

Les constantes

Les constantes sont des éléments qui ne changent pas. Elles sont utiles pour comparer des valeurs. Ces constantes sont toujours écrites en majuscules.

Exemple: Key.TAB : qui fait référence à la touche de tabulation.

[16] Dans le chapitre suivant, il sera question d'action *stop( )*. De manière générale, lorsque le choix existe, il est préférable dans la programmation objet d'utiliser la méthode plutôt que l'action.

Les propriétés

Elles sont des attributs qui définissent un objet de type clip. Leur nom commence toujours par un "underscore" (_).

Exemple: _totalFrames : renvoie le nombre total d'Images contenues dans le clip spécifié.

Les variables

Ce sont des identifiants qui contiennent des valeurs de n'importe quel type de données. Elles peuvent être créées, modifiées ou mises à jour. On peut récupérer les valeurs qu'elles contiennent et les utiliser dans les scripts.

Exemples: x = 5 ou nom = "flash"

Les opérateurs

Ce sont des instructions dédiées à la gestion des conditions, utilisées le plus souvent pour comparer des variables.

Exemple: Si a = b ou c >= 200 où a, b, et c sont des variables.

Toutes ces instructions sont détaillées dans l'aide du programme ou même dans la fenêtre Référence qui peut être activée à tout moment dans le logiciel.

Les événements

Toutes les actions, fonctions, ... sont décrites dans les scripts qui ne sont, en fait, que des blocs de texte. Ces scripts doivent encore être activés pour pouvoir modifier la lecture de l'animation.

Toutes les instructions placées dans ces scripts doivent être liées à des événements afin d'être exécutées à un moment donné lors de la lecture de l'animation. Il faut donc associer ces scripts à des contextes (événements). Ces contextes peuvent être:

- une Image du scénario principal,
- une Image d'un clip,
- un bouton contenu dans la scène, ou plus précisément un état de ce bouton,
- un clip contenu dans la scène.

Avec ce minimum de connaissances, nous allons pouvoir débuter la partie la plus importante qui est la conception de l'interactivité dans une animation.

Les informations supplémentaires qui seront nécessaires lors de la réalisation d'interactivités plus spécifiques seront détaillées au moment opportun ou nécessiteront la consultation de l'aide du logiciel. Quoi qu'il en soit, c'est surtout par vérifications successives que l'on pourra tester la cohérence de son travail. Il ne faut pas hésiter à réaliser un test à chaque modification de script.

L'interactivité

Qu'est-ce que l'interactivité ?

Dans une animation simple, la lecture des séquences et des images se fait de manière linéaire. Les animations *Flash* envisagées jusqu'à présent sont des produits (simples ou complexes) qui ne nécessitent aucune action de la part de l'utilisateur. Ce dernier est passif. Il ne doit que lire (ou regarder) l'animation. Il ne peut pas interagir avec le produit. Il s'agit en quelque sorte de la lecture ou la visualisation d'un "film".

L'introduction d'interactivité va, par contre, permettre à l'utilisateur de diriger la manoeuvre. Soit de lire tout ou seulement une partie du produit, un peu comme on peut lire un chapitre d'un film sur un DVD. La partie non interactive du produit est alors lue de manière séquentielle.

Une lecture linéaire ou non va dépendre des choix du lecteur, de l'utilisateur. Dans ce cas, cette lecture s'apparente plus de la navigation sur *Internet*.

Enfin, le produit peut être complètement interactif, alors le lecteur devient acteur de son parcours. Cette fois, les animations (telles qu'elles sont envisagées jusqu'ici) ne sont plus que des "prétextes" mais d'autres animations se mettent en place. Les animations deviennent des actions entre le produit et l'utilisateur.

La planification de l'interactivité

Avant de commencer à créer des animations interactives, il est indispensable de formuler les objectifs que vous voulez atteindre. Cette étape est certainement aussi importante, si pas plus, que l'élaboration d'un storyboard[17]. Il faut être conscient de ce que l'utilisateur va devoir faire pour utiliser correctement le produit. Citons quelques exemples:

- les utilisateurs devront entrer leur nom qui sera utilisé dans les messages,
- les différentes occurrences d'un bouton vont envoyer vers des parties différentes de l'animation,
- une des séquences présente un formulaire à compléter, ...

Dès que les actions sont définies, on peut utiliser le langage de programmation du logiciel pour mettre en pratique. Il sera certainement nécessaire de réajuster plusieurs fois ces commandes pour obtenir un résultat optimal. Des tests pourront être réalisés à tout moment lors de la construction d'une séquence ou d'une animation.

Les règles de base de l'interactivité

Lors de la création d'une interactivité, il faut tenir compte:

- de l'événement qui va entraîner une modification de la lecture de l'animation. Ces

[17] Storyboard: carnet de bord détaillant les différentes étapes de l'animation.

événements dépendent du comportement de l'utilisateur face à l'animation mais plus spécifiquement des objets qui vont se présenter à lui et des manipulations qu'il va réaliser. Ces événements peuvent être:

- la position du curseur sur la scène,
- le comportement de la souris sur certains objets,
- l'utilisation des touches du clavier,
- le temps qui passe.

- des objets de l'animation pouvant détecter ces événements. Il existe trois éléments réactifs sur lesquels le concepteur peut agir: l'Image, le bouton et le clip.
- de la cible, c'est-à-dire l'effet qui sera produit lors de l'activation de l'événement.

Ces différentes étapes de la création d'une interactivité vont être détaillées ci-dessous.

La complexité de l'interactivité

L'interactivité peut aller du plus simple au plus complexe. Cette interactivité est créée grâce au langage de programmation intégré du logiciel. Ce langage de programmation s'appelle *ActionScript*. Il s'agit de scripts décrivant les actions que le logiciel doit réaliser lors de l'intervention de l'utilisateur. Il n'est pas absolument nécessaire de connaître toutes les commandes disponibles dans le langage *ActionScript* pour commencer à insérer de l'interactivité dans une animation. Ces actions peuvent encore être découvertes au fur et à mesure des besoins. L'important est de maîtriser les concepts de base avant d'envisager tout contenu interactif.

Et le premier concept à intégrer est qu'une action doit toujours être affectée à un élément de l'animation.

Comment le logiciel peut-il interpréter l'interactivité ?

La notion espace/temps qui est présente en permanence dans la création d'une animation est aussi présente dans les actions. Certaines de ces actions sont affectées sur les Images du scénario et certaines actions sont affectées sur des objets de la scène. Il existe, en fait, 3 éléments réactifs pouvant détecter les comportements de l'utilisateur:

- un instant donné: l'événement est le temps. L'action se lit lorsque la tête de lecture passe sur l'Image où est placée l'action,
- un objet réactif: le symbole bouton qui détecte le comportement de la souris,
- un objet interactif: le symbole clip d'animation qui détecte les comportements et les événements sur l'animation.

	actions sur objets	**actions d'image**
notion	espace	temps

question	qu'est-ce qui déclenche l'action ?		Quand se passe l'action ?
réponse	un bouton	un clip d'animation	à un moment donné

Détaillons un peu plus ces différents types d'action. Afin de mieux les comprendre, il peut être utile également de consulter les notes se trouvant dans le chapitre précédent qui traite plus spécifiquement *d'ActionScript*.

Les actions de base de navigation

Les actions de base qui sont possibles permettent de contrôler la tête de lecture dans le scénario et d'éventuellement charger une page web dans une fenêtre de navigateur.

Les actions sur les Images

Généralement ces actions sont toutes regroupées sur un même calque placé au-dessus dans le scénario et nommé "Actions". Ces paramètres ne sont pas obligatoires mais permettent une plus grande clarté et une meilleure lisibilité. Mais il est également possible de placer une action sur une Image-clé précise d'une animation. Ce sera plus particulièrement le cas lorsqu'on travaille avec plusieurs séquences dans une animation et lorsque l'utilisateur passe d'une séquence à l'autre suivant les événements qu'il provoque (consciemment ou inconsciemment d'ailleurs).

Les actions sur les boutons

Les actions doivent toujours être placées sur une occurrence du bouton et non pas sur le symbole lui-même. En effet, un symbole de type "bouton" doit pouvoir être réutilisé plusieurs fois sans pour autant avoir toujours le même comportement. Il faut également imbriquer ces actions dans un gestionnaire[18] **on** et spécifier les événements souris ou clavier qui déclenchent l'action.

Les actions sur les clips

Les procédures d'utilisation des actions de navigation sont les mêmes que celles utilisées sur les boutons. A savoir, les actions sont placées sur une occurrence de clip et sont imbriquées dans un gestionnaire **onClipEvent**.

Quelles sont ces actions ?

Play et Stop

Syntaxe: play ( );

[18] Un gestionnaire constitue l'ensemble des lignes liées à un ou plusieurs événements. Un *gestionnaire* comprend la définition de l'événement et toutes les instructions liées à celui-ci. La littérature parle d'ailleurs de gestionnaire d'événements. Dans *ActionScript*, les actions spéciales associées à un bouton ou à un clip sont des *gestionnaires*. C'est en quelque sorte l'association entre une action et un bouton ou entre une action et un clip.

L'action *play* lit le scénario à partir de l'Image courante.

Syntaxe: stop( );

L'action *stop* arrête la tête de lecture sur l'Image en cours sur le scénario.

Les actions *play* et *stop* sont surtout utilisées en début ou fin de séquence.

Des parenthèses sont placées à la fin des actions, des fonctions et des méthodes. Elles permettent de les exécuter en leur attribuant des paramètres ou arguments. Des actions simples comme play et stop ne nécessitent pas de paramètres. Il est cependant nécessaire d'écrire les parenthèses mais elles restent vides.

Etiquette

Avant de passer à l'action suivante, il est intéressant maintenant de parler d'étiquettes. Les étiquettes sont des balises du temps. Il est parfois préférable de faire référence à un nom plutôt qu'à un numéro d'Image[19]. Cette étiquette sera insérée par l'intermédiaire du panneau des Propriétés. Elle sera valable pour tous les calques d'un même scénario. L'étiquette est représentée par un petit drapeau rouge sur le scénario. Le nom sera visible si des Images virtuelles le suivent.

Pour une question de clarté, il est préférable d'isoler les étiquettes sur un calque réservé à cet effet. On peut aussi constater que l'utilisation de ces étiquettes facilite la lecture dans une animation contenant des actions faisant référence à des Images dans d'autres séquences. Cela évite toute confusion.

Goto

Cette action va permettre de lire l'animation de manière non linéaire. Elle permet à la tête de lecture de passer d'une Image à une autre Image non consécutive ou à une Image précise se trouvant dans une autre séquence. Cette action est surtout utilisée conjointement au clic d'un bouton. Plusieurs syntaxes sont possibles:

- *Syntaxe:* gotoAndPlay (séquence, image);

  envoie la tête de lecture à l'image spécifiée d'une séquence et lit à partir de cette image. Si aucune séquence n'est spécifiée, il s'agit de la séquence courante. Le nom de la séquence doit être écrit entre guillemets. Le numéro de l'image peut être remplacé par son nom d'étiquette (entre guillemets aussi).

- *Syntaxe*: gotoAndStop (séquence, image);

  envoie la tête de lecture à l'image spécifiée d'une séquence et l'arrête.

- *Syntaxe*: nextFrame ( );

  envoie la tête de lecture à l'image suivante et l'arrête. Il n'y a pas de paramètre pour cette action.

[19] Surtout si on insère une Image dans le scénario après avoir placé cette étiquette.

- *Syntaxe*: PrevFrame ( );

  envoie la tête de lecture à l'image précédente et l'arrête. Cette action ainsi que la précédente sont surtout utilisée dans les actions sur les boutons et sur les clips.

- *Syntaxe*: nextScene ( );

  la tête de lecture va à la séquence suivante et s'arrête sur la première Image de cette séquence. Il n'y a aucun paramètre affecté à cette action

- *Syntaxe*: prevScene ( );

  la tête de lecture va à la séquence précédente et s'arrête sur la première Image. Il n'y a aucun paramètre affecté à cette action.

Une dernière action peut aussi être placée dans ce contexte. Il s'agit de charger un document depuis une adresse URL.

Syntaxe: getURL (url [, fenêtre [, "variable"]])

cette action permet de charger un document depuis une URL (relative ou absolue) dans une fenêtre spécifique ou suivant les propriétés que vous déterminez. Ces propriétés sont facultatives.

Tableau synthèse de la syntaxe

sur une Image	sur un bouton	sur un clip

ACTION	on (Evénement souris) {
	ACTION
	}
		onClipEvent (Evénement de l'animation){
		ACTION
		}

Les différents événements possibles sur la souris et sur l'animation peuvent être consultés à partir du dictionnaire ActionScript présent dans l'aide du logiciel ou à partir du panneau ***Référence****.*

Des interactivités complexes

Outre les interactivités de navigation de base qui viennent d'être envisagées, *Flash* permet des interactions beaucoup plus complexes. L'imagination de chacun peut tester les possibilités du logiciel mais certaines sont beaucoup plus courantes que d'autres. Nous allons en envisager quelques-unes parmi les plus utilisées.

Pour comprendre le fonctionnement de ces interactivités il faudra également consulter les notes sur *l'ActionScript* sans lequel rien ne sera possible. Il faudra plus spécifiquement comprendre les principes de base de la programmation orientée objet (*POO*). Il n'est cependant pas nécessaire de connaître tout le langage *ActionScript* pour débuter dans la création de ces interactions. La lecture du dictionnaire *ActionScript* dans l'aide ou l'ouverture de la fenêtre

Référence directement dans le logiciel sera souvent nécessaire pour trouver la bonne instruction et la bonne syntaxe. N'hésitez pas à vous en servir.

Les exemples ci-dessous permettront, sans doute, une meilleure compréhension du fonctionnement de cette interactivité.

Premier exemple: les mouvements d'un clip contrôlés par un script

Si les mouvements des objets (clips ou graphiques) peuvent être déterminés sur une animation par une interpolation de mouvement[20], il est également possible de contrôler les mouvements d'un clip par l'intermédiaire d'un script.

Ce sera à vous de choisir la méthode qui convient le mieux en fonction des besoins. Il ne faut pas choisir la solution de facilité mais analyser les contraintes de l'animation que vous voulez réaliser afin d'obtenir un résultat optimal et une taille de fichier minimale. Et cela, surtout, si l'animation est destinée à être diffusée sur Internet.

Ces mouvements peuvent être déclenchés automatiquement dès le démarrage de l'animation mais ils peuvent aussi être activés par l'intermédiaire d'un bouton.

Pour utiliser le langage *ActionScript* dans la construction de mouvements d'objets, il faut avant tout transformer ces objets en clips. Ceux-ci peuvent être animés ou non. La décision de prendre un symbole de type clip plutôt que de type graphique pour un symbole non animé va offrir un plus grand choix d'actions (méthodes ou propriétés).

Détail des étapes de la réalisation

Avant de se lancer dans la rédaction du script, il est indispensable de savoir le type de mouvement que l'on veut mettre en place ainsi que les événements qui vont s'y rapporter. Quelques questions doivent donc être posées: l'objet doit-il être en mouvement au lancement de la lecture de l'animation? Faut-il déclencher l'animation par un bouton? Le mouvement est-il rectiligne? Dans quel sens? Quelle est sa durée?...

On peut imaginer que suivant les questions posées, la mise en place du script sera différente. Il y a cependant des invariants dans cette animation. En effet, quel que soit le mouvement que l'on va définir, il faut avant tout:

- transformer l'objet à mettre en mouvement en clip. A moins que vous ne décidiez que ce clip soit animé. Au quel cas, il faudra le définir avant de le construire,
- placer une occurrence de ce symbole sur la scène (sur la première Image du calque 1 du scénario principal). Il est presqu'impératif de donner un nom à cette occurrence[21]. Le script sera en effet appliqué à l'occurrence du symbole et non pas au symbole lui-même. Le symbole (se trouvant dans la bibliothèque) pourra être réutilisé par la suite pour subir une autre action.

[20] Voir à ce propos le chapitre sur l'animation.

[21] Surtout si le symbole est réutilisé plusieurs fois. C'est le cas notamment des boutons.

Une fois ce premier travail réalisé on peut maintenant se focaliser sur la rédaction du script.

Un premier cas

Analyse

Le contenu du script doit correspondre au mouvement que l'on voudrait voir sur l'animation lors de la lecture. La principale différence avec une interpolation de mouvement classique est qu'il **ne faut pas** insérer des Images-clés sur le scénario. Il faut préciser dans le script que c'est au logiciel de le faire lui-même[22].

Le clip étant positionné avec précision sur la scène, il faut maintenant préciser que le logiciel[23] doit créer lui-même les Images-clés et re-positionner le clip sur chacune de ces nouvelles Images.

Script

```
1 onClipEvent (enterFrame) {
2      _x += 4;
3 }
```

Quelques remarques concernant ce premier script:

- **enterFrame** est une méthode de l'objet clip[24]
- **_x** exprime la coordonnée (en pixels) en abscisse du centre de l'occurrence par rapport au centre de l'objet dans lequel il est imbriqué.
- un déplacement vertical s'effectuerait avec la propriété **_y**. La combinaison des deux propriétés donnerait un déplacement oblique.
- le déplacement du clip s'effectue par pas de 4 pixels dans le sens horizontal de la gauche vers la droite. Le point (0,0) étant situé dans le coin supérieur gauche de la fenêtre du lecteur *Flash*.
- l'action est déclenchée à la cadence de l'animation que vous avez définie à l'ouverture du fichier.

L'exécution de cette animation n'a pas de limite. En effet, l'objet clip va dépasser les limites de la fenêtre de lecture de l'animation et un contrôle sur cet objet n'est plus possible, contrairement à l'animation créée par interpolation de mouvement.

Un second cas

---

[22] On peut déjà penser que le poids du fichier risque d'être plus petit étant donné que l'on insère aucune autre Image-clé dans le scénario mais simplement quelques instructions de type "texte"). L'intérêt sera sans doute plus grand lorsque le mouvement sera varié.

[23] C'est du lecteur *Flash Player* dont il s'agit puisque c'est au moment de la lecture du script que le travail se fera.

[24] Consultez les notes concernant l'utilisation de l'objet *MovieClip* en fin de document

Il serait intéressant d'arrêter l'animation du clip dès que celui-ci a atteint sa position finale. Cette position devant être déterminée par le concepteur bien entendu.

Dans une interpolation de mouvement, l'action est simple: l'insertion d'une action stop ( ) sur la dernière Image-clé et le problème est réglé. En effet, la tête de lecture s'arrête au lieu de continuer la lecture en boucle. Qu'en est-il dans un mouvement créé par un script?

Il faut préciser dans le script que le logiciel doit arrêter d'insérer des Images sur lesquelles il placera chaque fois le clip. Et cela, lorsque la position du clip aura atteint la limite de la fenêtre du lecteur *Flash* par exemple.

Il y aura donc deux étapes dans le script:

- tester la position du clip dans la fenêtre et si cette position est correcte,
- ajouter une Image-clé et insérer le clip dans cette nouvelle Image.

Script

```
onClipEvent  (load) {
    position = 1;
}
onClipEvent  (enterFrame) {
    _x += 4 * position;
    if (_x >550) {
        position = 0;
    }
}
```

Observations:

- position est un "drapeau" permettant d'effectuer un test à chaque insertion d'image. La valeur 1 est vraie tandis que la valeur 0 est fausse.
- la condition est testée après chaque insertion d'image.

Et si on veut que l'objet ait un mouvement de va et vient, il faudra encore compléter ce script pour préciser que lorsque l'extrémité est atteinte, le mouvement du clip est inversé.

C'est dans ce dernier cas que l'on constate qu'il est plus simple d'utiliser un script plutôt que des interpolations de mouvement. Avec un script, le mouvement est perpétuel si on ne l'arrête pas.

Voici encore quelques scripts de mouvements un peu plus complexes

Script 1: mouvement de va et vient sur une ligne horizontale

```
onClipEvent (load){
            sens = 1;
            }
onClipEvent (enterFrame){
    _x += 3*sens;
```

```
        if (_x>400){
            sens = -1;
        }
        if (_x<50){
            sens = 1;
        }
    }
```

Script 2: mouvement de va et vient sur une ligne horizontal avec rotation de l'objet lors du changement de sens

```
    onClipEvent (load){
        sens = 1;
    }
    onClipEvent (enterFrame){
        _x += 3*sens;
        if (_x>400){
            sens = -1;
            _rotation = 180;
        }
        if (_x<50){
            sens = 1;
            _rotation = 0;
        }
    }
```

Script 3: mouvement dans une direction avec accélération

```
    onClipEvent (load){
        acc = 0;
    }
    onClipEvent (enterFrame){
        acc += 0.1;
        _x += coef;
    }
```

Script 4: un mouvement circulaire

```
    onClipEvent (enterFrame){
        t += 0.1;
        _x = 50+Math.cos(t)*20;
```

```
    _y = 50+Math.sin(t)*20;
}
```

Script 5 : un mouvement elliptique

```
onClipEvent (enterFrame){
    t += 0.1;
    _x = 50+Math.cos(t)*40;
    _y = 120+Math.sin(t)*10;
}
```

*Un avantage non négligeable de l'utilisation de scripts pour réaliser des mouvements d'objets clips est que ces clips peuvent **tous** se trouver sur le même calque.*

Il est même possible de n'écrire qu'un seul script dans lequel on fait référence aux différents objets clips placés sur la scène. Il faut dans ce cas que tous les clips soient sur un même calque. Le scénario est alors réduit à seule Image[25].

Deuxième exemple: des boutons contrôlés par des scripts

Après le clip, le bouton est le deuxième objet le plus utilisé dans les animations *Flash*. Lui aussi possède un gestionnaire d'événements[26] riche en méthodes et propriétés. Toutefois, il sera bien nécessaire d'associer une autre action à un événement produit par un bouton.

Tout comme pour le clip, c'est sur l'occurrence du bouton que vient se greffer le script. Et plus spécifiquement sur l'état **Cliquable** de ce bouton. Mais le logiciel se charge de cette étape lui-même.

Envisageons quelques cas simples:

Pour éviter qu'une animation ne démarre automatiquement lors de la lecture, il est conseillé de placer sur la première Image un script: *stop ( )*. La lecture de l'animation se fera alors, par exemple, par l'enfoncement d'un bouton.

Analyse

Il faut insérer une première Image sur laquelle on place le script d'arrêt de la tête de lecture. On place également sur cette première Image une occurrence du bouton que l'on a préalablement dessiné et enregistré comme tel. Après avoir sélectionné l'occurrence du bouton présent sur la scène et après lui avoir donné un nom d'occurrence, on peut lui ajouter un script. Par exemple: aller à la seconde Image et s'arrêter sur cette Image.

[25] On en arrive alors à de la programmation pure. Comme *l'ActionScript* ne peut se suffire à lui-même, il faut le placer sur une Image d'un scénario.

[26] Consulter également l'utilisation de l'objet Bouton en fin de document.

Script

```
on (release) {
    gotoAndStop (2);
}
```

Il n'y a pas que l'arrêt ou la marche de la tête de lecture qui puisse être prise en charge par un bouton. Ceux-ci peuvent également déclencher ou arrêter des mouvements de clips, modifier le contenu d'une variable, changer la séquence de lecture,....

Rappel: il faut que la tête de lecture soit à l'arrêt sur l'Image contenant un ou plusieurs bouton(s) si l'on veut que ceux-ci soient utilisables.

Quelques exemples qui font appel à des scripts que l'on doit maintenant commencer à comprendre.

Script 1: **un bouton qui déclenche le mouvement d'un clip**

```
on (press){
    monClip._x += 5
}
```

Script 2: **un bouton qui teste une valeur de variable introduite par l'utilisateur**[27]

```
on (release){
    if (test>0 && test<=3){
        reponse = "OK";
    } else {
        reponse = "FAUX";
    }
}
```

Script 3: **un bouton qui ouvre une fenêtre du navigateur et qui charge la page demandée**

```
    on (press) {
        getURL("http://www.det.fundp.ac.be/cefis/");
    }
```

[27] Il faut avoir défini, par ailleurs, une variable "test" sur un objet de type "texte de saisie" et une variable "reponse" sur un autre objet de type "texte dynamique". Ces objets texte seront envisagés un peu plus loin.

Script 4: **un bouton "curseur" que l'on peut déplacer le long d'une droite**

```
on (press) {
    startDrag (boutonson, false, 95, 370, 250, 370);
    volumeson = 1;
}
on (release) {
    stopDrag ( );
    volumeson = 0;
}
```

Script 5: un bouton qui peut recevoir des codes correspondants à des touches du clavier (ici pour mettre un clip en mouvement de rotation)

```
on (keyPress "g") {
    roue._rotation = roue._rotation-5;
}
on (keyPress "<Right>") {
    roue._rotation = roue._rotation+5;
}
```

Troisième exemple: la création d'un curseur personnalisé et animé

Analyse

Pour pouvoir créer un curseur animé, il faut avant tout disposer d'un clip animé. La première étape du travail est donc la réalisation de ce clip.

Dans un second temps, il faut faire comprendre au logiciel que le curseur actuel doit être remplacé par le clip. Comme dans tout programme informatique, le remplacement est la combinaison de 2 actions: suppression et insertion. Il faut donc supprimer[28] le curseur actuel (qui est représenté par une flèche oblique) et ajouter le nouveau curseur.

Détail des étapes

La construction du clip est entièrement autonome par rapport au reste du travail. Il ne s'agit pas de programmation mais tout simplement de la création d'un objet sur la scène enregistré comme symbole de type clip. Ce symbole sera automatiquement inséré dans la bibliothèque.

Lors de la seconde phase de construction, il est indispensable de récupérer le symbole clip sur la scène et lui donner un nom d'occurrence. Cela, afin d'éviter que le symbole lui-même ne

[28] Il ne s'agit pas de supprimer de manière pure et simple mais plutôt de masquer le curseur actuel.

prenne les propriétés spécifiques que l'on va attribuer à un "exemplaire" de ce symbole.

Il faut ensuite donner les instructions à cette occurrence pour qu'elle réagisse comme souhaité. Une fois l'occurrence du clip sélectionnée, commencez alors à rédiger le script par l'intermédiaire du panneau **Actions**.

Script

```
1 onClipEvent (load) {
2           Mouse.hide ( );
3           startDrag (this, true, 0, 0, 550, 400);
4 }
```

Explications du script

- Puisqu'il s'agit d'une action sur un clip, il faut d'abord faire appel au gestionnaire d'événements des clips: *onClipEvent* ( )

- Il faut ensuite préciser quelle instruction doit être exécutée sur le clip: *load* (l'action est initiée dès que le clip est instancié et apparaît dans le scénario)

- les actions menées sur ce clip (actif dans le scénario) sont au nombre de 2:

- la disparition de la souris: Mouse.hide[29] ( ). Il s'agit d'une action sur l'objet souris donc une méthode de la classe Mouse.
- l'accrochage du nouveau clip au pointeur. Il faut donc trouver une méthode dans la classe MovieClip qui accroche ce clip et faire en sorte que ce clip suive les mouvements du pointeur: c'est l'instruction StartDrag[30] ( ). Des paramètres doivent être précisés dans cette méthode. Examinons ces paramètres:
  - this[31]: signifie que c'est de ce clip là qu'il s'agit. C'est un mot-clé qui permet de faire référence à un objet ou à une occurrence de clip.
  - true: Une valeur booléenne spécifiant si le clip "accroché" est verrouillé au centre de la position de la souris (true) ou verrouillé sur le point auquel l'utilisateur a cliqué sur le clip (false). Ce paramètre est facultatif.

[29] Mouse.hide: cette méthode ne nécessite aucun paramètre. Elle a pour but de masquer le curseur dans une animation. Le curseur étant visible par défaut.

[30] StartDrag: cette action rend le clip *cible* déplaçable pendant la durée de l'animation. La cible étant bien entendu le clip dont nous parlons dans cet exemple.

[31] This : mot réservé qui désigne l'objet dont il est question dans l'action. Il s'agit ici du clip qui est déplacé par le curseur de la souris. Ce mot *this* pourrait être remplacé par le nom de l'occurrence du symbole clip qui est utilisé. Ce nom doit alors être placé entre guillemets.

< les 4 valeurs suivantes sont les coordonnées des points qui délimitent le champ d'action du clip (à gauche, en haut, à droite, en bas). Il est conseillé d'utiliser les mêmes valeurs que celles utilisées dans les propriétés du document *FLA*[32].

Une fois ce script terminé, il reste à tester l'animation.

Quelques remarques

L'animation ainsi créée ne comporte actuellement qu'une seule image. Si le curseur animé doit être actif pendant toute la durée de l'animation, il faudra insérer une dernière Image-clé sur le calque contenant ce curseur animé. La position du calque dans le scénario a aussi de l'importance. Il doit se trouver en avant de tous les autres calques sinon il risque de disparaître lors de la lecture de l'animation complète.

Si vous insérez ce curseur animé dans une animation comportant plusieurs séquences, le curseur ne sera actif que dans la séquence où il a été défini. En effet, lors de la lecture d'une autre séquence, l'action n'a pas été définie.

Il n'est pas non plus possible de définir 2 curseurs animés différents dans une même séquence mais un curseur différent peut être défini dans chacune des séquences d'une animation.

Quatrième exemple: accrocher un objet à un curseur

Il ne s'agit plus de masquer le curseur actuel et le remplacer par un clip mais simplement d'accrocher un clip au curseur. Les étapes seront les mêmes mais le script ne contiendra plus toutes les instructions de l'exemple précédent.

Les étapes

La construction d'un clip animé (ou non) reste la première étape de la construction. L'insertion de ce clip sur la scène de la première Image du scénario en est la seconde. Une fois que l'occurrence du clip a été renommée, on peut lui ajouter une action.

Script

```
1  startDrag ("nom de l'occurrence du clip", true)
```

Il n'y a plus qu'une seule ligne dans cette action. En effet, l'action n'est plus placée sur l'occurrence du clip mais sur un calque réservé spécifique. L'action est donc placée sur une Image mais en faisant référence dans les paramètres à l'occurrence du clip. Cette technique est possible quand il n'y a qu'une seule action à placer sur un objet. Dans le cas contraire, il faut faire appel au gestionnaire d'événements afin que toutes les actions soient regroupées. Les remarques envisagées dans l'exemple précédent restent d'application.

Cinquième exemple: déplacer un objet en utilisant le curseur

Cet exemple reprend les instructions de l'exemple précédent mais sont complétées puisque le

[32] Les dimensions, en pixels, de la fenêtre de sortie de l'animation.

relâchement du bouton gauche de la souris doit stopper le déplacement de l'objet. Il ne s'agit donc plus d'un clip associé au mouvement de la souris mais d'un bouton puisqu'il nous faut agir sur la "sélection" de l'objet par un clic sur le bouton gauche de la souris et la "dé-sélection" par un relâchement de ce même bouton de la souris.

L'objet qui sera déplacé aura le statut de clip. La première étape est donc de construire ce clip. Il servira dans un deuxième temps d'objet à placer sur l'Image du bouton. Puisqu'il s'agit bien d'un bouton que l'on va manipuler.

Lors de la création de ce bouton, les différents états ne doivent pas être nécessairement tous complétés. Seul l'état **Haut** peut contenir le clip créé. A moins que vous ne décidiez de modifier la couleur de l'objet sélectionné lorsqu'il est accroché au curseur de la souris.

La dernière étape sera la rédaction du script associé au bouton.

Sript

```
on (press){
    startDrag (this);
}
on (release){
    stopDrag ( );
}
```

Les paramètres de l'action startDrag peuvent être complétés si l'on veut que l'objet ne puisse pas dépasser les limites d'un cadre réservé, par exemple.

En répétant la même manipulation sur plusieurs clips, on peut déjà créer un puzzle où les pièces de départ serait positionnée sur le même calque. Chaque partie du puzzle serait un bouton différent. Pour que l'utilisateur réalise le travail de reconstitution à la perfection, il faudrait aussi traiter les coordonnées des points d'alignement des différents "morceaux" de ce puzzle.

Sixième exemple: un exemple de coloriage simple

Cet exemple permet de comprendre l'utilisation des classes objets conjointement avec les clips et les boutons. Il est cependant assez limité dans ses capacités. Il faudra, par la suite, combiner ces notions avec d'autres pour élargir les possibilités offertes par le langage.

Le but de cet exemple est de colorier une forme placée sur la scène en sélectionnant des boutons de couleurs différentes.

Les étapes

Il faut d'abord créer un clip contenant autant d'Images que l'on veut de couleurs différentes de coloriage. La première Image contiendra la forme de couleur blanche ou incolore. Les autres Images contiendront la même forme mais avec d'autres couleurs (rouge, bleu, vert, jaune,...)

Pour éviter que ce clip ne s'exécute indéfiniment sur la scène, il faut placer une action sur la première Image: *stop();* Placer ensuite ce clip sur la scène. L'occurrence de ce clip se nommera: **dessin**.

Créer ensuite autant de boutons (de la même forme et de la même grandeur pour une question d'esthétique uniquement) ayant chacun les couleurs de la forme dessinée dans le clip.

Sur chacun de ces boutons, il faut maintenant écrire un petit script qui donnera la couleur du bouton à la forme du clip. Il ne s'agit pas, dans ce cas, de prélever le code de couleur du bouton et le transférer à la forme[33] mais simplement de lire l'Image du clip qui correspond à la même couleur que celle du dit bouton.

Script:

```
on (press){
    dessin.gotoAndStop (n);[34]
}
```

On observe dans ce script que l'action *gotoAndStop* ( ) est appliquée à l'objet **dessin** et non pas à une image du scénario principal. Il s'agit, dans ce cas, de l'utilisation de la hiérarchie des clips[35] dans une animation.

Septième exemple: l'utilisation des objets de type texte

Les objets de type texte peuvent avoir 3 statuts différents: texte statique, texte dynamique et texte de saisie[36].

Les objets "texte statique" ne seront pas envisagés dans cette partie *d'ActionScript* étant donné qu'ils n'entrent pas en jeu dans l'interactivité. Par contre, les deux autres types d'objets texte sont beaucoup utilisés dans les animations *Flash* interactives puisqu'il s'agit aussi d'un moyen de communication entre l'utilisateur et le programme que l'on veut utiliser.

Les textes de saisie et les textes dynamiques[37] seront:

- créés sur la scène et resteront des zones vides,
- affectés d'un nom d'occurrence pour une plus grande lisibilité et
- dotés d'un nom de variable pour qu'ils puissent être traités par le langage de programmation.

Pour manipuler ces textes, il faudra donner les instructions par l'intermédiaire de boutons. Prenons un exemple simple: le comptage des lettres d'un mot.

---

[33] Cet exercice peut également être réalisé mais demande une connaissance un peu plus approfondie *d'ActionScript*.

[34] n est le numéro de l'Image du clip qui correspond à la même couleur que celle du bouton.

[35] Voir le chapitre suivant réservé à cette hiérarchie.

[36] Relisez éventuellement les notes du chapitre "La scène et ses objets" pour connaître la signification de ces 3 statuts.

[37] Sous entendu les objets de type texte de saisie et les objets de type texte dynamique.

Les étapes

Créer une première zone de texte de saisie. Donnons-lui comme nom d'occurrence: **comptage** et comme nom de variable **mot**.

Créer une seconde zone de texte de type dynamique cette fois. Son nom d'occurrence pourrait être **resultat** et son nom de variable **longueur**.

Créer enfin un bouton quelconque. Sur ce bouton, ajouter le script permettant de compter les lettres du mot qui sera encodé dans la zone de saisie. On travaille ici avec des objets de la classe **"String"**[38]. Le contenu de cette variable peut aussi être des caractères "chiffres" ou des caractères spéciaux.

Script

```
on (release){
    longueur = mot.length
}
```

Remarques:

L'événement sur le bouton aurait également pu être **on (press)**,

Le contenu de la variable mot est traité par la propriété **length** qui donne la longueur ou plutôt qui compte le nombre de caractères de la chaîne de caractères. Elle renvoie donc une valeur numérique.

Le contenu de la variable **longueur** est ensuite remplacé par la valeur trouvée par la propriété **length** puis est affichée dans la zone texte qui a été réservée sur la scène.

Une consultation des méthodes et des propriétés de l'objet "**String**" disponibles sera sans doute nécessaire.

De la même manière, on peut écrire des scripts qui utiliseront des objets encodés qui sont des variables numériques. Voici un exemple de calcul simple:

Script: **calcul d'une racine carrée**

```
on (release, keyPress "<Enter>"){
    aucarre = Math.sqrt(valeur);
}
```

- les événements possibles sont le relâchement du bouton gauche de la souris ou la pression de la touche "Enter" ou "Entrée",

[38] Chaîne de caractères

- valeur étant la variable de la zone de texte de saisie et aucarre la variable de la zone de texte dynamique,
- **sqrt** est une méthode de la classe **Math**.

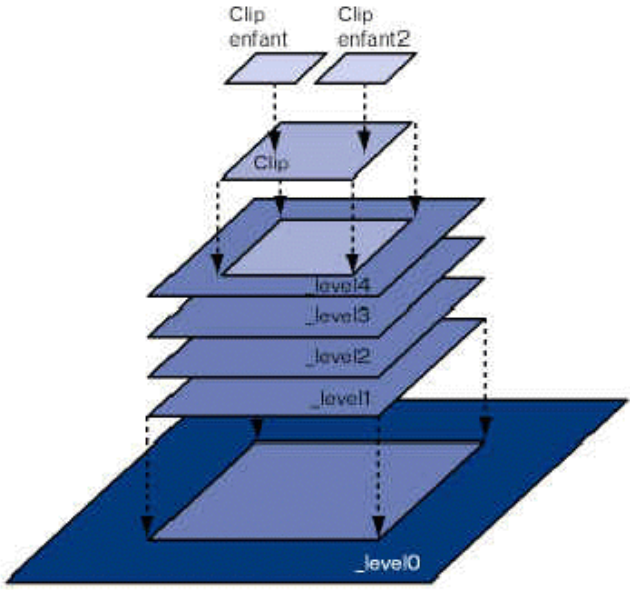
Les objets de type texte dynamique vont également être fort utilisés dans les animations interactives contenant des compteurs (de bonnes réponses par exemple). Ces textes de saisie et textes dynamiques peuvent encore être combinés avec des listes de variables. Citons comme exemple un traducteur français-anglais.

Ces actions nécessitent une analyse et une organisation un peu plus complète des informations que l'on doit traiter. Elles seront envisagées dans le chapitre suivant.

La hiérarchie dans les clips

La hiérarchie

Comme on l'a déjà signalé lors de l'utilisation de séquences différentes au sein d'une même animation, il est également possible d'insérer un clip dans un clip. Ceux-ci ne seront pas situés au même niveau de hiérarchie. Il en va de même lorsque plusieurs animations doivent être lues par *Flash Player*. Les fichiers *SWF* peuvent être chargés à différents niveaux. Comme le montre le schéma ci-dessous[39].



Pour avoir accès directement à ces animations et/ou aux objets présents dans une animation, on utilise la notation **POINT** dans le langage *ActionScript*[40].

La syntaxe à point est un outil typique du langage orienté objet. Il permet d'associer les objets, de définir un chemin dans une structure d'objets ou encore de définir une propriété.

Exemple 1:

> Soit un objet "ballon" et une propriété "changeCouleur" qui permet d'en changer la couleur. La commande sera : ballon.changeCouleur

[39] Schéma extrait du manuel d'Utilisation de *FlashMX* fourni avec le logiciel.

[40] La complexité du logiciel *Flash MX* est qu'il utilise la notion d'objet aussi bien dans l'interface graphique que dans son module de programmation *ActionScript*. Les 2 types d'objets sont bien entendu en étroite relation.

Dans la syntaxe à point, l'expression commence toujours par le nom de l'objet et se termine par la propriété qu'on lui affecte.

La syntaxe à point sert aussi à relier des objets et des animations et décrit visuellement leur hiérarchie enfant/parent

Exemple 2:

```
etoile.coccinelle._xscale += 5
```

L'exemple ci-dessus cible la propriété "_xscale" du symbole clip d'animation "coccinelle" lui-même placé dans le symbole clip d'animation "etoile" placé sur le scénario principal.

Il est possible de cibler les scénarios imbriqués de deux manières différentes:

Pour les objets reliés, on parle "d'alias":

_root qui pointe vers le plus bas des objets[41] c'est-à-dire vers l'objet dont le niveau hiérarchique est le plus haut.

_parent qui pointe vers l'objet précédent.

Deux méthodes sont encore possibles pour indiquer le chemin vers un niveau hiérarchique supérieur selon que l'on travaille en mode absolu ou relatif.

le mode absolu

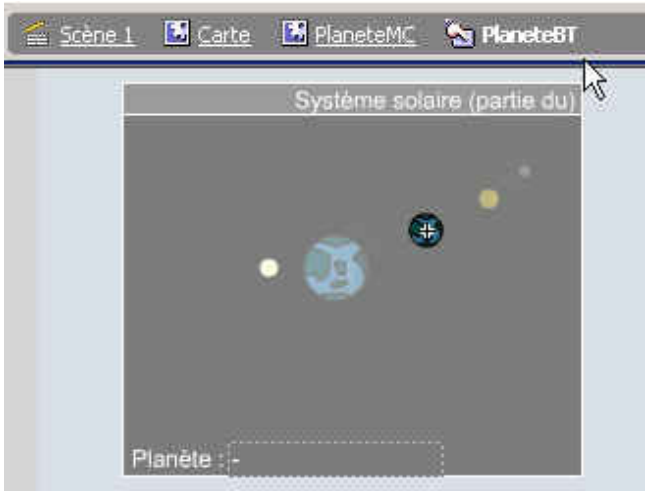
Ce mode prend comme référence la racine du fichier, c'est-à-dire le scénario principal et prend la notation: **_root**.

le mode relatif

En mode relatif, le chemin d'où est ciblé l'objet est primordial. Chaque chemin est défini en partant de celui-ci. Le scénario prend la notation: **this**

Lorsqu'il s'agit de hiérarchie d'animations on utilise la syntaxe **_levelx**[42].

Cette notation permet de réduire sensiblement la longueur de l'écriture des scripts. Encore faut-il que l'arborescence soit présente dans le travail de conception.



Exemple 3:

Le bouton "PlaneteBT" est placé dans le clip "PlaneteMC" qui lui-même est placé dans le clip "Carte". Ce dernier est placé sur la première Image du scénario principal. Voici le script qui indique le nom de l'occurrence du clip "PlaneteMC" dans la fenêtre de texte dynamique dont la variable

[41] Les objets sont empilés.

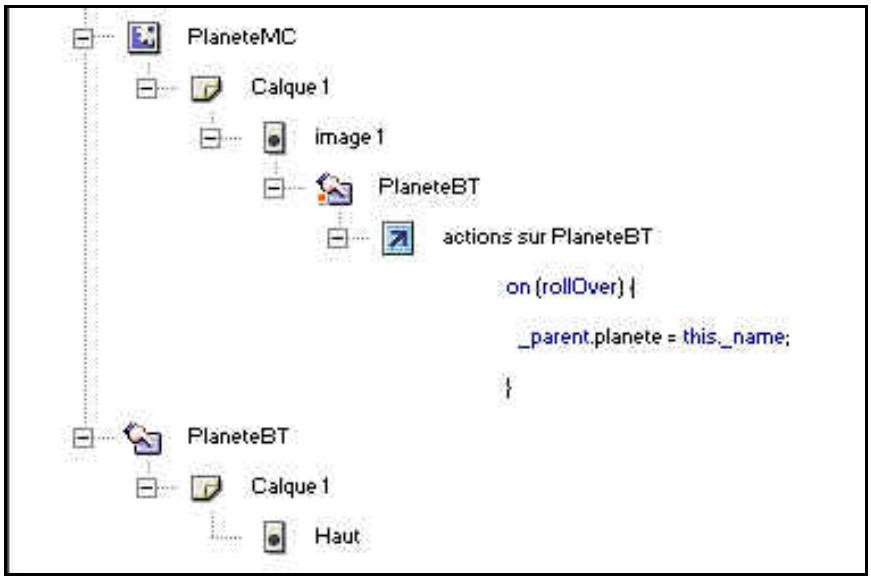
[42] **x** étant le numéro du niveau de l'animation. 0 étant le premier niveau (principal).

est aussi planete.

```
on (rollOver) {
        _parent.planete = this._name;
}
```

Cette méthode d'écriture des imbrications permet d'utiliser un même script (car provenant d'un même bouton) pour plusieurs occurrences d'un même symbole.

Voici une partie de l'image de l'explorateur permettant de comprendre cette imbrication.



ATTENTION: Les actions placées sur des clips imbriqués sont enregistrées dans les symboles eux-mêmes. Dans l'exemple ci-dessus, l'action sur le bouton est la même quelle que soit la planète placée sur la scène.

Dans ces cas, IL FAUT que les actions soient valables (car les mêmes) pour toutes les occurrences de ce symbole.

L'utilisation des actions, des fonctions et des variables peut également être basé sur cette notation Point qui découle de la *POO*. Il s'agit de la notion d'héritage présente dans *ActionScript*. En effet, dans la *POO*, les classes peuvent échanger entre-elles des propriétés et des méthodes selon un ordre bien établi. Prenons un exemple:

Soit la classe *véhicule* (qui serait une classe de moyens de transport) qui contient des sous-classes *voiture*, *camion*, *vélo*. Chacune de ces sous-classes pourrait elle-même contenir d'autres sous-classes.

Ces classes et ces sous-classes possèdent des propriétés et des méthodes:

Superclasse : véhicule - Propriétés : roues

Sous-classe: vélo - Propriétés: selle, pignon --- voiture - Propriétés: siège
Sous-classe de vélo: pignon - Propriétés: nombre de dents --- selle - Propriétés: couleur

- La classe qui transmet des propriétés est une superclasse,
- la classe qui hérite d'une superclasse est une sous-classe,
- la classe qui hérite et qui transmet des propriétés est à la fois superclasse et sous-classe.

Exemple:

Si vous définissez une fonction test( ) dans un clip nommé monClip, lui-même contenu dans le scénario principal, l'appel de cette fonction serait: **_root.monClip.test( );**

Si un autre clip imbriqué a également besoin de cette fonction, pour faire appel, on utiliserait l'écriture: **_parent.test( );**

Cette syntaxe à point permet de travailler sur un objet à partir de n'importe quel endroit du scénario.

Il est préférable, lors de la définition d'une fonction de l'associer à l'objet _global en vue de faciliter sa mise en oeuvre. Dès ce moment et quel que soit l'endroit du scénario où vous vous trouvez, vous pouvez faire appel à cette fonction par test( ); puisqu'elle a été définie comme étant une fonction globale.

Il peut en être de même pour les variables qui peuvent être globales ou locales (celles-ci n'ont une existence qu'à l'intérieur de la fonction). L'utilisation de variables globales reste cependant plus délicate car elle est active en permanence et risque d'interférer avec d'autres variables locales qui pourraient l'altérer, la détruire ou la recréer.

Les objets dans Flash

Le premier objet ***Object*** est le père de tous les objets dans Flash. Cet objet n'est pas destiné à une tâche particulière. On l'utilise souvent pour créer des objets personnalisés qui ont des tâches spécifiques.

Pour créer un nouvel objet (ou plus exactement une occurrence d'un objet) de quelque nature qu'il soit, il faut utiliser une fonction constructeur. La syntaxe est :

monObjet = new Object ( );

où *monObjet* est le nom de ce nouvel objet; *new* est l'opérateur et *Object*( )[43] est la fonction constructeur.

Cependant, il n'est pas nécessaire de créer tous les objets que l'on veut utiliser car il en existe déjà. On peut d'ailleurs les classer en 3 catégories différentes:

- les objets instanciables par l'inspecteur des propriétés,
- les objets instanciables,
- les objets prédéfinis.

[43] Les fonctions constructeurs s'écrivent toujours avec une majuscule.

Ces objets sont repris dans le tableau[44] ci-dessous.

Catégorie	Objets	Modes de création	Manipulation
Instanciables par l'inspecteur de Propriétés	Button	Sur la scène	Méthodes - Propriétés Evénements
	MovieClip	Sur la scène createEmptyMovieClip duplicateMovieClip attachMovie	Méthodes Propriétés Evénements
	TextField	Sur la scène createTextField	Méthodes - Propriétés Evénements (écouteurs)
Instanciables	Array	new Array [ ]	Méthodes - Propriétés
	Boolean	new Boolean	Méthodes
	Color	new Color	Méthodes
	Date	new Date	Méthodes
	LoadVars	new LoadVars	Méthodes - Propriétés Evénements
	Number	new Number	Méthodes - Constantes
	Object	new Object { }	Méthodes - Propriétés
	String	new String	Méthodes - Propriétés
	Sound	new Sound	Méthodes - Propriétés Evénements
	TextFormat	new TextFormat	Méthodes - Propriétés
	XML	new XML	Méthodes - Propriétés Evénements
	XMLSocket	new XMLSocket	Méthodes-Evénements
Prédéfinis	Accessibility	-	Propriétés
	Key	-	Méthodes - Constantes Ecouteurs
	Math	-	Méthodes - Constantes
	Mouse	-	Méthodes - Ecouteurs
	Selection	-	Méthodes - Ecouteurs
	Stage	-	Méthodes - Ecouteurs

[44] Extrait de *Flash MX Animation*, *interactivité, ActionScript*, Guylaine Monnier, Dunod, Paris, 2002

	System	-	Méthodes - Propriétés

Les composants

Outre les objets précédents, *FlashMX* a encore complété sa panoplie d'objets prédéfinis. Il s'agit d'éléments d'interface utilisateur. Ce sont des clips complexes auxquels sont associés des paramètres et des méthodes. Ces composants d'interface sont au nombre de 7.

ListBox	Zone de liste à sélection unique ou multiple
CheckBox	Case à cocher
ComboBox	Liste déroulante à sélection unique
PushButton	Bouton poussoir simple
RadioButton	Bouton radio (dans un même groupe, pas de sélection multiple)
ScrollBar	Barre de défilement horizontal ou vertical
ScrollPane	Fenêtre dotée de barres de défilement

Ils permettent une interactivité très complète sans trop de travail de conception. Ces composants peuvent également être accompagnés de scripts. Ceux-ci porteront sur les propriétés même du composant (couleur, police de caractères, alignement,...) ou sur des actions en relation avec l'animation principale.

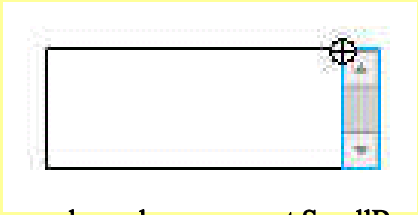
Envisageons quelques exemples:

Exemple 1: Utilisation du composant ScrollBar

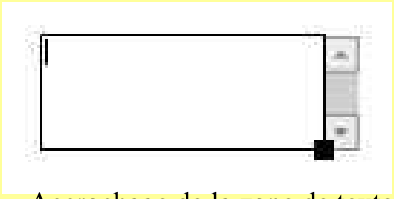
Ce composant va permettre de placer une quantité importante de texte dans une zone de texte dynamique prédéfinie. Au moment de la lecture du texte, il faudra utiliser cet "ascenseur" pour pouvoir en visualiser le contenu.

Méthode: création d'une zone de texte avec barre de défilement vertical

- Créer une zone de texte à l'aide de l'outil texte de la barre d'outils du logiciel.
- Dans les propriétés de cette zone de texte, sélectionner "texte dynamique" et donner un nom d'occurrence à cet objet.
- Sélectionner le composant "ScrollBar" dans le panneau Composants et le glisser sur la scène contre la zone de texte créée précédemment.
- Vérifier que ce composant est "accroché" à l'objet texte. Observer les deux schémas suivants.
- Insérer le texte dans la zone de texte.
- Lire la séquence pour observer le résultat. Il faut que le texte soit suffisamment long pour que les flèches haut et bas de la barre de défilement soient actives.



Accrochage du composant ScrollBar



Accrochage de la zone de texte

Si vous ne souhaitez pas devoir encoder tout le texte directement dans cette zone, il est aussi possible de l'enregistrer dans un fichier séparé et de faire appel à ce texte par l'intermédiaire d'un script. Les différentes étapes de la méthode précédente sont identiques sauf l'avant-dernière. Au lieu d'insérer le texte dans la zone prévue, il faut sur l'Image contenant ces objets écrire le script suivant:

Script:

```
loadText = new LoadVars( );
loadText.load ("sujet.txt");
loadText.onLoad = function ( ){
    defile.text = this.sujettext;
};
```

Quelques commentaires sont nécessaires:

- *1ère ligne du script*: il s'agit de la création d'un nouvel objet de la classe **LoadVars**. Cette classe permet de faire appel à des éléments externes. Cet objet est surtout utilisé pour insérer des pages Web.
- *2ème ligne du script*: c'est maintenant que le lecteur Flash charge le fichier texte nommé "*sujet.txt*". On observe que ce fichier doit être un texte brut (format *.txt*). Mais il pourrait également s'agir d'une adresse absolue sur Internet.
- *3ème ligne du script*: le gestionnaire d'événement **LoadVars.onLoad,** parfois appelé fonction de rappel, est invoqué lorsque l'opération **load** est terminée. La fonction reçoit un paramètre booléen (true ou false) si le chargement et l'analyse des données reçus par l'objet **LoadVars** est correct ou non. Le nouvel objet est rempli avec les variables téléchargées par l'opération **load** et les variables sont alors disponibles.
- *4ème ligne du script*: l'occurrence de l'objet texte dynamique (appelé ici *defile* est affecté de l'objet *sujettext*.

Encore quelques précisions importantes:

- Le fichier externe "*sujet.txt*" doit commencer par "*sujettext =*" afin que le lecteur *Flash* puisse contrôler s'il existe une correspondance entre le fichier texte appelé et le script.
- Même si le contenu du fichier externe est au format *txt*, donc avec une police de caractères

universelle, il est possible de modifier cette police, ainsi que d'autres attributs, par l'intermédiaire du **panneau de Propriétés** de l'objet. Le texte sera formaté avec la police, la taille et les autres attributs définis.

- Le fichier externe doit se trouver dans le même dossier que le fichier *FLA* et donc *SWF* faisant appel à lui.

Cette façon de créer une interaction risque d'être une étape de trop pour certains mais facilite la vie lorsqu'on veut insérer des textes plus conséquents dans une animation. La rédaction de textes par l'intermédiaire d'un éditeur de textes est quand même plus confortable qu'une petite fenêtre dans une animation *Flash*. Mais ce sera à chacun de juger...

Exemple 2: Utilisation du composant ScrollPane

Ce composant permet d'afficher dans une fenêtre dotée de barres de défilement horizontale et verticale , un clip dont les dimensions sont supérieures à celles de la fenêtre. Cela permet de ne pas trop encombrer l'écran. La méthode est assez simple.

Méthode: affichage d'un clip dans une fenêtre réduite

- L'objet qui s'affichera dans la fenêtre doit avoir le statut de clip. S'il s'agit d'une image JPG, elle sera transformée en clip. Cette image doit avoir été, au préalable importée dans la bibliothèque.
- Modifier les propriétés de ce clip par l'intermédiaire de la boîte de dialogue[45] ***Propriétés*** *en activant l'option* ***Exporter pour ActionScript****.*
- Insérer sur la scène une occurrence du composant ScrollPane (par un cliquer-glisser). Donner lui les bonnes dimensions.
- Renommer cette occurrence.
- Par l'onglet Paramètres de ce panneau de Propriétés, nommer le clip qui sera associé ou lié à ce composant. Compléter la première ligne des paramètres "ScrollContent" avec le nom du clip invoqué.
- Le clip n'est pas visible sur la scène dans le fichier FLA mais apparaît dès la lecture de l'animation

Un autre paramètre intéressant pour ce composant est **dragContent**. Si ce paramètre est *true*[46], l'utilisateur pourra utiliser, dans cette fenêtre, le cliquer-glisser de la souris pour déplacer le clip afin de le visualiser complètement.

[45] Un clic droit sur le clip se trouvant dans la bibliothèque permet l'accès aux propriétés de ce clip. Il sera peut-être nécessaire d'afficher les propriétés avancées pour trouver les options de liaison.

[46] Par défaut ce paramètre est *false*.

La publication

Bien qu'étant encore très loin d'avoir envisagé toutes les possibilités de l'utilisation de Flash, car ce programme n'a sans doute de limites que celles de l'imagination du concepteur, nous pouvons déjà envisager la publication de nos premières présentations.

Cette publication peut se faire sous plusieurs formats déjà cités précédemment.

Le format SWF

Le format le plus courant est sans doute le format *SWF*. Il permet de lire les animations par l'intermédiaire du *Flash Player*. Ce lecteur est totalement gratuit et peut être téléchargé sur le site propriétaire de Macromédia à l'adresse www.macromedia.com .

Les animations seront lues directement sans autre procédure. Cependant, il est également possible, et s'est la grande mode actuellement, d'insérer ces animations Flash dans des pages Web. Les animations gardant leur statut d'animations *Flash*.

L'insertion de ces animations se font par l'intermédiaire du code *HTML*. Deux balises doivent être renseignées lors de l'intégration du fichier *SWF* dans la page *HTML*: la balise **Embed** et la balise **Object**.

Voici un code regroupant ces deux balises nécessaires:

```
<html>
<head>
<title>Document sans titre</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
    <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/s
wflash.cab#version=6,0,29,0" width="600" height="350">
  <param name="movie" value="monfichier.swf">
  <param name="quality" value="high">
  <embed src="monfichier.swf" width="600" height="350" quality="high"
pluginspage="http://www.macromedia.com/go/getflashplayer"
type="application/x-shockwave-flash"></embed></object>
</p>
</body>
</html>
```

L'utilisation d'un logiciel de création de pages *HTML* tel que *Dreamweaver* permet une insertion aisée de fichiers *SWF*. C'est le logiciel, lui-même qui se charge d'insérer le code nécessaire.

Le format HTML

Ce format de publication permet de contrôler la lecture du fichier *SWF*, son affichage dans une page *HTML*. Le résultat obtenu est identique à la précédente manipulation. Ce sera alors au concepteur de pages *HTML* de compléter la page afin que le fichier Flash apparaisse dans un document plus complet. Cette technique est plus souvent réservée aux webmasters qui souhaitent créer une bannière animée dans leur modèle de pages Web.

Les formats GIF, JPG et PNG

Lors de cette manipulation, seule la première image de l'animation est transformée dans le format choisi. Suivant le type d'image, le choix se portera sur l'un ou l'autre des formats.

S'il s'agit réellement d'une animation, il est préférable de choisir l'option "Exporter l'animation" et ensuite choisir un format d'animation correct: *AVI* ou *Gif animé*. L'exportation d'une animation au format *JPG* par exemple créera autant de fichiers *JPG* que l'animation ne comporte d'Images dans la séquence.

Le format MOV

La publication au format *MOV* crée un fichier lisible par le lecteur *QuickTime*. Il est également possible d'exporter l'animation en choisissant le format *MOV* pour créer ce type de fichier.

La projection Windows[47]

La publication sous cette forme est différente des précédentes. En utilisant cette méthode, vous pouvez rendre votre animation tout à fait autonome. L'animation pourra être enregistrée sur un cédérom par exemple et ne nécessitera pas la présence du lecteur Flash.

En fait, en créant une projection *Windows*, le fichier incorpore le lecteur lui-même et c'est un fichier exécutable qui est créé. La taille du fichier sera cependant sensiblement augmentée. Il faut compter environ 770 ko supplémentaire à la taille de base du fichier *SWF*.

[47] Il est également possible de créer une projection Macintosh.

Utilisation de l'objet MovieClip

Les gestionnaires d'événements - Clips: onClipEvent ( )

charger	**load**	c'est l'événement par défaut. Les instructions conditionnées à cet événement s'exécutent une seule fois au chargement du clip d'animation
entrer Image	**enterFrame**	les instructions s'exécutent toutes les Images, à la cadence définie pour l'ensemble de l'animation.
vider	**unload**	les instructions s'exécutent lorsque le clip est déchargé par l'action unloadMovie ou lorsque le clip disparaît de la scène
souris enfoncée	**mouseDown**	les instructions s'exécutent lorsque l'utilisateur clique sur le bouton gauche de la souris
souris relâchée	**mouseUp**	les instructions s'exécutent lorsque l'utilisateur relâche le bouton gauche de la souris
déplacement de la souris	**mouseMove**	les instructions s'exécutent lorsque l'utilisateur déplace la souris sur la scène
touche enfoncée	**keyDown**	les instructions s'exécutent lorsque l'utilisateur enfonce une touche du clavier
touche relâchée	**keyUp**	les instructions s'exécutent lorsque l'utilisateur relâche une touche du clavier
données	**data**	vérifie le bon chargement des informations lors de l'utilisation de méthodes

Quelques propriétés de l'objet MovieClip

_x	exprime la coordonnée en abscisse du centre de l'occurrence par rapport au centre de l'objet dans lequel il est imbriqué
_y	exprime la coordonnée en ordonnée du centre de l'occurrence par rapport au centre de l'objet dans lequel il est imbriqué
_xscale	exprime la valeur de la largeur en pourcentage de l'objet
_yscale	exprime la valeur de la hauteur en pourcentage de l'objet

_width	exprime la valeur en pixels de la largeur de l'occurrence
_height	exprime la valeur en pixels de la hauteur de l'occurrence
_alpha	exprime la valeur en pourcentage de la transparence de l'occurrence (entre 0 et 100)
_rotation	exprime la valeur de rotation d'une occurrence (comprise entre -180 et 180 degrés)
_visible	permet de ne plus voir une occurrence du clip. Le clip est cependant toujours présent sur la scène
_currentframe	exprime la valeur du numéro de l'Image sur laquelle est placée la tête de lecture
_totalframe	permet de connaître le nombre total d'Images du scénario en cours
_mouse et _ymouse	permet de récupérer les coordonnées de la souris
_name	recupère le nom d'occurrence d'un clip
_target	recupère le chemin de cible de l'occurrence d'un clip

Utilisation de l'objet Bouton

Les gestionnaires d'événements - Boutons: on ()

appuyer	**press**	la souris est enfoncée sur la zone cliquable du bouton
relâcher	**release**	la souris est relâchée mais dans la zone cliquable du bouton. Si la souris est relâchée en dehors de cette zone cliquable, il n'y a aucun effet
relâcher en dehors	**releaseOutside**	la souris est relâchée une fois en dehors de la zone cliquable
touche	**keyPress " "**	une touche du clavier est enfoncée. Indiquez entre les guillemets le nom de la touche.
survoler	**rollOver**	la souris passe sur la zone du bouton. Elle le survole
sortir du survol	**rollOut**	toujours en survolant la zone, la souris en sort
faire glisser au-dessus	**dragOver**	la souris est enfoncée sur le bouton, en ressort et y revient, toujours enfoncée
faire glisser en éloignant	**dragOut**	la souris est enfoncée sur le bouton et en sort en restant enfoncée

Plusieurs gestionnaires peuvent être mis en oeuvre sur un même bouton, pour autant qu'ils n'entrent pas en conflit.

Quelques exemples supplémentaires

Afin de mettre au clair tous les concepts vus, il est peut-être intéressant de détailler quelques exemples qui reprennent une grande partie des notions envisagées dans les notes du cours. Ces exemples sont bien évidemment loin de reprendre tous les concepts de Flash.

Exemple 1: la neige qui tombe

Deux alternatives pour créer cet effet: soit la création d'une suite d'images sur lesquelles on place des "flocons" de neige qui se déplacent. Le travail est fastidieux. Soit la création d'un clip "neige" que l'on reproduira et que l'on déplacera par *ActionScript*.

Cette méthode est assez simple:

1. Créer un clip "neige" qui représente un flocon de neige. Sa forme est quelconque.

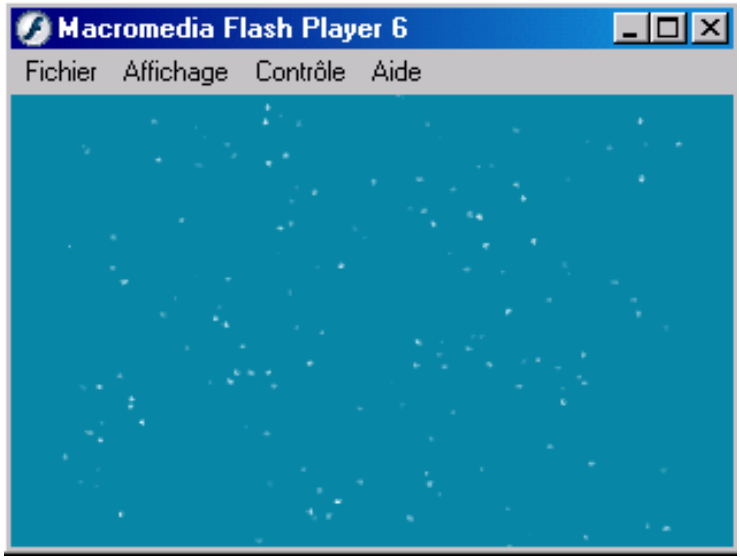
2. Placer une occurrence de ce clip sur la scène et le mettre en mouvement par un script. Le script est donc inséré sur l'occurrence du clip:

onClipEvent (load){	
_x = random(550);	*random*: fonction qui renvoie un entier compris entre 0 et le nombre entre ( ); permet de définir la position du flocon de neige
_y = random(350)+50;	
vitesse = random(4)+1;	vitesse: variable qui détermine la vitesse de chute du flocon
coef = random(100);	coef: variable qui détermine la grosseur du flocon
_xscale = coef;	*_xscale* et *_yscale*: échelle de grandeur du flocon
_yscale = coef;	
_alpha = random(80)+10;	*_alpha*: transparence du flocon (entre 0 et 100)
_rotation = random(360);	orientation du flocon
}	
onClipEvent (enterFrame){	
_y += vitesse;	
if (_y>400){	condition si la position aléatoire du flocon est trop grande
_y=-10;	
coef=random(100);	
}	
}	

3. Le script ci-dessus est créé pour un seul flocon.

4. Pour avoir plusieurs flocons sur la scène, il faut encore ajouter un script sur l'Image cette fois:

for (i=1; i<+350;i++){
duplicateMovieClip ("flocon","flocon"+i,i++);
eval ("flocon"+i)._x=random (600);
 	
eval ("flocon"+i)._y=random (150)+50;
}
	boucle de répétition
	duplicateMovieClip: répétition de l'insertion du clip "flocon"
	
	eval est une fonction qui permet de récupérer les variables, propriétés,... définies sur l'objet "flocon"



Exemple 2: la pluie

Cet exemple est semblable au précédent. Quelques variantes cependant: les gouttes tombent sur l'écran comme sur une vitre.

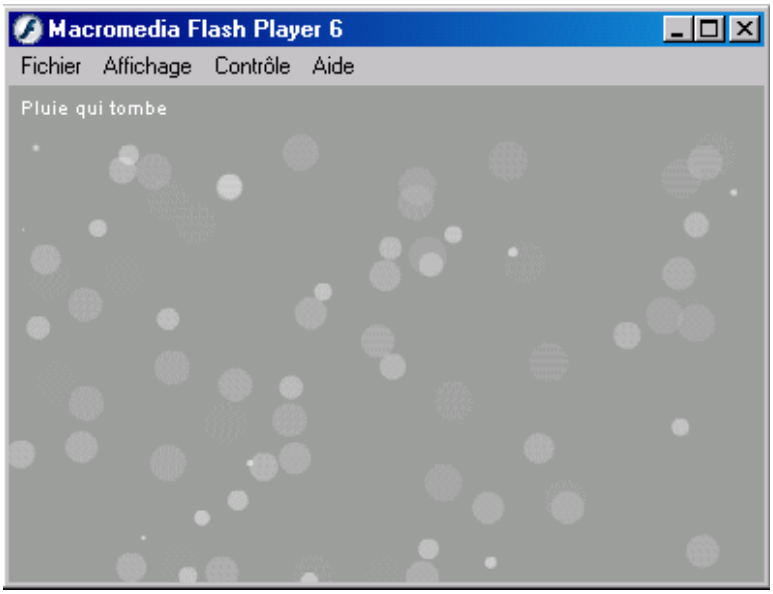
L'ActionScript sur le clip "goutte":

```
onClipEvent (load) {
        _x = random(600);
        _y = random(350)+50;
        coef = random(100);
        _xscale = coef;
        _yscale = coef;
        _alpha = 100-coef;
}
```

```
onClipEvent (enterFrame) {
        _xscale += 3;
        _yscale += 3;
        _alpha -= 3;
        if (_alpha<=0) {
                _x = random(600);
                _y = random(350)+50;
                coef = random(100);
                _xscale = coef;
                _yscale = coef;
                _alpha = 100-coef;
        }
}
```

et le script sur la première Image de l'animation:

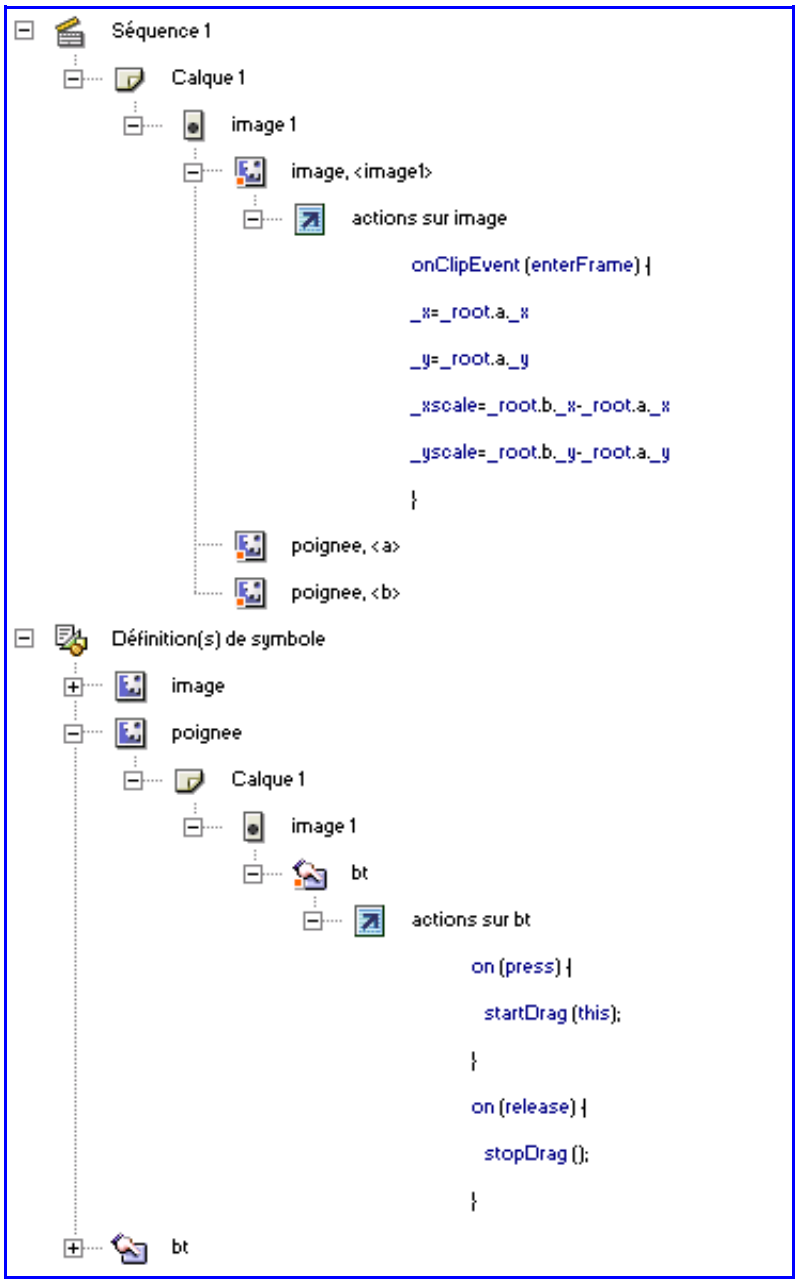
```
for (i=1; i<=150; i++) {
        duplicateMovieClip ("goutte", "goutte"+i, i++);
        eval("goutte"+i)._x = random(600);
        eval("goutte"+i)._y = random(350)+50;
}
```



Exemple 3: déformation d'objets

Il s'agit dans ce cas de créer une poignée sur un objet. Cette poignée permettra de redimensionner l'objet dans la fenêtre *Flash*.

L'analyse du problème nous apprend que cette poignée devra avoir le statut de bouton. Ce bouton pour pouvoir être déplacé sur la scène doit avoir le statut de clip. Ce clip devant permettre de déformer une image ou un objet quelconque doit quant à lui être associé à cette image ou à cet objet. Ce dernier, pour pouvoir être modifié doit aussi avoir le statut de clip. Voici l'arborescence de l'animation.



Quelques explications.

Créer le symbole bouton:

Sur la première image, dessiner un petit carré de couleur, de quelques pixels (s'aider de l'inspecteur des propriétés pour définir ses dimensions), mais ne pas écrire *d'actionScript* sur ce bouton.

Les autres images du bouton peuvent rester vides mais peuvent être complétées par un autre symbole (par exemple, la double flèche, comme dans la majorité des programmes).

Créer le symbole poignée qui a le statut de *CLIP*.

Ce symbole contiendra comme seul objet, le bouton qui vient d'être dessiné. Sur ce bouton sera inséré *l'actionScript* suivant:

```
on (press) {
        startDrag (this);
}
on (release) {
        stopDrag ();
}
```

qui permettra de déplacer la poignée.

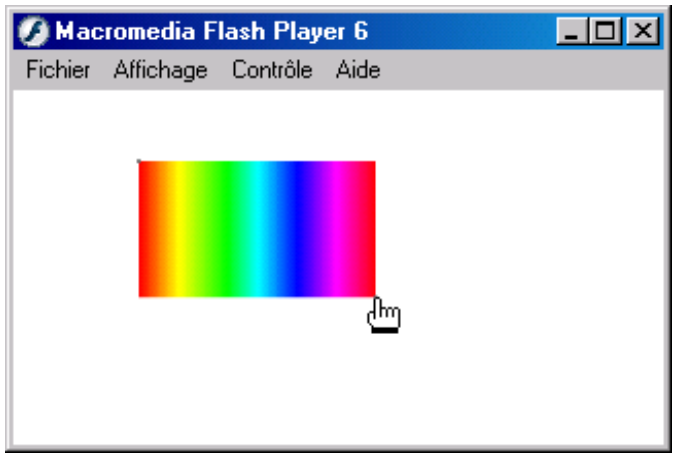
Créer le symbole image qui a le statut de clip.

Ce symbole peut contenir un objet quelconque, une forme, une image bitmap, un clip,...

Construire l'animation.

- Placer sur la scène, le symbole image et lui donner un nom d'occurrence (ce n'est pas indispensable mais préférable pour prendre de bonnes habitudes);
- Repérer la position et les dimensions de cet objet dans l'inspecteur des Propriétés.
- Placer sur la scène, le symbole poignée afin de le faire coïncider (point d'ancrage) avec le point d'ancrage de l'objet image. Se servir des informations de coordonnées et de dimensions affichées dans l'inspecteur des Propriétés pour plus de précision. Donnez aussi un nom d'occurrence à cette poignée (a).
- Placer une seconde poignée(b) sur un autre endroit de l'image.
- Sur cette image du scénario, écrire le script permettant d'interagir sur les dimensions de l'image lorsque le fichier sera exporté en *Flash*.

```
onClipEvent (enterFrame) {
_x=_root.a._x
_y=_root.a._y
_xscale=_root.b._x-_root.a._x
_yscale=_root.b._y-_root.a._y
}
```



Exemple 4: Défilement d'une image dans une fenêtre

La visualisation d'une image dont les dimensions sont plus grandes que celles de la fenêtre *Flash* peut se faire par l'intermédiaire d'une fenêtre. La version *FlashMX* dispose maintenant d'objets prédéfinis: les **composants**. Parmi ceux-ci, le composant *ScrollPane*.

La première étape sera d'importer l'image dans la bibliothèque du fichier. Avant de créer la fenêtre, il faut créer un symbole clip dans lequel on place l'image importée. Il est indispensable de modifier les propriétés de ce clip en sélectionnant l'option "**Exporter pour ActionScript**". Cette propriété peut être activée au moment de la création du symbole clip ou en sélectionnant l'objet dans la bibliothèque puis en cliquant sur le bouton droit pour activer le menu contextuel et enfin en sélectionnant l'item "**propriétés**".

Il reste à placer l'objet *ScrollPane* sur la scène. Par l'intermédiaire de l'inspecteur des propriétés, modifier, dans les paramètres, le contenu de cette fenêtre par le nom du clip invoqué.L'image sera visible et les ascenseurs seront actifs dès que le fichier sera publié.

Scroll Content	carte
Horizontal Scroll	auto
Vertical Scroll	auto
Drag Content	false

Dans cet exemple, il n'est même pas nécessaire d'écrire du code *ActionScript*.

Exemple 5: Faire bouger un objet par un clic de souris

Dans cet exemple, l'action d'un clic sur le bouton gauche de la souris provoquera un déplacement de cet objet vers le pointeur de la souris.

Les différentes étapes:

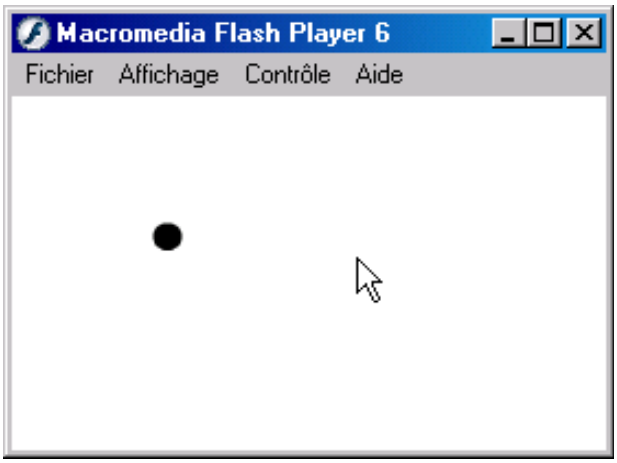
Créer un objet de type clip. Cet objet peut contenir un simple graphique ou un clip.

Insérer ce clip sur la scène et lui donner un nom d'occurrence (perso).

Sur l'image de la scène rédiger le script suivant:

xa = perso._x;
ya = perso._y;
surveil = new Object();
surveil.onMouseDown = function() {
  xa = _root._xmouse;
  ya = _root._ymouse;
};
Mouse.addListener(surveil);
//
//
perso.onEnterFrame = function() {
  perso._x -= (perso._x-xa)*0.1;
  perso._y -= (perso._y-ya)*0.1;
};
	xa et ya sont les coordonnées de l'occurrence "perso" placée sur la scène.
	Création d'un nouvel objet "surveil" par la création d'une fonction sur ce nouvel objet (lors du clic de la souris), on remplace les coordonnées de l'objet par les coordonnées de la souris.
	Ajouter l'écouteur "surveil" à l'objet Mouse afin d'activer cet objet lors d'un clic de souris.
	Déplacement de l'occurrence de l'objet lors du clic de la souris.
	0.1 détermine le pas de déplacement de l'objet entre 2 Images

Une remarque: en modifiant la cadence du document (40 images par seconde plutôt que 12) on obtient un déplacement plus fluide de l'objet.



Lexique de base

ActionScript: langage de programmation spécifique au logiciel permettant de personnaliser les animations.

Calque: feuille transparente contenant les éléments d'une animation.

Groupe: c'est un ensemble d'objets ou de formes. Les formes faisant partie d'un groupe ne peuvent plus être modifiées sauf en accédant au groupe par un double-clic.

Guide: calque ayant une fonction spécifique et contenant une trajectoire (ligne droite ou courbe) que devra suivre un objet placé sur un autre calque dont la propriété sera d'être guidé.

Image: c'est la plus petite division temporelle d'une animation. (A chaque image correspond un état et une position bien précise des différents objets qui composent l'animation.

Image-clé: elle est le passage obligé d'une animation. Flash se sert entre autres de ces images pour calculer les effets d'animation. Les images-clés sont également le seul endroit où l'on peut faire apparaître ou disparaître un objet lors d'une animation.

Image-clé vide: elle est similaire à une image-clé, mais elle ne contient aucun objet.

Masque: c'est un calque opaque qui cache ce qui se trouve au-dessous de lui et qui comporte des trous par lesquels on révèle les calques du dessous. Les formes que l'on dessine sur le calque de masque représente le ou les trou(s).

Occurrence: c'est la copie d'un symbole dans une animation. Il est possible de donner à des occurrences d'un même symbole, des attributs différents (taille, rotation, transparence,...) sans pour autant modifier le symbole d'origine.

Pelure d'oignon: c'est un mode d'affichage des objets sur la scène. C'est seulement un outil de travail qui permet de visualiser les images qui sont avant et après dans le scénario.

Scénario: ligne du temps

Scène: endroit où l'on place les différents objets

Séquence: partie d'une animation permettant une lecture linéaire ou non de toute l'animation suivant l'interactivité créée.

Symbole: c'est un élément réutilisable dans une animation. Il peut s'agir d'un dessin simple, d'une animation complète, d'un bouton, d'un fichier son d'une police de caractères. Ces symboles peuvent être utilisés plusieurs fois en différents endroits d'une animation: on parle d'occurrences du symbole.

Quelques raccourcis clavier utiles

, et ;	pour déplacer la tête de lecture image par image, vers la gauche ou vers la droite.
ALT + F3	pour ouvrir le panneau "Explorateur d'animations"
ALT+clic gauche de la souris sur un symbole	pour créer une occurrence de ce symbole (cela évite de devoir garder la bibliothèque ouverte en permanence.
CTRL + Entrée	pour lire l'animation à l'aide du lecteur Flash (et donc création du fichier *SWF*)
CTRL + F8	pour créer un nouveau symbole
CTRL+G	pour grouper plusieurs formes
CTRL+MAJ+G	pour dissocier un groupe de plusieurs formes
Entrée	pour lire l'animation dans la fenêtre de travail
F1	pour avoir accès au dictionnaire ActionScript
F11	pour ouvrir ou fermer le panneau "Bibliothèque"
F5	pour insérer une Image dans le scénario
F6	pour insérer une Image-clé dans le scénario
F9	pour ouvrir et fermer le panneau "Actions"

Bibliographie

Abstrakt Graphics, *Flash MX*, Micro Application Cookbook, Paris, Novembre 2002

AUBRY C., *ActionScript: programmer sous Flash 5*, Nantes Cedex 1, Editions ENI, 2001

BHANGAL S., *ActionScript. Optimisez vos sites Flash*, Dunod, Paris, 2001

BLATZ C., *Flash MX ActionScript*, Micro Application Référence, Paris, Août 2002

CHARTON E., *ActionScript pour Flash MX*, Campus Press, Paris, 2002

DECLERCQ S. *Flash sur Flash 4*, Compétence Micro Hors-série, Fontainebleau, Mai 2000

HOUSTE F., *Guide Microapp Macromedia Flash MX*, Micro Application, Paris, 2002

JAUNEAU M., *FlashMX pour débutants*, Compétence Micro Expérience, Ecuelles, Juillet 2003

LOTT J., *ActionScript en action*, O'Reilly, Paris, Novembre 2003

MONNIER G., *Flash MX: Animation, Interactivité*, *ActionScript*, Dunod, Paris, 2002

MOOCK C., *ActionScript pour Flash MX - La référence (2e édition)*, O'Reilly, Paris, Juillet 2003

Utilisation de Flash MX, document électronique accompagnant le logiciel, Macromedia Inc. Etats-Unis, 2002

VANDEPUT E., *Programmer avec des objets: une découverte de la POO à travers le langage Java*, http://www.det.fundp.ac.be/cefis/publications/etienne/poo2002.pdf

WARBERSSON K., *Animations web avec Flash 5*, Compétence Micro Hors-série, Ecuelles, Novembre 2001

WOLTER S. & ÜNLÜ S., *Flash MX ActionScript et Animations*, Micro Application Référence, Paris, Novembre 2002

WOODS P.S., *MX Flash*, First Interactive, Paris, 2002