Ingénierie Pédagogique Multimédia
Master professionnel

MASTER IPM EAD 2010-2012

Cours Flash

Chapitre 6 : Programmation avec ActionScript 3, partie 1

Rappel : les fichiers fla et swf sont dans le fichier « 6_Programmation_AS3_1.zip ».

SOMMAIRE

1 Objectifs du chapitre

La programmation en Flash étant une grosse partie de ce cours, le chapitre 6 dédié à la programmation ActionScript 3 est découpé en plusieurs parties qui vous seront fournies tout au long des semaines à venir.

A la fin de cette partie de chapitre, vous saurez :

- créer des animations multi-séquences ;
- programmer des actions simples pour piloter la tête de lecture,
- comment modifier les propriétés d'un clip,
- gérer des variables,
- et surtout à quoi correspond ActionScript, où on écrit du code ActionScript, comment on exécute ce code, etc.

2 Introduction

2.1 Pourquoi doit-on programmer ?

La programmation dans Flash permet principalement de rendre les animations interactives. Sans programmation, l'utilisateur se contente de regarder une animation comme un film au cinéma, il ne peut rien faire de plus ! Grâce à la programmation, il pourra interagir avec vos animations, par exemple déplacer des objets, saisir des valeurs (son email par exemple), etc.

La programmation (que ce soit en Flash ou non) permet beaucoup d'autres choses : faire des calculs, de la simulation, connecter des applications entre elles… mais en ce qui nous concerne ici, le but principal est **l'interactivité**.

Je ne résiste pas à l'envie de vous envoyer sur http://www.documaga.com/wp-content/uploads/2010/02/scaleofuniverse.swf qui vous montrera un très joli exemple d'interactivité vous permettant de vous balader dans les échelles de l'Univers. Sans programmation, il vous serait impossible de naviguer à votre aise de l'infiniment grand à l'infiniment petit et vice-versa.

2.2 Comment sait-on programmer ?

Même réponse que « comment sait-on écrire ? », **ça s'apprend**. Le but de l'IPM n'est pas de vous transformer en programmeur chevronné. C'est un autre métier qui demande des années. Non, le but de cette partie de cours est multiple :

- Comprendre les mécanismes de base de la programmation en Flash,
- Etre capable de réaliser des maquettes interactives,
- Acquérir une expérience vous permettant plus tard de dialoguer avec des programmeurs Flash et d'estimer le temps nécessaire pour réaliser une application interactive.

La meilleure façon d'apprendre à programmer est… la pratique, encore la pratique, et toujours la pratique. Donc n'hésitez pas à faire et refaire mes exercices, voire à créer vos propres exercices. Vous pouvez également consulter des ouvrages et des sites web pour cela, mais attention à ne pas perdre trop de temps sur ça…

2.3 ActionScript ?

ActionScript (abrégé par AS) est le langage de programmation de Flash depuis la version 4.
ActionScript existe en 3 versions : ActionScript 1 (AS 1) qui est proche de Javascript, AS 2 (depuis Flash 8) qui est très proche de Java, et AS 3 (depuis Flash CS3), évolution de AS 2, qui est encore plus proche de Java (si vous ne connaissez pas Java, cela n'a aucune importance pour ce cours, c'est juste une remarque pour ceux qui le connaissent). Nous n'étudierons ici que la version 3 (AS 3) qui est actuellement la version de référence pour la programmation en Flash et le langage le plus demandé dans les offres d'emploi autour de Flash.
Dans tout le reste du cours sur la programmation avec ActionScript, AS désignera la version 3, c'est-à-dire AS 3.

Comme tout langage de programmation, AS utilise :

- une *syntaxe rigoureuse* ; vous devez donc faire très attention à ce que vous écrivez (respect des espaces, des ponctuations, des minuscules/majuscules…), ainsi qu'à l'endroit où vous écrirez vos programmes,
- des *mots clés* que vous devrez connaître, ou être capable de deviner ou de retrouver dans l'aide en ligne ou tout autre type d'aide (sites web, livres,…),
- des *structures algorithmiques* pour structurer vos programmes (boucles, tests, …)
- des *variables*…

Nous verrons ces différents concepts (et d'autres) tout au long du cours.

Ce chapitre vous donnera de solides bases d'AS. Notez que le langage ActionScript comporte des centaines de fonctions… N'hésitez pas à parcourir l'aide ☺ .

Parfois, je donnerai des compléments possibles à certaines notions, et hors programme IPM (pour satisfaire la curiosité de ceux qui ont envie d'aller plus loin). Dans ce cas, j'indiquerai clairement que ces notions sont hors programme de Master et sont dédiées aux informaticiens ou aux Flasheurs.

Même après ce cours, il reste encore beaucoup de choses à découvrir dans la programmation Flash, mais je vous rappelle que le Master n'a pas pour objectif de faire de vous des Flasheurs professionnels. Néanmoins, avec tout ce que nous aurons vu avec les chapitres sur Flash, vous devrez être capable de faire des maquettes/prototypes suffisamment réalistes pour montrer ce qu'une version aboutie (fait avec des graphistes et des Flasheurs) devrait donner.

Pour information, et pour que vous sachiez ce que l'on peut faire de plus avec Flash, voici les grands sujets que nous n'aurons pas abordés en ActionScript (et que vous n'avez pas besoin de connaître en tant qu'IPM) :

- *programmation objet*, « la vraie, la pure, la dure »… avec notion de classe, d'héritage, d'instance, etc. (public informaticien)
- *programmation évoluée* : utilisation de patrons de conception (*design patterns*) tels que MVC, etc. (public informaticien)
- *programmation dynamique*, par exemple pour générer du code Action Script… (public informaticien)
- les *composants Flash* (utilisation et création). L'utilisation est du niveau IPM, la création est pour un public informaticien.
- les *comportements Flash* (cf. le panneau `Comportements`). Leur utilisation est assez simple ; à la fin du cours Flash, vous devriez être capables de comprendre leurs usages par vous-mêmes.
- les *effets de scénario* (menu `Insertion>Effets du scénario`).

Le programme est chargé, mais grâce à ce dernier chapitre, Flash n'aura plus beaucoup de secrets pour vous ☺.

Un dernier conseil : j'ai fait en sorte que les parties et sous-parties de ce chapitre se travaillent dans l'ordre indiqué par la table des matières. Elles sont de difficulté croissante.

3 Webographie

Avant de voir ce qu'est AS3, comment on écrit du code, etc., voici quelques liens qui pourront vous servir de ressources complémentaires.

3.1 Outils pour écrire du code AS3

Voici quatre outils dédiés aux Flasheurs et aux informaticiens. Pour notre cours d'AS3, vous n'avez pas besoin de les utiliser, mais vous risquez de les rencontrer dans votre vie professionnelle :

- Flash Builder (un des meilleurs, et gratuit pour les étudiants) : http://www.adobe.com/fr/products/flashbuilder/
- FlashDevelop (un des meilleurs) : http://www.flashdevelop.org
- outil Sepy (assez bon) : http://www.sepy.it/
- outil SciTe (le moins bon des 4) : http://www.scintilla.org/SciTE.html

3.2 Tutoriels

Quelques sites présentant des tutoriels sur AS3 (il y en a des centaines d'autres !) :

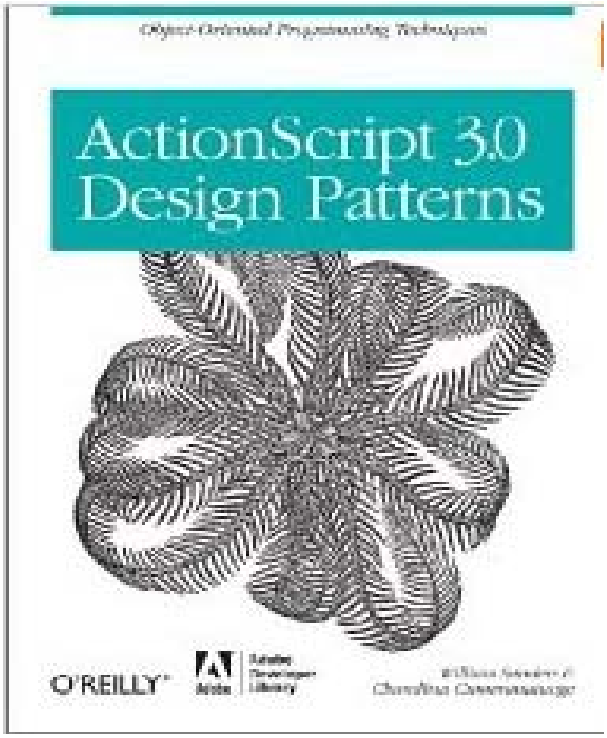
- http://www.gotoandlearn.com/
- http://www.yazo.net/ qui contient entre autre la **palette Yazo** qui vous permettra de gagner (peut-être) du temps pour l'écriture de scripts AS3 classiques (cf. le site web pour les détails). Ce site contient également un **mémento AS3** très pratique (cf. plus bas).
- http://www.toxiclab.org/search.asp?imageField.x=0&imageField.y=0&search=actionscript
- des tutoriels vidéos chez Adobe : http://tv.adobe.com/product/flash/
- tutoriels d'exception en AS3 : http://www.designspartan.com/tutoriels/10-tutoriels-dexception-pour-flash-as-3/
- Flash and Math: http://www.flashandmath.com/

3.3 Aide, documentation…

- la livedoc d'Adobe (plutôt pour public informaticien) : http://livedocs.adobe.com/flash/9.0_fr/ActionScriptLangRefV3/package-summary.html
- la doc officielle Adobe : http://www.adobe.com/support/documentation/fr/flash/
- une des bibles sur AS3 : http://www.actionscript.org/index.php
- Mémento AS3: http://www.yazo.net/racine/memento/MementoAS3.pdf

3.4 Livres

Je ne vous conseille pas d'ouvrage en particulier, il y en a tellement ! et il en sort très régulièrement sur Flash, sur AS3, sur Flex (qui fonctionne avec AS3) !!!
Sachez juste qu'aucun livre n'est nécessaire pour ce cours. Si vous voulez compléter vos connaissances en achetant un livre, vérifiez le niveau d'AS3 demandé ou visé. Des ouvrages comme celui-ci-dessous par exemple sont uniquement dédiés à des informaticiens chevronnés.


ActionScript 3.0
Design Patterns
O'REILLY

4 L'environnement de programmation

Entrons maintenant dans le vif du sujet. Avant de programmer, il faut connaître l'environnement dans lequel vous allez travailler.

Pour programmer en AS, il faut utiliser au minimum le panneau `Actions`, mais d'autres panneaux peuvent se révéler utiles, dont l'Aide ☺ et tout ce qui concerne le déboggage (plutôt pour informaticien).

Le panneau **`Actions`** (menu `Fenêtre>Actions` ou raccourci F9) se présente comme ci-dessous (cf. Figure 1) et contient :

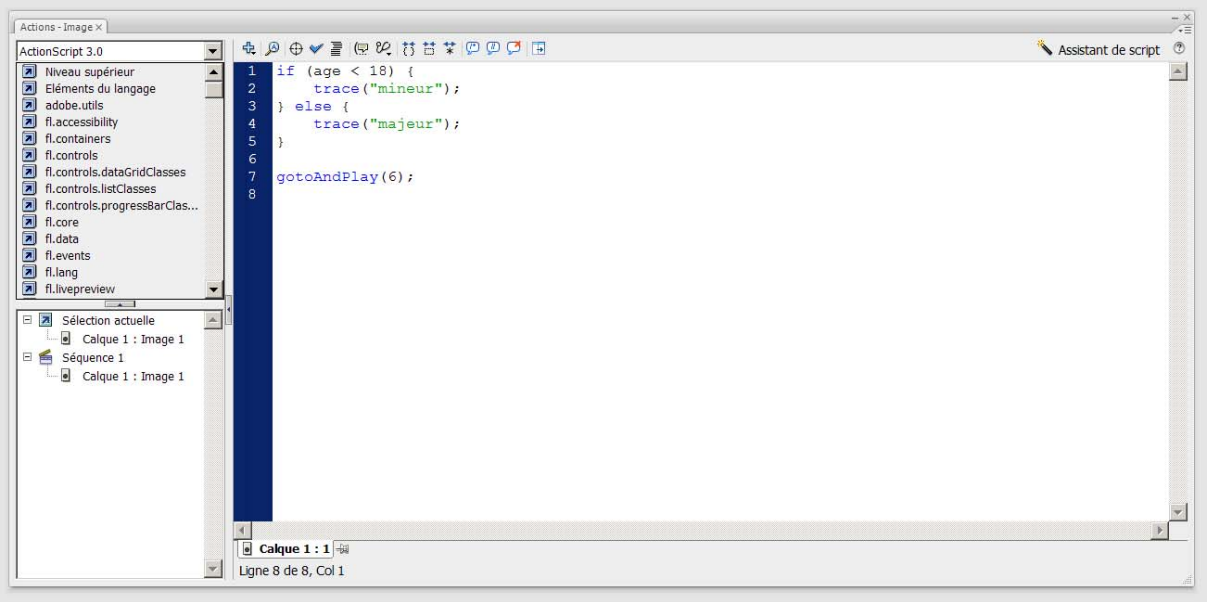


Figure 1 : Le panneau Actions (Flash CS3)

- (à gauche, en haut) le langage AS trié par types d'actions. Vérifiez bien que vous avez ActionScript 3 affiché comme ci-dessous (cf. Figure 2). Dans le cas contraire, cela veut dire que vous êtes actuellement en train de visualiser AS1, AS2 ou Flash Lite (qui est une version d'AS pour les téléphones mobiles).

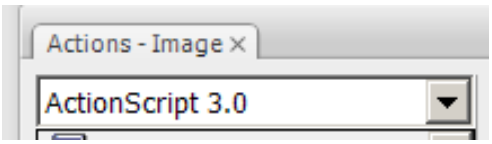


Figure 2 : Affichage de la version d'AS sélectionnée

- (à gauche, en bas), d'une part la sélection actuelle (c'est-à-dire là où se trouve le code affiché), et d'autre part la structure de l'animation (ce qui correspond presque à l'Explorateur d'animations accessible par le menu `Fenêtre>Explorateur d'animations` que je vous invite à approfondir par vous-mêmes)
- (à droite) la zone de saisie où vous taperez vos programmes AS. Cette zone contient en haut différents boutons qui sont :

- le bouton  pour ajouter une nouvelle action. D'autres possibilités existent pour ajouter des actions. Cependant, sachez dores et déjà que le plus simple et le plus rapide sera de taper directement le code. Vous pouvez donc oublier ce bouton…
- le bouton  pour rechercher et remplacer du texte dans le code,
- le bouton  pour insérer un chemin cible (nous verrons cela plus loin),
- le bouton  pour vérifier la syntaxe de votre code. En cas de problème, Flash vous affiche les erreurs trouvées.
- le bouton  pour mettre en forme votre code, c'est-à-dire le présenter de façon lisible (raccourci = CTRL-SHIFT-F). Quand vous utilisez cette fonctionnalité, Flash vérifie automatiquement la syntaxe de votre code et vous affiche les erreurs éventuelles.
- le bouton  pour afficher les conseils de code (cette aide précieuse est activée par défaut),
- le bouton  (plutôt pour informaticien) qui appelle les options de débogage (définir un point d'arrêt, le supprimer, ou supprimer tous les points d'arrêt),
- les boutons  (plutôt pour informaticien) qui permettent (de gauche à droite) de réduire le code entre accolades, de réduire le code sélectionné, ou de tout développer le code réduit. Voici quelques exemples ci-dessous. Vous pouvez taper le code présenté ici pour faire les tests, même si pour le moment vous ne savez pas encore à quoi cela peut servir :

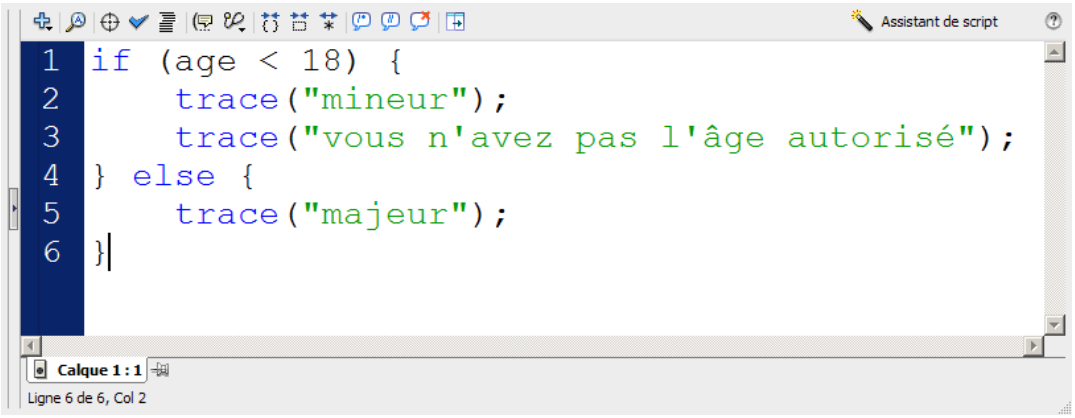


Figure 3 : exemple de code AS3 (ne vous occupez pas ici de la signification de ce code)

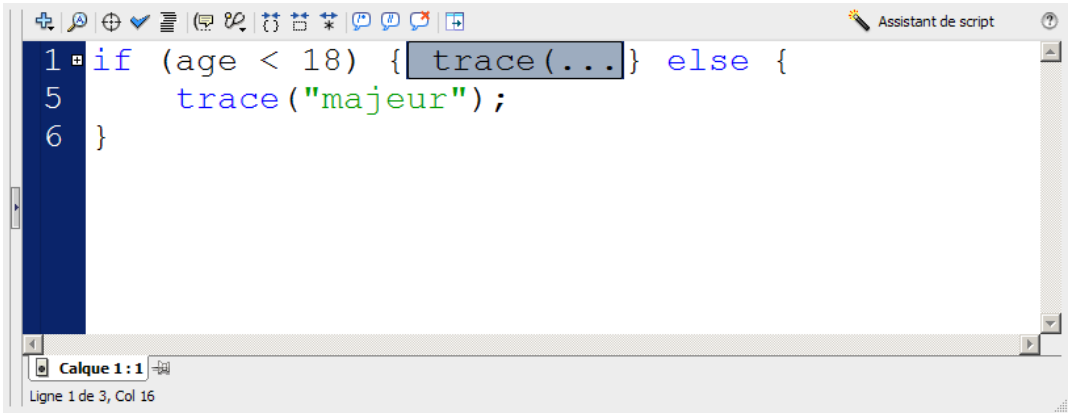


Figure 4 : on a cliqué sur le premier « trace » puis sur le bouton de réduction du code entre accolades

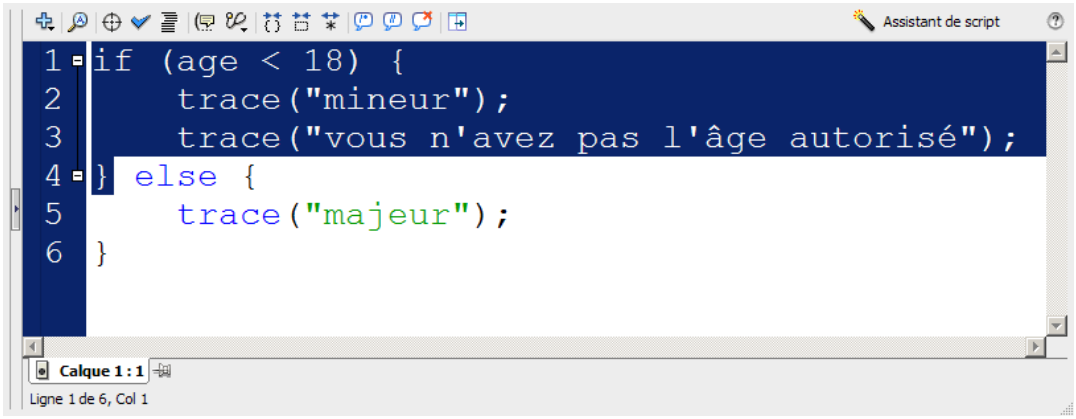


Figure 5 : exemple de code sélectionné

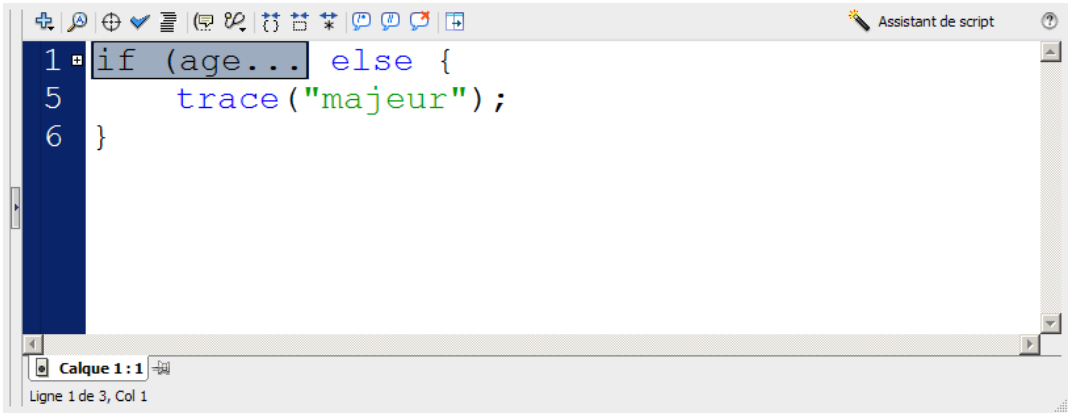


Figure 6 : on a cliqué sur le bouton de réduction de code sélectionné

- les boutons  qui permettent (de gauche à droite) de mettre les lignes sélectionnées dans un bloc de commentaires, de mettre chaque ligne de code en commentaires, et de supprimer les commentaires pour les lignes sélectionnée. Voici quelques exemples ci-dessous :

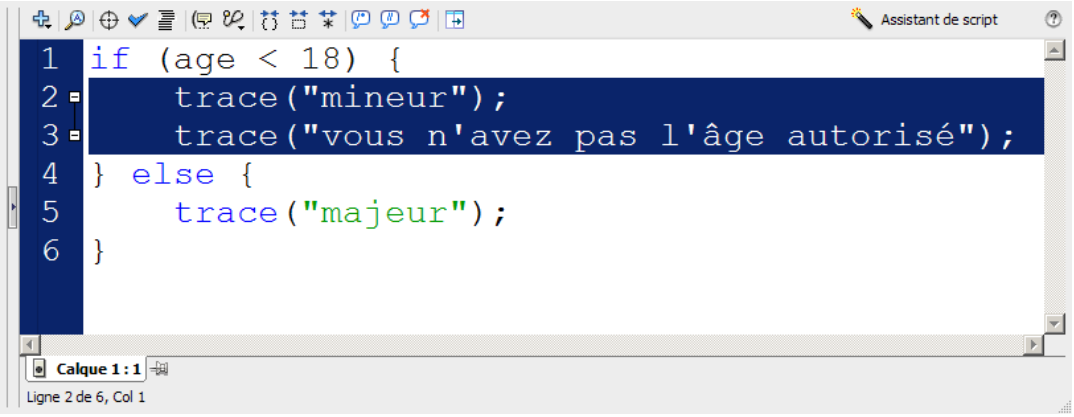


Figure 7 : code sélectionné avant application de mise en commentaires

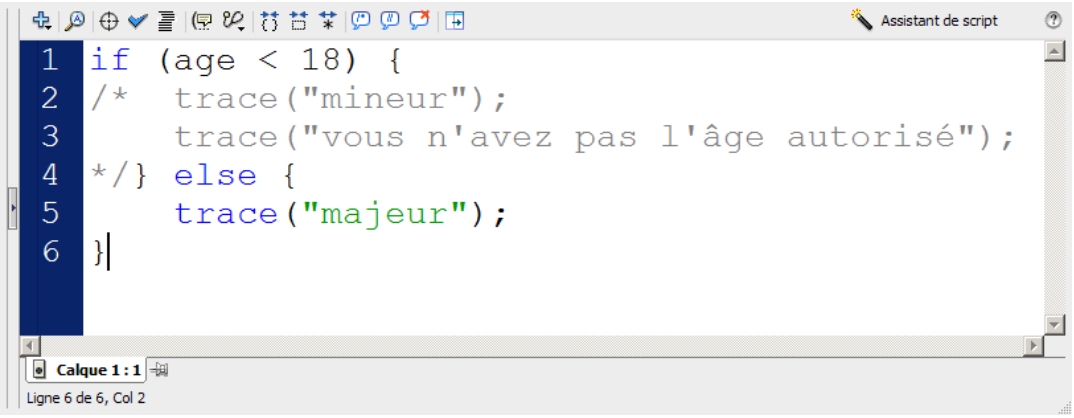


Figure 8 : mise en bloc de commentaires avec le premier bouton

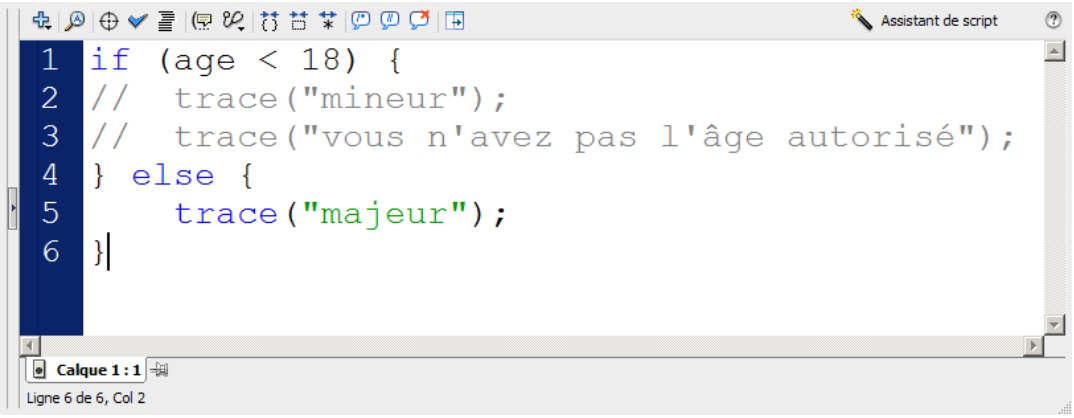


Figure 9 : mise en commentaire de chaque ligne avec le second bouton

- le bouton qui permet d'afficher ou de masquer la partie à gauche du code,
- le bouton Assistant de script, oubliez-le car totalement inutile en AS3,
- le bouton qui vous permet d'accéder à l'aide en ligne de Flash. Je ne vous en dit pas plus, il faut que vous alliez voir par vous-mêmes ! Par exemple, cliquez sur le « if » dans le code présenté ci-dessus, puis sur le bouton d'aide. Vous obtenez alors l'aide sur le « if ». Après, à vous de naviguer dans l'aide…
- (en bas) le bouton qui permet de verrouiller votre code, c'est-à-dire de le laisser à l'écran même quand vous sélectionnez un autre objet. **Faites donc TRES**

ATTENTION quand vous tapez du code ! Vérifiez en bas et à gauche du panneau Actions quel est l'objet concerné par votre code. En général, vous n'aurez pas à utiliser ce bouton.
Dans les figures précédentes, il était écrit « Calque 1 : 1 » à gauche du bouton. Cela signifie que le code affiché se trouve dans le calque qui s'appelle « Calque 1 », et plus précisément dans l'image 1 de ce calque (le « : 1 » indique le numéro de l'image).

- le bouton  qui affiche un menu complet pour le panneau Actions ; vous y trouverez toutes les possibilités énoncées ci-dessus ainsi que d'autres fonctionnalités telles que « préférences » qui vous permettra de choisir vos options de « format automatique », la taille et la police pour le code AS, etc.

Note : quand vous avez sélectionné plusieurs lignes de code, vous avez peut-être vu des signes « - » à gauche, comme ci-dessous :

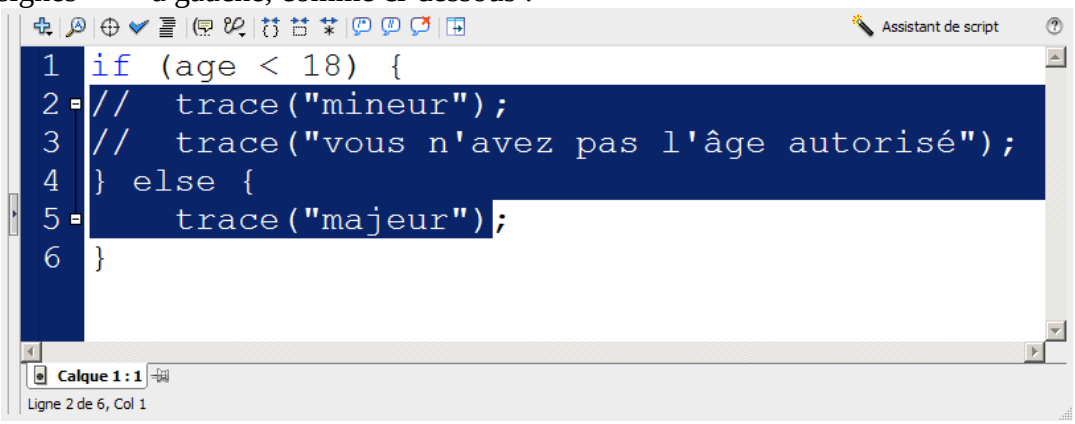


Figure 10 : exemple de code sélectionné. Notez les petits carrés blancs avec – dedans…

Le fait de cliquer sur ces « - » réduit le code sélectionné, ce qui affiche :

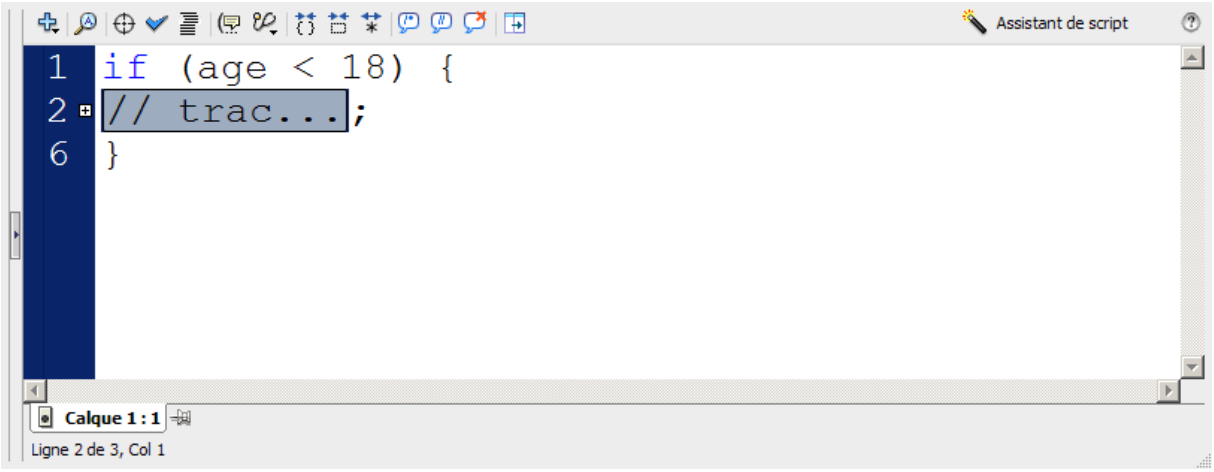


Figure 11 : résultat produit en cliquant sur un des petits carrés blancs

En cliquant sur le carré blanc à nouveau (avec + dedans cette fois-ci), vous pouvez « développer » le code réduit.

5 Conseils de code et Assistance

Si vous voulez que le panneau Actions vous aide à saisir du code pendant que vous tapez votre code (en plus des bulles d'aide qui s'affichent), vous devez ajouter un suffixe spécial à chaque nom d'occurrence. Par exemple, pour afficher des conseils de code pour les MovieClip (c'est-à-dire les clips), vous devez affecter à tous les objets MovieClip le suffixe `_mc` (pour Movie Clip), comme dans les exemples suivants :

```
Cercle_mc.gotoAndPlay(1);
Carré_mc.stop();
```

En procédant ainsi, lorsque vous tapez « `Cercle_mc .` », vous verrez apparaître une liste proposant toutes les fonctions associées aux clips (cf. Figure 12).

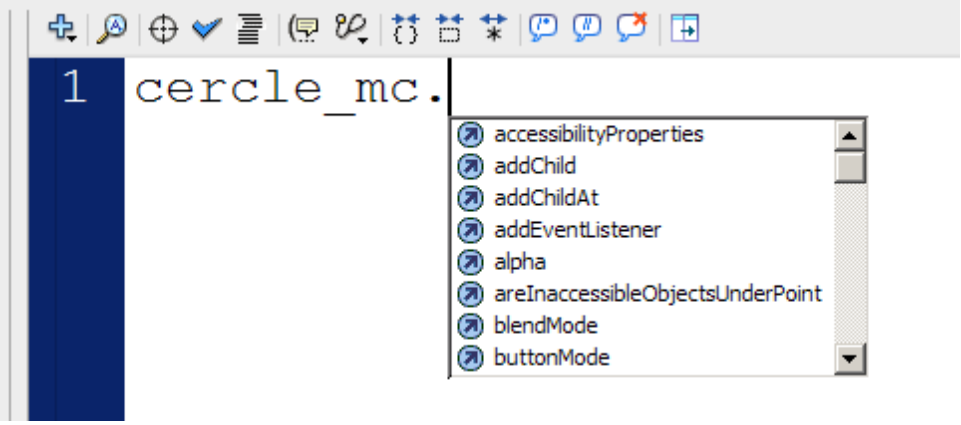


Figure 12 : Assistance d'écriture de code Action Script

Le tableau suivant présente les suffixes et les classes (types) d'objets correspondantes :

Type d'objet	Suffixe
Array	_array
Button	_btn
Camera	_cam
Couleur	_color
ContextMenu	_cm
ContextMenuItem	_cmi
Date	_date
Error	_err
LoadVars	_lv
LocalConnection	_lc
Microphone	_mic
MovieClip	_mc

MovieClipLoader	_mcl
PrintJob	_pj
NetConnection	_nc
NetStream	_ns
SharedObject	_so
Sound	_sound
String	_str
TextField	_txt
TextFormat	_fmt
Video	_video
XML	_xml
XMLNode	_xmlnode
XMLSocket	_xmlsocket

Tableau 1 : Suffixes à utiliser pour faire afficher automatiquement les fonctions associées

Par ailleurs, si vous êtes sur un objet dont Flash est capable de comprendre le contenu, Flash vous proposera automatiquement une aide pour la saisie de code. Dans l'exemple ci-dessous, Flash propose automatiquement les fonctions associées à la chaîne de caractères ; quand la liste est ouverte, il suffit de taper les premières lettres pour que Flash vous positionne dans la liste des fonctions possibles.

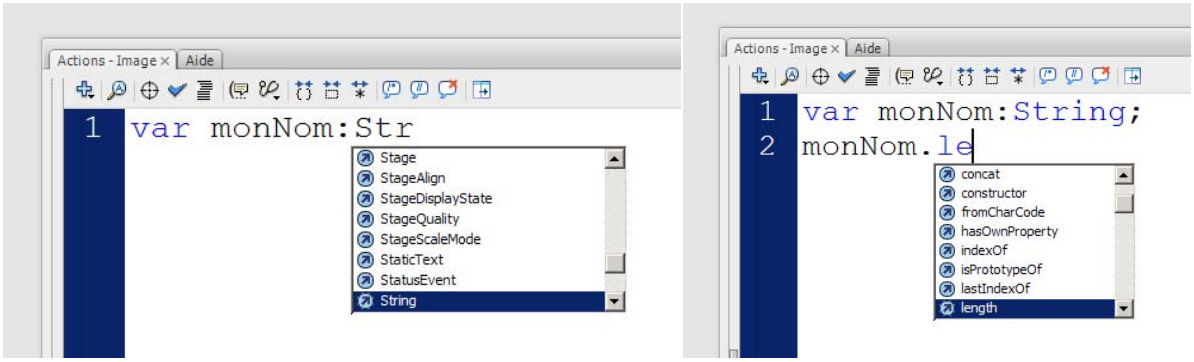


Figure 13 : Assistance d'écriture de code ActionScript

Pour valider ce que Flash vous propose, sans être obligé de tout taper au clavier, par exemple pour que Flash écrive `String` alors que vous n'avez tapé que « `st` » comme à gauche dans la Figure 13, il vous suffit d'appuyer sur la touche Entrée, ou Tabulation, ou encore de continuer à écrire le code qui devrait être après ce que Flash propose, par exemple la touche ; . Ainsi dans la Figure 13 à gauche, pour la ligne 1, on tapera au clavier : « `var monNom:St;` » au lieu de « `var monNom:String ;` ». Pratique non ?

Généralement, Flash vous affiche également la syntaxe des fonctions, à condition que vous respectiez la casse des fonctions. Sur la Figure 14, à gauche, Flash n'affiche rien, alors qu'à droite Flash affiche une bulle d'aide contenant la syntaxe de la fonction. Notez que le code de gauche ne fonctionnera pas en AS3 ! **Il est donc très important que vous respectiez impérativement la syntaxe de Flash** si vous ne voulez pas avoir de surprises à l'exécution de votre code.

Figure 14 : Affichage de la syntaxe quand on respecte la casse du langage

6 Où écrire du code AS3 ?

La réponse est triple :

- dans des images de scénario
- dans la classe du document
- dans des fichiers ActionScript.

Pour le moment, nous allons travailler avec la première solution uniquement. Nous aborderons les deux autres solutions plus tard dans le cours.

6.1 Comment écrire du code dans des images ?

Pour écrire du code AS3 dans des images, vous devez préalablement avoir ouvert ou créé un fichier Flash AS3. Pour créer un tel document, deux solutions :

- soit vous cliquez sur « Fichier Flash (AS 3.0) » dans la fenêtre d'accueil de Flash comme ci-dessous,

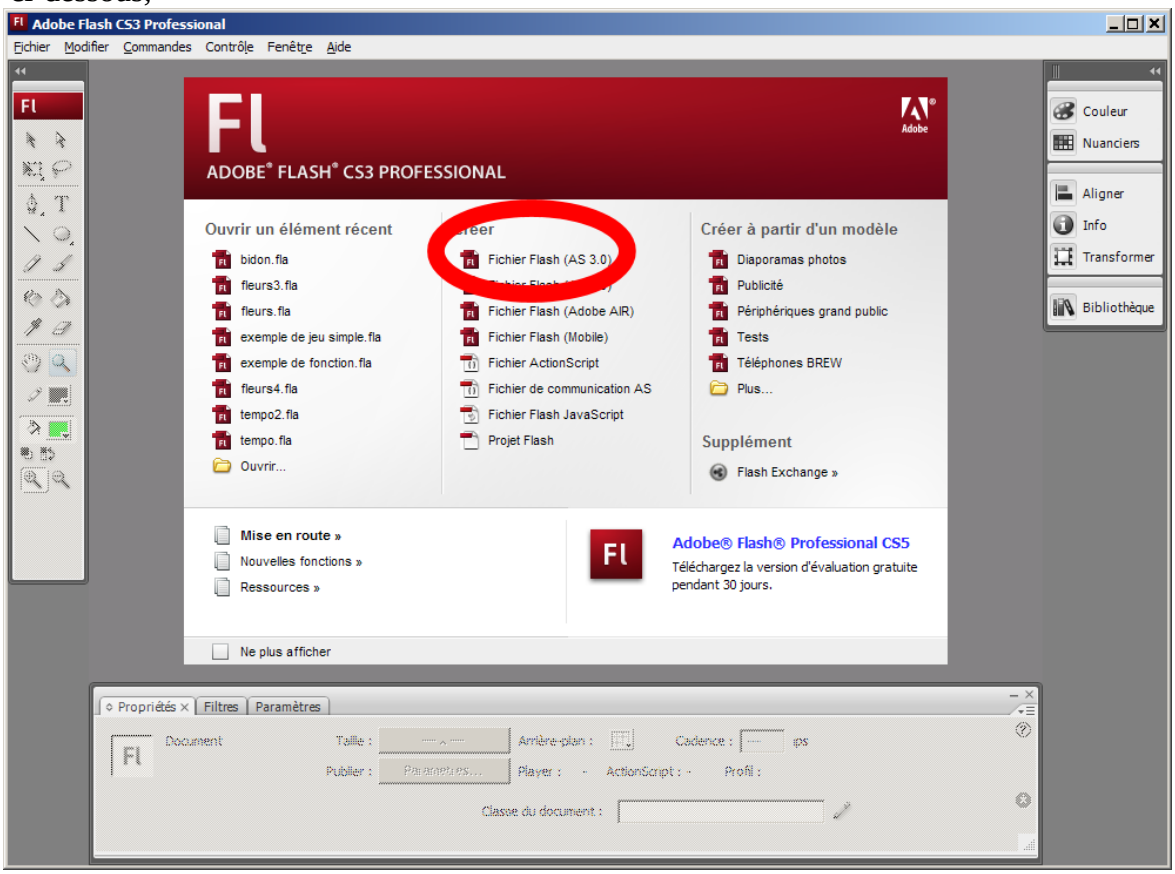


Figure 15 : créer un fichier Flash AS3 par la fenêtre d'accueil

- soit vous faites `Fichier>Nouveau...` et vous choisissez Fichier Flash (AS 3.0) comme ci-dessous.

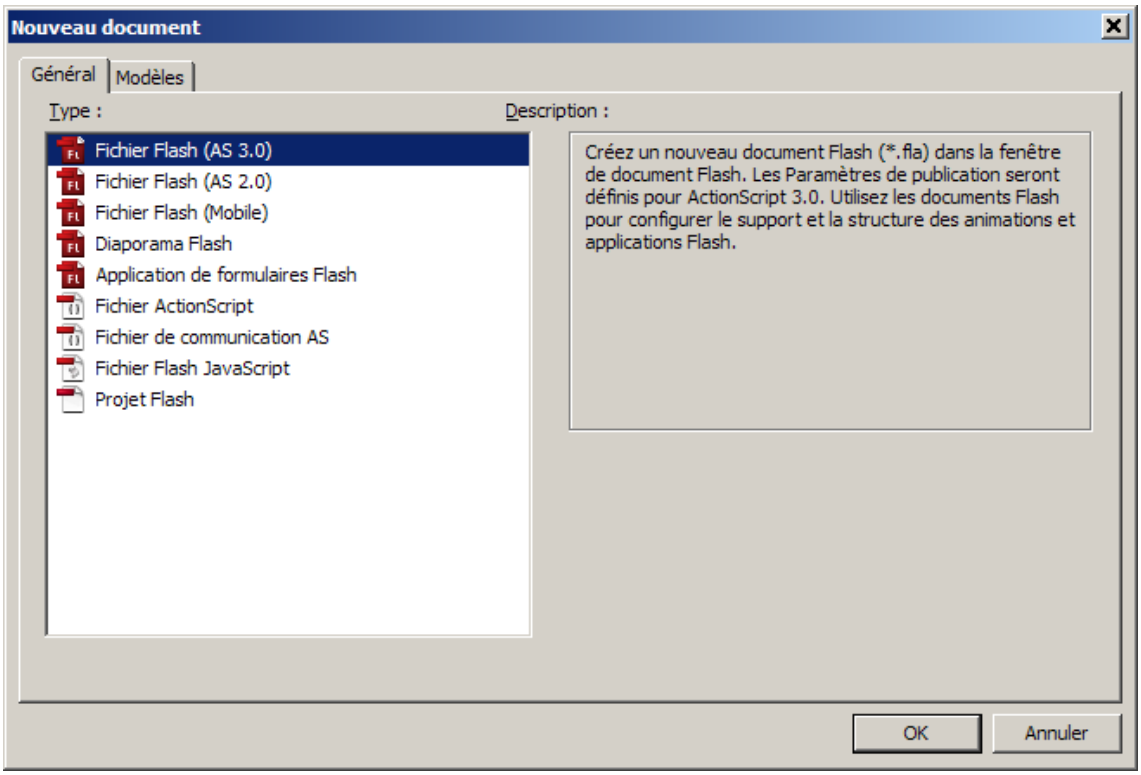


Figure 16 : créer un ficher Flash AS3 par le menu Fichier>Nouveau…

Pour écrire du code, on applique le même principe que pour dessiner, ou pour placer des occurrences de clips ; il faut une image clé pour cela !

Astuces :

1. Sachant qu'on peut mettre du code sur n'importe quelle image-clé, on peut vite se perdre dans son code. Aussi, une bonne pratique de programmation AS est de créer un calque que vous appellerez « actions », ou « script », ou « AS3 », ou tout autre nom (sachant que les deux premiers sont les plus utilisés par les Flasheurs), dans lequel il n'y aura QUE du code AS3, c'est-à-dire pas de dessin, pas de bouton, pas de masque, etc. Généralement ce calque est le premier ou le second du scénario.
2. Quand vous devez écrire du code, commencez par vérifier que vous êtes bien sur le calque « actions » et sur une image-clé, sinon faites ce qu'il faut pour cela.

Voici un exemple. Le calque du haut ne contient que des images clés vides ou des images clés avec du code (il y a un « a » au-dessus du symbole image-clé vide).

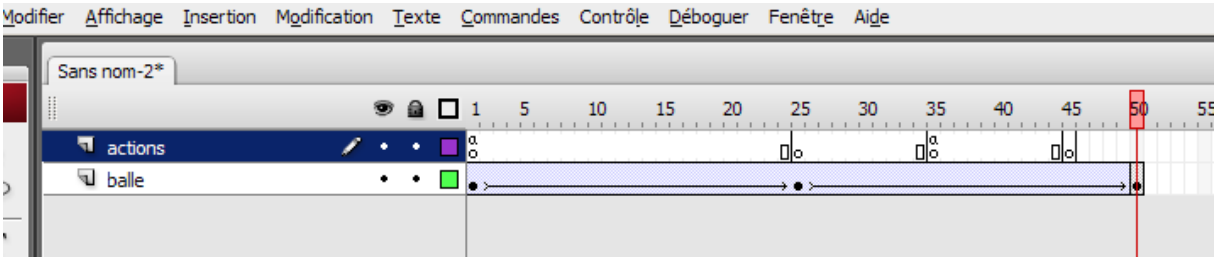


Figure 17 : exemple de calque « actions » et d'images clés avec du code ou non

Par contre, sur la Figure 18, à droite, vous pouvez voir que la dernière image-clé contient à la fois du dessin (puisqu'il y a une interpolation de forme) et du code. Le résultat sera

strictement le même dans le comportement de l'animation que si le code avait été séparé dans un calque « actions », mais il est plus facile de repérer et de corriger du code quand celui-ci est dans un calque séparé (comme à gauche dans la Figure 18). Ceci est une habitude des Flasheurs, et je vous recommande vivement de l'appliquer !

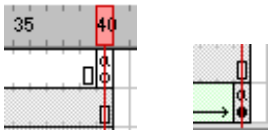


Figure 18 : indication de la présence de code Action Script dans une image

Pour voir le code qui est écrit dans une image, il faut sélectionner la cellule qui contient le « a » et ouvrir le panneau Actions (raccourci F9).

Notez pour finir que dans une image (je ne dirai plus image-clé puisque cela est obligatoire pour écrire du code) il peut y avoir autant de lignes de code que l'on souhaite.

7 Gestion de la tête de lecture

Nous allons commencer par voir comment modifier la lecture d'un scénario, que ce soit le *scénario principal* ou le *scénario d'un symbole clip ou graphique*.

Ouvrez le fichier « play, stop, goto initial.fla ». Il ne contient pour le moment qu'une simple interpolation de forme. Quand vous testez l'animation (CTRL+ENTREE), l'animation se joue en boucle, ce qui est le comportement par défaut comme vous le savez.

Nous allons maintenant faire en sorte que l'animation ne se joue qu'une seule fois. Pour cela, il faut dire à la tête de lecture que lorsqu'elle arrive sur la dernière image, elle doit s'arrêter. Comment faire cela ? Suivez les indications ci-dessous :

1. Ajouter un calque « actions »
2. Allez à l'image 100 de ce calque, créez une image clé et écrivez le code « stop() ; » comme montré sur la Figure 19.

Testez à nouveau l'animation avec CTRL+ENTREE ; l'animation s'arrête sur l'image 100.

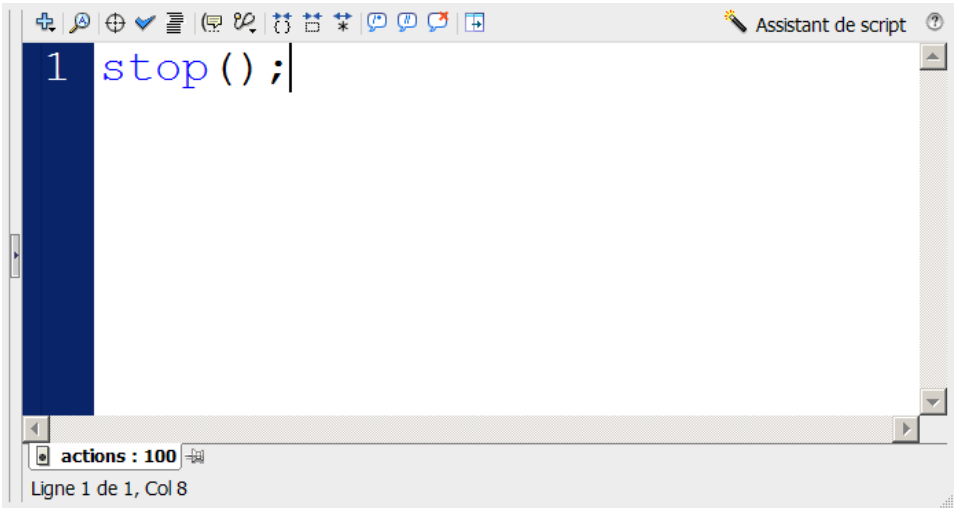


Figure 19 : code de l'image 100 du calque actions

`stop()` est une action AS3 qui dit à la tête de lecture de s'arrêter. Cela revient au même résultat que lorsqu'on appuie sur la touche ENTREE pendant la lecture d'une animation. La tête de lecture s'arrête, mais rappelez-vous que si l'image contient des clips, ceux-ci continuent à se jouer…

Nous allons maintenant ajouter d'autres images-clés dans le calque actions. Je vous donne ci-dessous les images (pensez à créer une image clé dans le calque actions chaque fois !) et le code associé, ainsi que l'explication du code. Vous trouverez dans « play, stop, goto final.fla » la version finale de cet exercice, mais elle contient des choses que vous allez découvrir un peu plus loin, donc patience…

- Image 1 :

```
gotoAndPlay(10);
```

  On demande à la tête de lecture d'aller (`goto`) directement à l'image 10 et (`And`) de continuer de lire (`Play`) l'animation depuis cette image.

- Image 10 :

```
trace("nous sommes à l'image " + currentFrame);
gotoAndPlay("milieu");
```

On affiche (`trace`) dans le panneau `Sortie` (qui s'ouvre alors automatiquement, et dont le raccourci est F2 pour le voir ou le masquer) un message indiquant le numéro de l'image courante, et on saute à l'image qui s'appelle `milieu`. Pour le moment, cette image n'existe pas. Nous allons voir plus loin comment la créer (dans la correction, elle se trouve dans le calque `labels`).
`currentFrame` renvoie automatiquement le numéro de l'image où se trouve la tête de lecture.

`trace` est une fonction qui permet de faire afficher des choses pendant l'exécution de l'animation ; c'est l'équivalent du `echo` en PHP, du `print` dans d'autres langages… C'est la meilleure façon de vérifier votre code quand vous programmez. Vous pouvez l'utiliser pour afficher des messages, des variables (nous verrons cela plus loin), etc. Elle ne s'utilise que dans Flash. Quand vous publiez l'animation en SWF, HTML ou EXE, et que vous exécutez cette animation en dehors de Flash, rien ne s'affichera. Cependant Flash exécutera ces appels de trace, ce qui est dommage car cela peut faire ralentir votre ordinateur. Heureusement, vous pouvez désactiver ces appels inutiles sans modifier votre code. Pour cela, regardez dans les paramètres de publication de votre animation (cf. un cours précédent ☺), dans l'onglet Flash vous avez une option « Omettre les actions trace ». En cochant cette option, tous les appels à `trace` seront ignorés à l'exécution.

Vous aurez certainement remarqué que l'on peut afficher plusieurs données en même temps dans un trace (plutôt que de faire deux `trace` consécutives). Pour cela, on utilise le « + » qui concatène (c'est-à-dire qui met bout à bout) les données à afficher.

- Image 50 :

```
trace("nous sommes à l'image " + currentFrame);
```

On affiche le numéro de l'image courante.

- Image 65 :

```
trace("on va de l'image " + currentFrame + " à l'image " +
(currentFrame + 5));
```

On affiche un message indiquant qu'on va de l'image courante à 5 images plus loin. Si vos ne mettez pas les () pour `currentFrame+5`, vous aurez une erreur car Flash ne comprend pas ce que vous voulez faire. Avec les (), il calcule d'abord `(currentFrame + 5)` et il concatène cette valeur au reste du message après l'avoir transformée en chaine de caractères.

- Image 70 :

```
trace("image " + currentFrame);
```

On affiche le numéro de l'image courante.

```
trace("il reste " + (totalFrames - currentFrame) + " à
parcourir");
```

`totalFrames` renvoie le nombre d'images du scénario qui contient ce code. Ici il s'agit du scénario principal, mais il pourrait tout aussi bien s'agir du scénario d'un clip, et donc on peut savoir combien d'images contient un clip ☺. Utile pour des préchargements par exemple…
`(totalFrames - currentFrame)` renvoie donc le nombre d'images à parcourir avant la fin du scénario.

- Image 100 :
  `trace("on s'arrête");`
  On affiche un message de fin.

  `stop();`
  On arrête l'animation.

Avant de tester votre animation, vous devez créer l'image qui s'appelle « milieu ». Pour cela, créez un nouveau calque, nommez-le « labels » (ou « étiquettes », ou autre). Créez une image clé à l'image 50, puis regardez les propriétés de l'image (panneau Propriétés) et nommez cette image « milieu » comme ci-dessous.

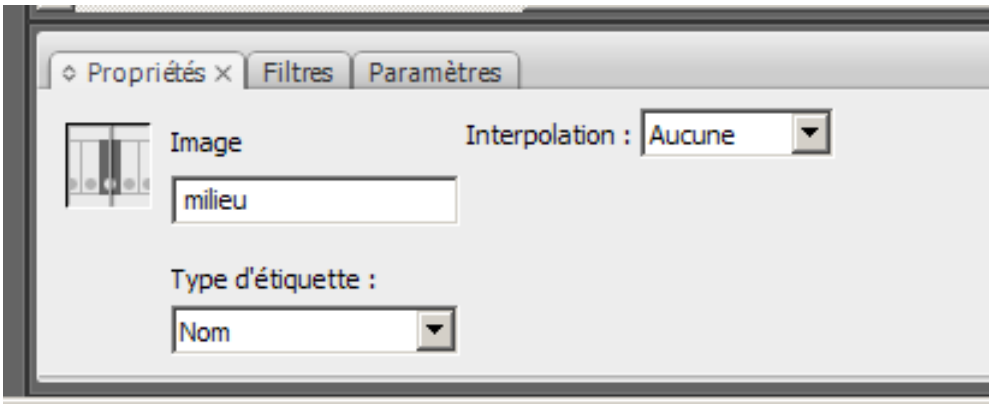
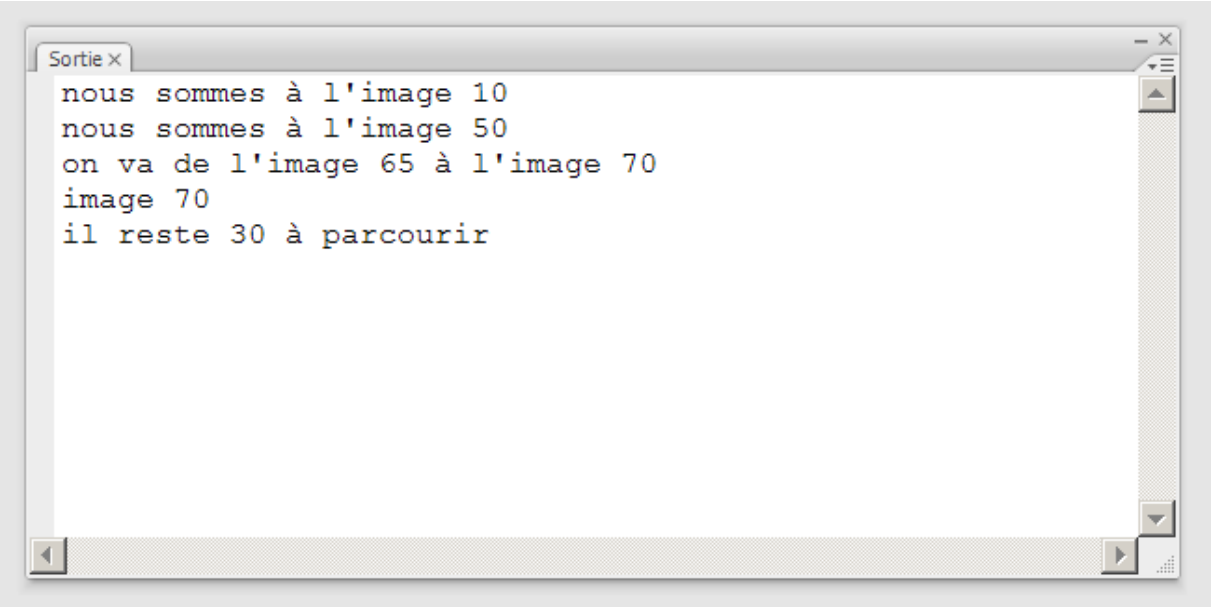


Figure 20 : propriétés de l'image qui s'appelle « milieu »

Vous pouvez à présent tester l'animation…

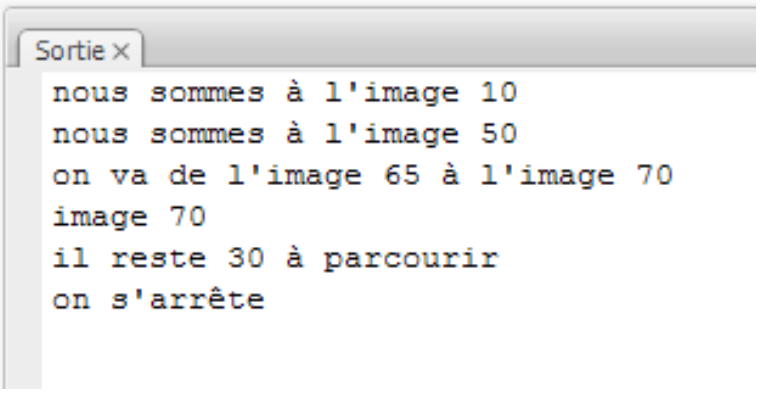
NOTE IMPORTANTE : dorénavant, quand je parlerai de tester, exécuter, lancer, etc. l'animation, il s'agira toujours de tester en faisant CTRL+ENTREE, ou bien encore Menu `Contrôle>Tester l'animation`. Il n'y a qu'avec cette solution que vous pouvez tester réellement votre code (et les clips !).

Quand vous tester l'animation, vous obtenez le résultat suivant dans le panneau Sortie.



Question : pourquoi le message de fin (à l'image 100) ne s'affiche-t-il pas ? La réponse est logique…
Lorsque Flash joue l'animation, la tête de lecture se déplace d'image en image (c'est le principe que nous avons vu jusqu'à présent pour les animations graphiques). Pour chaque image rencontrée, Flash affiche les dessins et les symboles tels que vous les avez placés sur la scène (c'est-à-dire dans le scénario). Mais en même temps, Flash regarde si l'image courante contient du code AS, et si oui, il l'exécute. Le code exécuté peut ainsi modifier l'affichage des objets présents sur la scène (nous verrons des exemples plus loin) et l'ordre de lecture des images, voir arrêter l'animation avant la fin…
Mettez-vous à la place de Flash, c'est le meilleur moyen de comprendre… Regardez à nouveau le code de chaque image. Flash commence par lire l'image 1 qui lui dit d'aller à l'image 10, qui lui dit d'aller à « milieu ». Puis Flash continue l'animation jusqu'à l'image 65 qui lui dit d'aller à l'image 70 et de s'arrêter. Donc on n'arrivera jamais à l'image 100.

Maintenant ajoutez une nouvelle ligne de code à la fin de l'image 70, et écrivez `play();`
Relancez l'animation… cette fois, vous obtenez le message de fin. L'explication est que quand la tête de lecture arrive sur l'image 70 avec le `gotoAndStop`, elle lit le code AS et l'exécute. Or, maintenant ce code lui dit de jouer (`play`) l'animation depuis cet endroit, donc la tête de lecture continue sa route, et finit par arriver à la fin du scénario où le code AS lui dit stop.



Pour résumer ce que nous venons d'apprendre :

- Ecrivez toujours votre code AS dans un calque « actions » pour le retrouver facilement.
- La tête de lecture se déplace d'image en image comme pour les animations graphiques, et elle exécute le code AS qu'elle trouve dans les images. Ce code peut éventuellement modifier le contenu d'une image et l'ordre de lecture des images.
- `play()` relance la lecture du scénario où il se trouve, et `stop()` l'arrête.
- `gotoAndPlay()` et `gotoAndStop()` envoient la tête de lecture sur une image désignée par son numéro, ou son nom, et disent à la tête de lecture de continuer la lecture depuis cette image ou de s'y arrêter. Nous verrons plus loin deux autres techniques pour ces fonctions (utilisation de nom de séquences et de variables).
- Quand vous devez écrire plusieurs lignes de code, chaque ligne se termine par un point-virgule.
- Les commentaires s'écrivent avec // ou /* */.
- On peut donner un nom à une image. Ce nom apparaît dans le scénario, et peut être utilisé en programmation AS.
- `currentFrame` et `totalFrames` renvoient le numéro de l'image courante et le nombre total d'images du scénario concerné.

Tout ce que nous venons de voir s'applique à n'importe quel scenario, que ce soit le scenario principal ou le scénario d'un symbole graphique ou d'un clip. A titre d'exemple, regardez et testez « play, stop, goto avec clip final.fla ». Alors que la balle qui rebondit est un clip, elle ne joue pas complètement son rebond. Pourquoi ? à vous de trouver la réponse… qui se trouve dans son scénario ☺

En complément de cette partie, vous pouvez également consulter l'aide pour les fonctions `prevFrame()` et `nextFrame()` très souvent utilisées.

8 Naviguer dans les scènes d'une animation

Ouvrez le fichier « multi-scènes initial.fla » et jouez l'animation. Vous constaterez que cette animation contient 4 séquences[1] qui s'enchaînent. Nous allons voir comment est structurée cette animation et comment naviguer d'une séquence à une autre.

Note : les boutons présents dans l'animation sont inutiles pour le moment. Rien ne se passe si vous cliquez dessus, c'est normal. Nous verrons plus tard comment les programmer.

8.1 Animation multi-séquences

Pour afficher la liste des séquences composant une animation, vous pouvez soit afficher le panneau `Séquence` (menu `Fenêtre> Autres Panneaux>Séquences`), soit cliquer sur le bouton associé aux séquences sous le scénario (cf. Figure 21).

Vous devez savoir que dès lors qu'il y a au moins deux séquences dans une animation, Flash les lit et les enchaîne automatiquement dans l'ordre où elles apparaissent dans ces listes. Ce comportement peut être modifié grâce à la programmation, ce que nous allons faire juste après…

Sachez enfin que :

- Pour *modifier l'ordre* des séquences, faites des drag&drop (glisser-déposer) dans le panneau Séquence.
- Pour *ajouter une séquence*, cliquez sur le bouton « + » dans le panneau Séquence, ou passez par le menu `Insertion>Séquence`.
- Pour *supprimer une séquence*, utilisez le bouton « poubelle » dans le panneau Séquence.
- Pour *renommer une séquence*, faites un double-clic sur son nom dans le panneau Séquence.

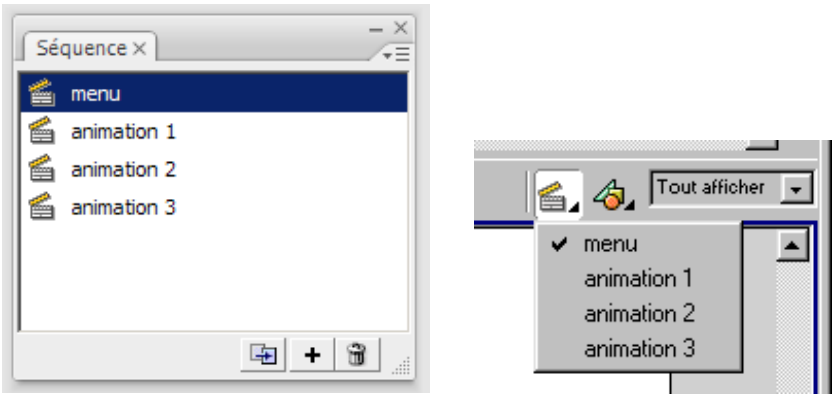


Figure 21 : Affichage de la liste des séquences

Dès que vous cliquez sur le nom d'une séquence (dans les listes ci-dessus), le scénario affiché est celui de lé séquence choisie.

[1] Jusqu'à la version Flash 4, les séquences s'appelaient « scènes ».

8.2 Naviguer dans les séquences

Vous allez commencer par dire à Flash de lire chaque séquence en boucle (sauf la première). Pour cela, vous allez ajouter un `gotoAndPlay(1)` à la fin des séquences « `animations` » 1, 2, et 3. Vous savez faire cela maintenant… Vous pouvez également faire un `gotoAndPlay("début")` par exemple. Ca aussi, vous savez le faire à présent.

Juste pour nous exercer, nous allons maintenant demander à Flash de ne pas lire la séquence « `Menu` », mais d'aller directement à l'image 15 de la séquence « `animation 2` » et de s'y arrêter.
Insérez le code ci-dessous à l'image 1 de la séquence « `menu` » :

```
gotoAndStop(15,"animation 2");
```

Simple, non ? On utilise le `gotoAndStop` vu auparavant, et il suffit de préciser ensuite le nom de la séquence. Le nom de la séquence doit être écrit entre guillemets (comme toutes les chaines de caractères en Flash), et vous devez impérativement respectez la casse. Si vous écrivez `"Animation 2"` au lieu de `"animation 2"`, vous aurez le message d'erreur suivant (vous devez savoir décoder les messages d'erreur par vous-mêmes…) :

```
ArgumentError: Error #2108: La séquence Animation 2 est introuvable.
    at flash.display::MovieClip/gotoAndStop()
    at multi_fla::MainTimeline/frame1()
```

Maintenant, remplacez le `gotoAndStop(15,"animation 2");` par `gotoAndPlay(15,"animation 2");` et relancez l'animation. Cette fois-ci, l'animation 2 démarre à l'image 15 comme avant, mais elle se lit en boucle puisque vous avez ajouté un `gotoAndPlay(1)` à la fin de la séquence.

Vous pouvez à présent naviguer simplement entre les séquences d'une animation Flash. Quand nous aurons vu comment programmer les boutons, cela prendra tout son sens…

9 Propriétés de clips

Jusqu'à présent, nous n'avons utilisé les clips que pour les animations graphiques (sauf un petit exemple de code au début de ce cours). Or, les clips sont une des bases fondamentales en programmation AS. Dans quelques temps, nous les programmerons pour qu'ils aient des comportements, de même que les boutons. Mais avant cela, nous allons voir comment modifier leurs propriétés…

Chaque occurrence de clip possède des *propriétés* qu'il est possible de manipuler par programmation.

> **Note** : dorénavant je parlerai simplement de clip et non plus d'occurrence de clip car avec AS on ne travaille que sur les occurrences des symboles.

Pour accéder à une propriété d'un clip, on écrit « `monClip.propriété` », où `monClip` est le *nom d'occurrence* d'un clip.

Qu'est-ce que le nom d'occurrence ? Il s'agit du nom que vous donnez à une occurrence de symbole (clip, bouton, graphique). Chaque nom est unique ; vous ne pouvez pas donner le même nom à deux occurrences présentes sur la scène même si l'une est un bouton et l'autre un clip.

Pour donner un nom à une occurrence, procédez comme pour donner un nom à une image ; sélectionnez une occurrence sur la scène, et tapez un nom dans le panneau Propriétés. Vérifiez bien que vous donnez un nom à l'occurrence et non pas à l'image !!! La Figure 22 donne un exemple de ce que vous devez voir (à opposer à la Figure 20 qui montre le nom d'une image).

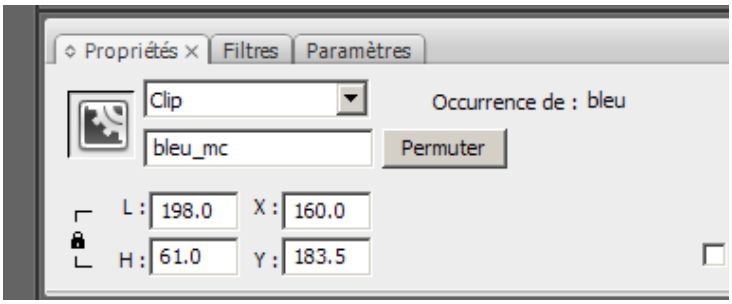


Figure 22 : le clip s'appelle « bleu_mc » et est une occurrence du symbole « bleu » de la bibliothèque

9.1 Propriétés de base

Flash fournit des propriétés de base pour les clips. Voici ci-dessous les plus courantes avec leur signification, les unités utilisées et les valeurs possibles (cf. Tableau 2).

Nom	Signification	Unité	Valeurs mini/maxi
width	Largeur	pixel	Aucune
height	Hauteur	pixel	Aucune
x	Position horizontale	pixel	Aucune
y	Position verticale	pixel	Aucune
scaleX	Echelle horizontale	Nombre	Entre - l'infini et + l'infini (Utiliser le chiffre 1 pour une échelle à 100%)
scaleY	Echelle verticale	Nombre	Entre - l'infini et + l'infini (Utiliser le chiffre 1 pour une échelle à 100%)
rotation	Rotation	degré	Aucune
alpha	Transparence	Nombre	Entre 0 et 1 : Ex. Utiliser 0.5 pour régler une

			occurrence à 50% de transparence.
visible	Visibilité	Booléen	true ou false

Tableau 2 : quelques propriétés de base pour les clips

Notes :

- Les propriétés `scaleX` et `scaleY` peuvent prendre des valeurs négatives pour effectuer une symétrie (Exemple : `carre1.scaleX=-1`).
- La propriété `rotation` peut être réglée à n'importe quelle valeur que Flash convertira en une valeur comprise entre -180 et +180.
- Les propriétés `x` et `y` peuvent être réglées avec des valeurs dépassant les limites de la scène (valeurs négatives et valeurs plus grandes que la largeur et la hauteur de la scène) afin de faire disparaitre une occurrence.
- En informatique, et donc en Flash, le point (0,0) est en haut à gauche de la scène. Les X augmentent quand on va vers la droite, et les Y augmentent quand on descend (contrairement à ce que vous avez appris en mathématiques, donc il faudra parfois adapter votre code en ce sens).

9.2 Exemple

Ouvrez le fichier « propriétés de base.fla ». Vous voyez une occurrence de clip en haut à gauche de la scène, et en position horizontale. Testez l'animation ; le clip apparaît maintenant à une autre position, tourné de 90°, et un assez transparent. C'est grâce aux propriétés !

Voici le code associé (image 1). A ce propos, c'est la dernière fois que je vous dis où est le code. A présent, vous devez être capable de trouver le code dont je parle.

```
bleu_mc.x=300;
bleu_mc.y=200;
bleu_mc.rotation=90;
bleu_mc.alpha=0.2;
```

Explications :

- Le clip s'appelle `bleu_mc` (ça, je ne vous le dirai plus).
- On modifie la position du clip en le plaçant à x=300 et y=200. On aurait pu faire cela aussi par le panneau Propriétés de l'occurrence..
- On modifie l'angle de rotation du clip. Le sens horaire correspond à des angles positifs, et le sens anti-horaire à des angles négatifs. Autrement dit, quand on écrit rotation=90, cela revient à faire faire un quart de tour dans le sens des aiguilles d'une montre. Vous pouvez le vérifier avec le clip qui tourne autour de son point d'alignement visualisé par un gros point noir.
- On modifie enfin la transparence du clip.

Important :
Flash met à jour l'affichage UNIQUEMENT que lorsqu'il change d'image. Aussi, si dans l'image vous écrivez :

```
bleu_mc.x=100;
bleu_mc.x=200;
bleu_mc.x=300;
bleu_mc.x=400;
```

vous ne verrez que la dernière position affichée ; vous ne verrez pas le clip se déplacer petit à petit à l'écran comme on pourrait l'espérer. Cela implique donc que pour faire avancer un clip à l'écran, il faudra utiliser une autre technique. Nous verrons cela plus tard.

A présent, vous pouvez tester l'autre partie du code (en mettant la première en commentaire si on veut rester logique) :

```
bleu.x = Math.random() * stage.stageWidth;
bleu.y = Math.random() * stage.stageHeight;
bleu.rotation = Math.random() * 360;
bleu.alpha = Math.random();
```

Explications :

- `Math` est un objet (une classe pour être exact) fourni par Flash et qui contient tout un ensemble de fonctions mathématiques (sinus, tangente, racine carrée…). Si vous tapez `Math.`, Flash vous affichera toutes ces fonctions. Après… merci l'aide de Flash pour le détail ☺.
- `Math.random()` renvoie un nombre aléatoire compris dans [0,1[, c'est-à-dire au mieux 0.99999999999. Cette fonction est très utilisée pour les jeux.
- `stage` est un objet fourni par Flash et correspond à la scène (stage=scène en anglais). Il possède également des propriétés dont `stageWidth` et `stageHeight` qui sont respectivement la largeur et la hauteur de la scène (en pixels).

`Math.random() * stage.stageWidth` renvoie donc un nombre aléatoire compris entre 0 et la largeur de la scène, par exemple 245.7865. Flash convertit cette valeur en nombre entier (les dixièmes de pixel n'existent pas !) quand on l'affecte à une position. Si on veut être très propre dans sa programmation (ce que je vous conseille), on écrira plutôt : `Math.floor(Math.random() * stage.stageWidth)` ce qui a pour effet de ne garder que la partie entière du nombre.

9.3 Autre exemple

Testez « propriétés de base 2.fla ». Vous y verrez par exemple l'incrémentation de valeurs de propriétés (par exemple le clip qui grossit toujours plus)…

9.4 Propriétés personnelles

Flash vous offre la possibilité de créer vos propres propriétés. Cela est très pratique ! Regardez et testez « fleurs.fla ». Les rectangles verts représentent des fleurs et des plantes qui poussent chacune à sa vitesse. Pour cela, j'ai utilisé du code du type :

```
fleur1.vitesseCroissance = 1 + Math.random() * 25;
```

`vitesseCroissance` n'existe pas en Flash (le code n'est pas bleu). En écrivant cela, Flash a automatiquement créé une propriété `vitesseCroissance` pour l'occurrence `fleur1`. On manipule cette nouvelle propriété comme une propriété de base.
Si je crée une nouvelle occurrence de fleur, celle-ci n'aura pas de propriété `vitesseCroissance`.

Pourquoi ai-je utilisé 3 images clés ?

- Sans l'image 3, l'animation boucle sur les deux premières images, donc on repasse sans cesse sur l'image 1 qui tire chaque fois au sort un nouveau nombre pour la croissance.

Vous pouvez tester cela en « décommentant » le `gotoAndPlay(1)` de l'image 2. Dans ce cas, les plantes et fleurs changent en permanence de vitesse de croissance. Effet intéressant…

- Sans l'image 3, et en écrivant `gotoAndPlay(2)` à la place du `gotoAndPlay(1)`, cela répondrait à mon objectif. Malheureusement Flash ne réexécute pas le code car faire un goto(2) alors qu'on y est déjà n'est pas logique pour lui. Il faut donc ajouter une autre image et faire le goto(2) depuis cette image, d'où la 3ème image.

Cette notion de propriétés personnelles est intéressante, mais vous comprenez que si j'ai 25 fleurs à l'écran, je vais devoir écrire 25 fois du code tel que « fleurX.vitesseCroissance = … », ce qui est long à écrire, lourd à lire et surtout TRES embêtant quand on doit changer le comportement de toutes les fleurs.
Comment faire mieux ? Regardez « fleurs en clips.fla ». Sur la scène principale, il n'y a que des occurrences de fleurs et une interpolation de forme. Aucun code AS ici ! Tout le code est dans le symbole « fleur autonome ». Vous allez y retrouver quasiment le même code et le même principe que précédemment. Du scénario principal précédent et son code AS qui manipule des occurrences, je suis passé au scénario du clip qui manipule une occurrence qu'il contient.
Maintenant, quand je crée une occurrence de « fleur autonome », celle-ci contient implicitement son code AS et donc manipule l'occurrence « uneFleur » qu'elle contient. Voyez-vous toute la puissance de ceci ? Par exemple, si demain on veut que toutes les plantes croissent moins vite, on ne change que la ligne « uneFleur.croissance = Math.random()*25; » dans l'image 1 du symbole clip « fleur autonome » et toutes les fleurs subiront ce changement. Avec la solution précédente, on aurait du changer chaque ligne fleurX.vitesseCroissance = … de chaque fleur. Quand il y a 200 fleurs, vous imaginez le gain de temps et la facilité de travail pour les mises à jour… Nous verrons plus tard comment créer automatiquement 200 fleurs par programmation…

10 Les variables

Pour conclure cette première partie, faisons un petit détour par les variables. Si vous avez déjà travaillé en PHP, Javascript ou autre langage, vous savez sans doute ce qu'est une variable… Toutefois, faisons un petit rappel/résumé, et voyons comment cela se passe avec Flash.

De façon générale en informatique, une *variable* est une sorte de boite (en fait une zone dans la mémoire vive, la RAM, de votre ordinateur) qui *possède un nom, un type et un contenu*. Par exemple, la variable « âge » a pour nom « âge », un type numérique et son contenu peut être « 10 », « 25 », « 90 », mais aussi « 17.5 », « 150.56 », ou bien encore « 4 000 000 000 ». Le contenu (mais pas le type) d'une variable peut changer au cours du temps, pendant l'exécution d'une animation par exemple. Chaque fois qu'on stocke un contenu dans une variable, on efface l'ancien contenu.

Les valeurs de « âge » données ci-dessus sont des valeurs numériques. Implicitement, « âge » fait penser que nous allons stocker un âge dans la variable, et donc que la valeur sera celle d'un âge… mais l'âge de quoi ? de qui ? Si c'est pour un être humain, cela sera de 0 à 150 ans, mais si c'est pour une planète cela peut être 4 Milliards d'années ! C'est pourquoi en informatique, d'une part on nomme intelligemment les variables (par exemple `agePersonne`, `agePlanète`…) et d'autre part on associe un *type* à une variable. Cela permet d'optimiser les calculs pour le processeur, d'aider Flash pour l'exécution voire pour trouver des erreurs, de faire certaines vérifications pour le programmeur, et de définir clairement le type des valeurs que l'on s'attend à trouver dans la variable.
Avec AS3, il est fortement conseillé de préciser le type de la variable.

Il existe deux grandes catégories de variables en Flash : les variables *locales* et les variables *de scénario (*ou *globales).* Nous verrons les premières dans quelques temps, avec les fonctions. Les variables de scénario sont tout simplement des variables que l'on déclare dans les images d'un scénario. A partir du moment où elles sont déclarées, elles sont connues dans toutes les images du scénario courant, mais aussi dans toutes les séquences de l'animation courante.

Pour déclarer une variable, on écrira :

```
var maVariable:type = valeur ;
```

Ou

```
var maVariable:type ;
```

La première solution crée la variable en lui donnant son nom, lui associe un type, et lui affecte une valeur. La seconde solution ne lui affecte pas de valeur. Dans ce cas, pour associer plus tard une valeur à la variable, on écrira `maVariable=valeur;`

Testez le fichier « les variables.fla ». Vous y trouverez le code ci-dessous.

```
/* La variable unNom est de type chaine de caractères (String) et n'a pas
de valeur pas défaut.*/
var unNom:String;
trace(unNom);//--> renvoie null, c'est-à-dire la valeur par défaut d'une
variable de type chaine de caractères

unNom="Master IPM";
trace(unNom);//-->Master IPM
```

```
/* La variable unNom est de type chaine de caractères (String) et n'a pas
de valeur pas défaut.*/
var unAutreNom:String = "Jean-Claude";
trace("Bonjour " + unAutreNom); //--> Bonjour Jean-Claude

/* si vous enlevez le commentaires pour les 2 lignes de code ci-dessous,
vous obtiendrez une erreur à l'exécution. Faites
l'essai pour voir l'erreur et comprendre ce qui ne va pas. */
//unNom=10;
//trace(unNom);

/* Les lignes ci-dessous correspondent à une erreur typique en Flash au
début. On a déclaré une variable sans lui associer
de type, et donc Flash fait de son mieux pour comprendre ce que vous voulez
mettre dedans.
Au début, on met 25, donc Flash associe implicitement le type numérique
(Number) à cette variable.
On peut donc écrire trace(age+10) qui affichera 35*/
var age=25;
trace(age); //-->25
trace(age+10); //-->35
/* Maintenant on décide de stocker une chaine de caractères. Flash associe
alors à présent le type String à cette variable.*/
age="toto";
trace(age);//-->toto
/* De même si on met à présent "25" et non plus 25, la variable est de type
String, donc "25"+10 est interprété par Flash
comme "25" + "10", soit "2510" alors que vous pensiez peut-être voir
35...*/
age="25";
trace(age+10); // --> 2510

/* La variable autreAge est de type int (c'est-à-dire integer, ce qui
signifie nombre entier). Quand on stocke
une valeur décimale, Flash ne garde que la partie entière du nombre.*/
var autreAge:int;
autreAge=33.46;
trace(autreAge); //-->33
```

Voici quelques uns des types de variables fournis en Flash :

- String : chaine de caractères
- Number : toute valeur numérique, y compris les valeurs avec ou sans décimale
- int : un nombre entier (sans décimale). ATTENTION, pas de majuscule dans le nom contrairement aux autres types !
- Boolean : une valeur booléenne qui vaut vrai ou faux (true ou false),
- Array : tableau de valeurs
- MovieClip : une occurrence de clip
- TextField : un champ de texte dynamique ou de saisie
- SimpleButton : un symbole de bouton
- Date : une date ou une heure
- etc…

Sachez qu'il est possible aussi de créer ses propres types de données, par exemple des types de structures contenant des données de types différents. Ainsi le type `Personne` pourra contenir un nom (`String`), un âge (`int`), des numéros de téléphone (tableaux de String), etc. Ceci est extrêmement courant en informatique. Avec Flash, cela se traduira par l'écriture

de Classes au travers de fichiers ActionScript (extension « .as »). Nous verrons cela à la fin du cours (plutôt pour informaticien).

Vous trouverez un autre petit exemple de variables dans « les variables 2.fla » dont voici le code :

```
// on mémorise la position y à atteindre pour ne pas la recalculer pour
chaque occurrence.
var positionVisée:int = stage.stageHeight/2;

// on affecte cette position à toutes les occurrences de la scène
bleu1.y=positionVisée;
bleu2.y=positionVisée;
bleu3.y=positionVisée;
poly1.y=positionVisée;
poly2.y=positionVisée;
```

Une dernière chose… Tout ce que vous découvrez dans ce cours se cumule au fur et à mesure. Pensez-y !!! Par exemple, nous avons vu gotoAndPlay(10) et gotoAndPlay("milieu") au début de cette partie. Maintenant que vous avez vu les variables, il est possible d'écrire du code comme :

```
var prochaineImage:int;
prochaineImage=currentFrame+ Math.random()*totalFrames ;
gotoAndPlay(prochaineImage);
```

ou encore

```
var prochaineImage:String;
var prochaineSéquence:String;
prochaineImage=10;
prochaineSéquence="fin";
gotoAndPlay(prochaineImage,prochaineSéquence);
```

Voyez-vous l'intérêt de ce genre de code ?

A bientôt pour la suite…