Avant-propos

Pourquoi un tel ouvrage ?

Après les quatre premières versions du logiciel Flash, c'est en 2000, à la sortie de Flash 5, que l'ActionScript 1 fait réellement son apparition. Avant cette date, un système de scripts très simple permettait d'ajouter quelques fonctions d'interactivité élémentaires telles que le déplacement de la tête de lecture du scénario, le chargement d'animations et de pages web, ainsi que la manipulation des variables.

Quand l'AS1 a commencé à se faire connaître des webmasters, nous étions non seulement loin d'imaginer que ce langage allait réellement devenir un standard, mais nous ne savions pas non plus qu'il allait passer par de telles étapes d'évolution. En effet, un langage abouti est avant tout un langage orienté objet. Sans entrer dans les détails, des langages tels que C, Java ou encore PHP proposent une logique de construction de programmes évolués, fondée sur le principe de « blocs de lignes d'instructions » réutilisables, et c'est là que réside toute la difficulté.

Lorsque Flash MX 2004 propose l'ActionScript 2.0 en 2003, on assiste à l'arrivée de la POO (Programmation orientée objet). À partir de cette date, les utilisateurs Flash se scindent en deux catégories : ceux qui utilisent Flash et l'AS2 en tant que langage de scripts, et ceux qui développent en AS2, en objet. De toute l'histoire du logiciel, c'est la première fois qu'une division des utilisateurs est constatée.

Cette cohabitation subsiste pendant de nombreuses années jusqu'à la sortie en 2008 de Flash CS4 (Flash 10), couplée à celle de l'AS3. Apparaît alors une nouvelle famille d'utilisateurs : outre les développeurs AS2 qui utilisent le langage en POO et ceux qui codent en mode procédural (séquentiel), il y a désormais les nouveaux développeurs AS3.

Cette troisième mouture du langage est présentée par Adobe (et par la communauté des développeurs AS) comme une version complètement orientée objet. Non seulement la syntaxe diffère de celle de l'AS2, mais elle est encore plus orientée objet. C'est à partir de cette année 2008 qu'un grand nombre d'utilisateurs pensent que l'AS3 n'est plus fait pour eux puisque toutes les publications papier et web qui s'y réfèrent le considèrent comme un langage complètement orienté objet. Mais en tant que développeur AS, et non plus en tant qu'auteur, je découvre lors de l'apprentissage de cette nouvelle version qu'il est possible de développer en mode séquentiel. Je décide alors de publier un ouvrage dédié à l'AS3, *ActionScript 3 – Programmation séquentielle et orientée objet*, qui fasse référence à des scripts écrits en mode séquentiel et en mode orienté objet.

Comme vous pourrez le découvrir dans le premier chapitre de ce livre, il existe plusieurs approches de construction d'une interface. Même s'il est vrai que la construction dynamique est la solution la plus efficace, il faut reconnaître que cette méthode sous-entend qu'un utilisateur Flash doit très bien connaître l'AS et s'affranchir de l'interface du logiciel ! Dans ce cas, qu'en est-il de ceux qui construisent les écrans d'un programme à l'aide de la fenêtre Scénario de Flash ?

Durant ces deux dernières années, lorsque j'ai eu à enseigner l'AS3 dans différents établissements français (GOBELINS, l'école de l'image, l'université Paris 1 Panthéon-Sorbonne, l'université Rennes 2, l'ESIEE, etc.), j'ai toujours pensé que la seule approche du langage était une approche dynamique. Mes étudiants et stagiaires ont donc appris à :

- limiter le nombre d'images-clés (en essayant de n'en avoir qu'une seule) ;
- avoir très peu de symboles (en favorisant les chargements de fichiers externes).

En mai 2009, Olivier Poncer de l'ESADS (ESAD de Strasbourg) me demande d'intervenir pour diriger un atelier de quatre jours dans la section « Didactique visuelle ». En entrant dans la classe, j'ai alors rencontré treize étudiants qui avaient déjà terminé le travail de construction de l'interface de leur animation Flash. La *timeline* (la fenêtre Scénario) de chaque projet comportait de nombreuses images et images-clés, la palette Bibliothèque contenait de nombreux symboles, et il ne restait plus qu'à ajouter la couche d'interactivité ! À cet instant, j'ai réalisé que mon approche habituelle du code dans une animation n'allait pas être adaptée. Il m'aura alors fallu deux ou trois heures, en découvrant précisément la construction des premières interfaces qui m'étaient présentées par les étudiants, pour comprendre que j'allais devoir utiliser une logique de programmation en AS3 comparable à celle que j'utilisais en AS1/AS2. Après ces quatre jours passés à accompagner ces étudiants pour ajouter de l'interactivité à leur projet, j'ai réalisé que le code déployé

pour eux ne ressemblait pas du tout au code que j'avais enseigné au cours des deux années précédentes, mais qu'il était plus abordable et compréhensible.

Même si cette logique de construction d'un programme (en AS3) n'est pas conseillée pour de gros projets, elle demeure utilisable et constitue même la seule approche possible pour ceux qui ont « peur » du code et/ou de l'AS3 en particulier. Elle convient également à tous ceux qui souhaitent ajouter ponctuellement de l'interactivité dans une animation.

Ce constat auquel je suis parvenu après quatre jours passés dans cette classe de l'ESADS m'a donné l'idée d'écrire ce livre.

Aide à la lecture

Pour vous aider dans la lecture de cet ouvrage, nous avons balisé les différents fichiers d'exemples et d'exercices à l'aide d'en-têtes qui se présentent de la manière suivante.

Fichier de travail
Nom : `ClipAnimControle1.fla`
Difficulté : ● ● ● ● ●
Observations : vous devez connaître la technique de navigation au sein d'une animation Flash.

La première information précise le nom du fichier Flash ou de l'archive au format ZIP qui permet d'étayer le propos ou de résoudre un exercice. Tous ces fichiers de travail peuvent être téléchargés à l'adresse http://www.yazo.net/eyrolles.

La difficulté est estimée sur une échelle de 1 à 5. Un seul point jaune signifie que vous ne devriez rencontrer aucun problème de compréhension, alors que cinq points correspondent à une notion délicate ou à un exercice difficile.

Enfin, les observations vous indiquent les prérequis nécessaires pour aborder le point traité ou l'exercice à réaliser. Elles fournissent également diverses informations pour vous aider dans votre apprentissage.

Par ailleurs, lorsque nous apporterons des explications à une notion ou lorsque nous développerons une idée, nous ferons référence à cette dernière en utilisant le terme « développement ».

Partie I

Premiers pas en ActionScript

Lorsque vous travaillez dans Flash, vous pouvez utiliser différentes approches pour ajouter de l'interactivité à une interface, comme vous le découvrirez au cours du chapitre 1, « Préparer l'interface et les médias ». D'une manière générale, cette première partie est consacrée à l'approche la plus simple, aucune connaissance préalable en programmation n'est donc nécessaire pour aborder les pages qui suivent.

Néanmoins, toutes les explications des chapitres 1 à 6 s'articuleront autour d'un même postulat : vous avez construit l'interface d'une animation grâce à de simples glisser-déplacer d'occurrences (de symboles) sur la scène et vous souhaitez ajouter de l'interactivité à votre réalisation.

Chapitre 1
Préparer l'interface et les médias

Avant de nous lancer dans le code à insérer dans une animation Flash, nous allons essayer de comprendre comment construire notre interface (à l'aide des palettes Scénario et Bibliothèque) et préparer nos médias.

Symbole et occurrence

Avant même d'évoquer la notion de programmation, il est important de bien expliquer la différence entre une occurrence et un symbole.

Un grand nombre d'utilisateurs de Flash confondent ces deux termes et c'est parfois la raison pour laquelle ils ne comprennent pas certaines explications. Pour simplifier celles relatives à ce développement, voici quelques points importants à prendre en compte.

- Lorsque vous transformez une forme ou une sélection présente sur la scène en un symbole (via le raccourci clavier F8 et l'exécution de la commande Convertir en Symbole du menu Modification), vous créez et placez ce dernier dans la bibliothèque.
- Lorsque vous saisissez un symbole qui ne peut se trouver par définition que dans la bibliothèque et si vous le placez sur la scène, vous obtenez une occurrence.
- Une occurrence est donc la représentation graphique d'un symbole.

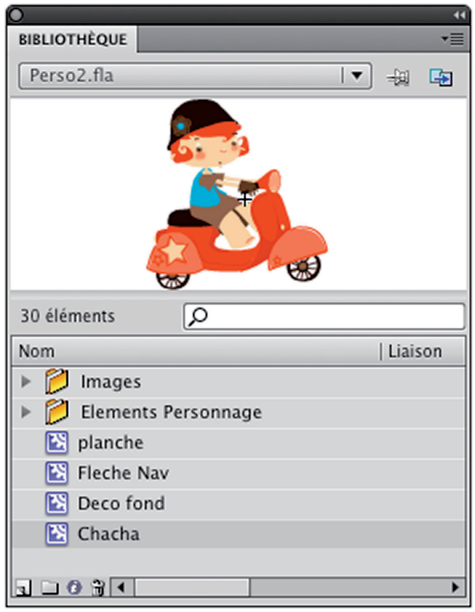

Figure 1-1
Tous les symboles de la bibliothèque peuvent à présent être placés sur la scène : vous allez obtenir des occurrences comme le montre la figure 1-2.

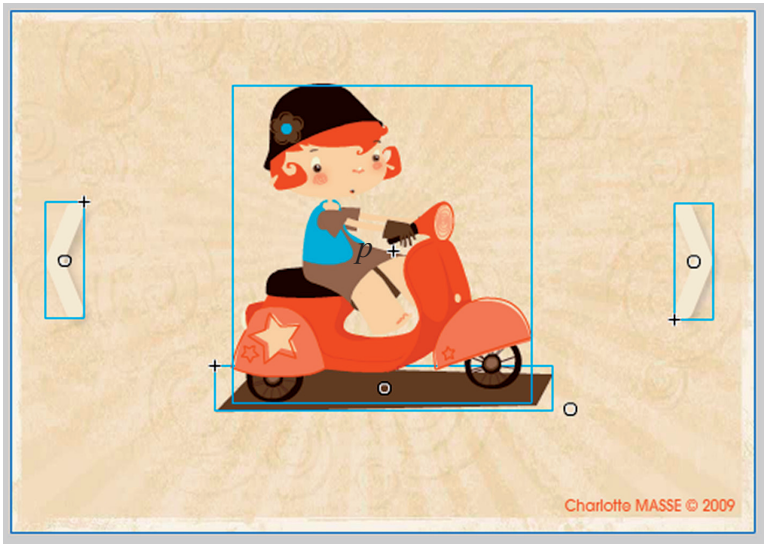

Figure 1-2
Lorsqu'une occurrence est sélectionnée, un cadre (bleu par défaut) l'entoure pour indiquer la place occupée et l'état de sélection.

- Une occurrence ne peut se trouver dans la bibliothèque et un symbole n'est jamais sur la scène.
- Si vous supprimez un symbole de la bibliothèque, les occurrences qui y sont rattachées le sont également.
- La suppression d'une occurrence n'entraîne pas la suppression du symbole auquel elle est rattachée.
- Vous pouvez obtenir plusieurs occurrences issues du même symbole, mais une occurrence n'est jamais rattachée à plusieurs occurrences.
- Une occurrence est toujours rattachée à un seul symbole.

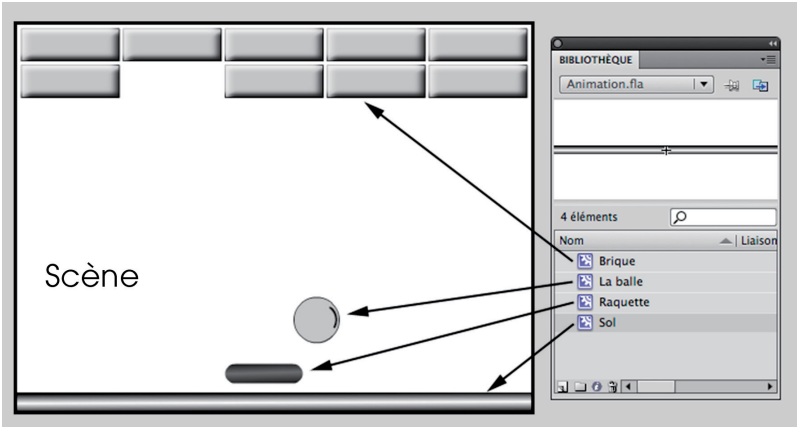

Figure 1-3
Un symbole (ex. : Brique) peut être placé plusieurs fois sur la scène. On obtient alors différentes occurrences issues du même symbole.

Maintenant que vous faites bien la distinction entre occurrence présente sur la scène et symbole de la bibliothèque, voyons comment nommer une occurrence.

Comment nommer les occurrences ?

Comme nous venons de le voir, lorsqu'un symbole est placé sur la scène, on obtient une occurrence issue de ce symbole. La construction d'une animation se traduit donc par le positionnement de plusieurs occurrences sur la scène.

Il est important que vous compreniez dès à présent que ce sont les occurrences que nous allons utiliser comme :

- zone de clic (ex. : un bouton, un objet que vous déplacerez sur la scène, une image cliquable, etc.) ;
- élément graphique à contrôler (ex. : un personnage qui se déplace sur la scène, une barre de menus qui entre ou sort de la scène, etc.).

Pour faire la distinction entre deux occurrences, nous devons donc les nommer. Voici comment procéder.

1. Sélectionnez la flèche noire dans la palette d'outils (ou appuyez sur la touche V de votre clavier).
2. Effectuez un clic sur l'occurrence que vous souhaitez nommer.
3. Cliquez dans le champ Nom de l'occurrence de la palette Propriétés (figures 1-4 et 1-5).
4. Donnez un nom à l'occurrence.

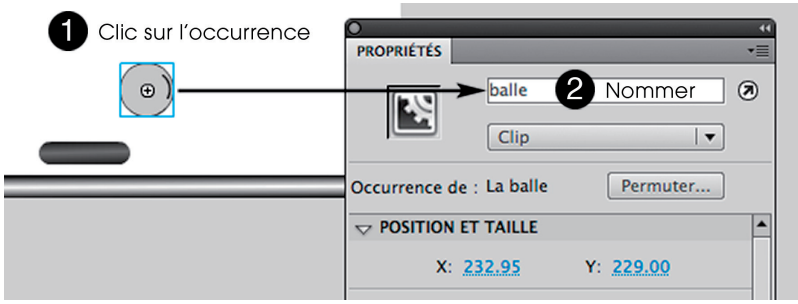

Figure 1-4
Pour nommer une occurrence, il suffit de cliquer dessus pour la sélectionner, puis de renseigner le champ Nom de l'occurrence de la palette Propriétés.

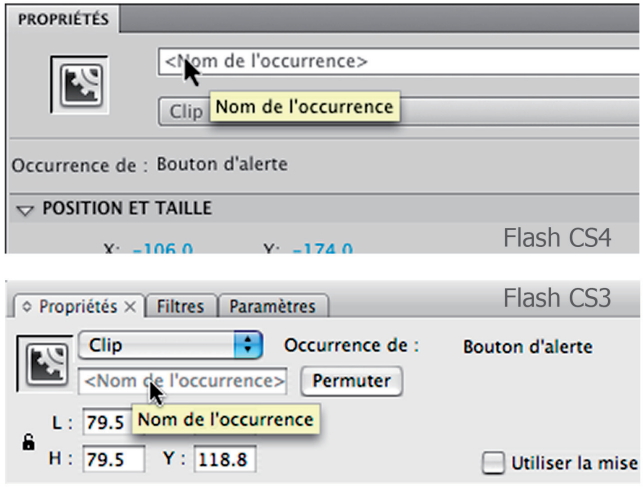

Figure 1-5
En fonction de la version de Flash sur laquelle vous travaillez, la palette Propriétés peut être différente. Celle de Flash CS4 (en haut) est allongée verticalement, alors que celle de Flash CS3 (en bas) est allongée horizontalement.

À noter que vous ne pouvez pas donner n'importe quel nom à une occurrence. Vous devez en effet respecter les règles suivantes.

- Le nom ne doit pas commencer par une majuscule (ex. : `curseur` est accepté, mais pas `Curseur`).
- Le nom ne doit pas contenir de caractères spéciaux (ex. : &, @, #, $, ı, etc.), d'espaces ou de caractères accentués (ex. : é, à, ô, ù, etc.).
- Le nom doit être le plus représentatif possible (évitez les acronymes, par exemple).

Noms, préfixes et suffixes

Comme nous venons de l'évoquer, vous ne devez surtout pas utiliser d'acronymes et d'abréviations. De ce fait, privilégiez des noms « un peu longs » comme : `boutonRetour-Sommaire`, `ecran-Accueil`, `zoneDiaporama` ou encore `boutonChargementImage`.

Vous rencontrerez parfois des préfixes ou suffixes dans les noms d'occurrences. Ainsi, les mentions `bt` ou `btn` peuvent définir en préfixe, ou suffixe, une occurrence dont la fonction sera d'être un bouton. Le suffixe `txt` indiquera, par exemple, qu'il s'agit d'une zone de texte. Certains suffixes tels que `_txt`, `_btn` ou `_mc` sont officiels et reconnus par Flash. Ils ont un effet très précis qui est d'afficher un menu contextuel déroulant pour vous aider à choisir une propriété ou une méthode. Ces deux derniers termes vous sembleront peut-être abstraits, c'est pourquoi nous ne détaillerons pas davantage notre explication à ce sujet.

Logique de construction d'une interface

Le titre de cette section est à nuancer car nous allons aborder dans les pages suivantes deux problématiques précises : la construction d'un écran (l'interface d'une image) et celle de tous les écrans d'une animation (l'interface de notre animation dans son ensemble). Ce titre peut donc se lire de deux manières différentes en fonction de la progression de la construction de votre animation.

Cette ambiguïté vient du fait que nous utilisons un scénario avec plusieurs images pour construire les différents écrans d'une même animation. En fonction du contenu de vos différentes images-clés, lors de la navigation au sein de votre animation, on peut avoir l'impression qu'on passe d'un écran à un autre ou, au contraire, que seules quelques zones de l'écran changent. Lorsqu'un développeur AS3 doit construire l'interface d'une animation, il travaille sur une seule image (clé). De ce fait, il ne rencontre pas les mêmes problèmes que ceux que nous allons aborder dans les pages qui suivent.

Comme nous l'avons évoqué dans les pages précédentes, il existe plusieurs approches pour construire l'interface d'une animation Flash. Cela va du simple glisser-déplacer d'un symbole (directement dans l'interface de Flash, de la bibliothèque vers

la scène) à une approche entièrement dynamique à l'aide d'une base de données MySQL ou XML et de scripts en AS3 faisant appel à des commandes de construction dynamique de l'interface. Afin de mieux appréhender ces différentes approches, voici une brève description de chacune d'entre elles.

Bibliothèque, scène et scénario

Si vous avez acheté ce livre, c'est que vous êtes au moins capable de placer sur la scène d'une animation Flash des symboles que vous avez préalablement construits (ces derniers ont été automatiquement ajoutés à la bibliothèque au fur et à mesure de leur création).

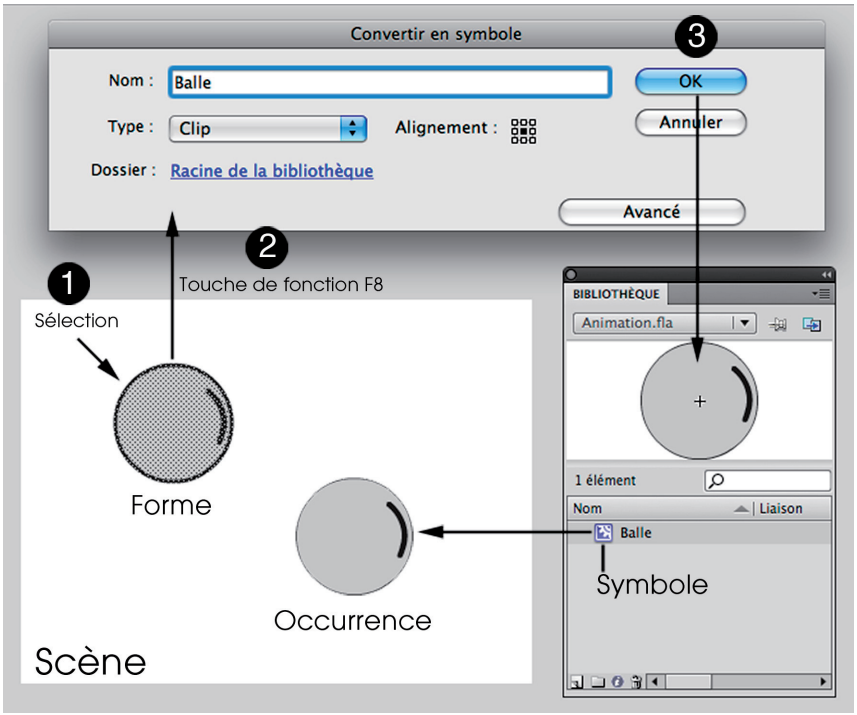

Figure 1-6
La création d'un symbole peut s'effectuer de trois façons différentes, celle-ci est la plus simple.

La technique du glisser-déplacer de symboles est la plus simple, mais elle reste très limitée dans ses possibilités de développement tant que vous n'utilisez qu'une seule image dans le scénario de votre animation. Rappelons que vous pouvez également

utiliser plusieurs images-clés, sur un ou plusieurs calques, solution la plus adaptée en général pour construire une animation interactive.

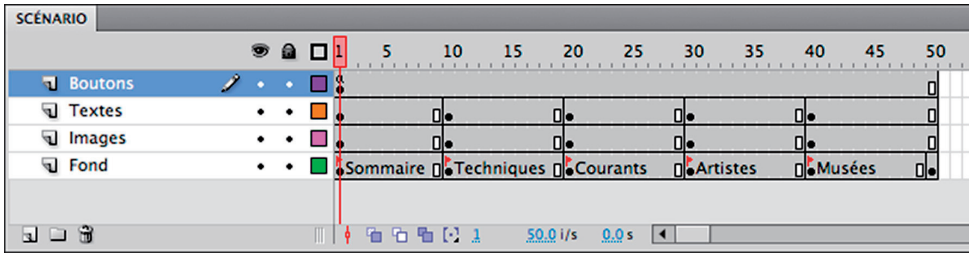

Figure 1-7
La construction d'une interface passe par l'utilisation de plusieurs calques, images et images-clés.

Dans l'exemple de la figure 1-7, nous avons une animation qui contient un écran d'accueil (appelé également « sommaire ») sur l'image 1, puis quatre écrans différents. Il s'agit d'une animation constituée de cinq écrans qui présentent un exposé sur la peinture.

Pourquoi ne pas utiliser uniquement cinq images adjacentes ?

Dans le cas où vous auriez besoin d'ajouter une image entre deux autres, il est préférable de disposer d'images inutilisées, c'est pourquoi l'espacement des images-clés dans le scénario de votre animation est un réflexe d'anticipation que vous devez toujours avoir.

Résumé

Cette méthode de construction d'interface est très simple à maîtriser, mais elle peut très vite devenir rébarbative, notamment dans le cas où plusieurs occurrences doivent être positionnées sur la scène.

L'avantage d'une telle construction est de pouvoir créer soi-même chaque écran de l'animation interactive. Très peu de lignes de code sont nécessaires pour ajouter de l'interactivité à votre animation (si vous souhaitez simplement déplacer la tête de lecture de votre animation). Reportez-vous au chapitre 3, « Naviguer au sein d'une animation », pour découvrir comment passer d'une image à une autre.

À noter qu'avec cette approche de construction d'une animation, l'ajout d'un *preloader* (appelé également « barre de préchargement ») est nécessaire pour gérer le préchargement de votre animation si vous souhaitez garder une certaine fluidité dans la lecture de votre animation.