ActionScript 3

Programmation séquentielle et orientée objet

3e édition

David Tardiveau

 ISBN : 978-2-212-13238-0

EYROLLES

Avant-propos

Pourquoi un tel ouvrage ?

> **ATTENTION : note aux néophytes en programmation**
>
> À la fin de la lecture de cet avant-propos, si les quelques pages qu'il contient vous semblent trop abstraites, lisez le chapitre 1 et passez directement au troisième.

Le manuel officiel de l'ActionScript 3

Si vous avez déjà ouvert la documentation officielle du langage ActionScript 3, vous aurez sûrement remarqué à quel point il est difficile d'en comprendre les explications qui, bien que très précises, demeurent abstraites pour les néophytes en programmation.

En tant qu'enseignant, j'aurais tendance à dire qu'à vouloir être trop précis, on perd souvent en clarté !

La société Adobe n'a malheureusement pas le choix car ce manuel dédié au langage ActionScript s'adresse à tout le monde, y compris les développeurs les plus expérimentés, et ces derniers ont également besoin d'informations qui leur parlent. Le guide de référence proposé par la société éditrice se doit donc d'être aussi exhaustif que possible.

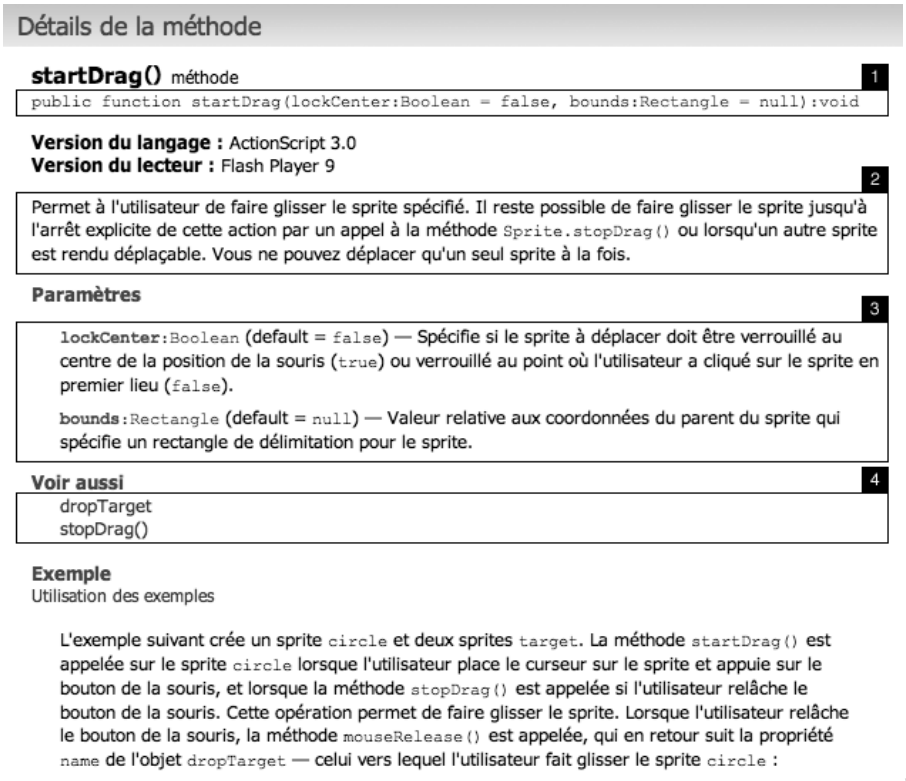
Mais ce manuel est-il si abstrait pour les débutants en programmation ? Prenons la définition de la méthode `startDrag()` de la figure A-1. Nous allons découvrir que, selon le niveau de chacun, on peut plus ou moins comprendre certains points. Analysons ensemble les quatre parties mises en avant au travers des cadres numérotés de 1 à 4.

Point numéro 1 : seuls les développeurs ayant déjà un bon niveau en programmation peuvent interpréter le sens de cette ligne d'instruction. Elle permet néanmoins de comprendre très rapidement la syntaxe de la méthode accompagnée des paramètres obligatoires et facultatifs.

Point numéro 2 : il s'agit certainement de l'explication la plus abordable par l'ensemble des développeurs, néophytes et confirmés, car elle définit le fonctionnement et/ou l'utilité du terme.

Point numéro 3 : cette partie de l'explication est primordiale car elle précise les paramètres à utiliser lorsque le terme est une méthode. Malheureusement, elle n'est compréhensible qu'à partir du moment où un développeur possède déjà les connaissances requises pour programmer.

Point numéro 4 : par analogie ou par opposition, il est parfois intéressant de comprendre des termes ayant une relation avec celui que vous cherchez à maîtriser initialement.



Détails de la méthode

startDrag() méthode 1

```
public function startDrag(lockCenter:Boolean = false, bounds:Rectangle = null):void
```

Version du langage : ActionScript 3.0
Version du lecteur : Flash Player 9

2

Permet à l'utilisateur de faire glisser le sprite spécifié. Il reste possible de faire glisser le sprite jusqu'à l'arrêt explicite de cette action par un appel à la méthode `Sprite.stopDrag()` ou lorsqu'un autre sprite est rendu déplaçable. Vous ne pouvez déplacer qu'un seul sprite à la fois.

Paramètres 3

`lockCenter:Boolean` (default = `false`) — Spécifie si le sprite à déplacer doit être verrouillé au centre de la position de la souris (`true`) ou verrouillé au point où l'utilisateur a cliqué sur le sprite en premier lieu (`false`).

`bounds:Rectangle` (default = `null`) — Valeur relative aux coordonnées du parent du sprite qui spécifie un rectangle de délimitation pour le sprite.

Voir aussi 4

dropTarget
stopDrag()

Exemple
Utilisation des exemples

L'exemple suivant crée un sprite `circle` et deux sprites `target`. La méthode `startDrag()` est appelée sur le sprite `circle` lorsque l'utilisateur place le curseur sur le sprite et appuie sur le bouton de la souris, et lorsque la méthode `stopDrag()` est appelée si l'utilisateur relâche le bouton de la souris. Cette opération permet de faire glisser le sprite. Lorsque l'utilisateur relâche le bouton de la souris, la méthode `mouseRelease()` est appelée, qui en retour suit la propriété `name` de l'objet `dropTarget` — celui vers lequel l'utilisateur fait glisser le sprite `circle` :

Figure 0-1

Les explications de l'aide officielle de l'ActionScript 3 ne sont pas toujours compréhensibles pour tout public.

```
import flash.display.Sprite;
import flash.events.MouseEvent;

var circle:Sprite = new Sprite();
circle.graphics.beginFill(0xFFCC00);
circle.graphics.drawCircle(0, 0, 40);

var target1:Sprite = new Sprite();
target1.graphics.beginFill(0xCCFF00);
```

```
target1.graphics.drawRect(0, 0, 100, 100);
target1.name = "target1";

var target2:Sprite = new Sprite();
target2.graphics.beginFill(0xCCFF00);
target2.graphics.drawRect(0, 200, 100, 100);
target2.name = "target2";

addChild(target1);
addChild(target2);
addChild(circle);

circle.addEventListener(MouseEvent.MOUSE_DOWN, mouseDown)

function mouseDown(event:MouseEvent):void {
  circle.startDrag();
}
circle.addEventListener(MouseEvent.MOUSE_UP, mouseReleased);

function mouseReleased(event:MouseEvent):void {
  circle.stopDrag();
  trace(circle.dropTarget.name);
}
```

Comme vous pouvez le constater, si quelqu'un vous dit : pour réaliser un puzzle, tu dois utiliser la méthode `startDrag()`, vous pourrez difficilement vous en sortir avec les explications précédentes, surtout si vous n'avez jamais programmé.

La présentation de ce script dans un contexte précis le rend, paradoxalement, encore moins évident à comprendre. L'utilisateur du manuel doit d'abord analyser le script avant de découvrir les parties indispensables pour réaliser un glisser-déposer.

Rappelons que l'aide officielle du langage vous sera un jour très utile car il s'agit d'un manuel extrêmement bien conçu, mais vous devez d'abord comprendre les concepts de la programmation et maîtriser ses mécanismes. Cette maîtrise passera certainement par une longue phase de pratique du langage. En attendant, si ce référentiel ne peut vous aider, il vous reste Internet et les livres pour apprendre l'AS3 (ActionScript 3).

Sur Internet, les forums constitueront pour vous une aide précieuse lorsque vous rencontrerez des problèmes. Le plus réputé dans la communauté francophone de Flash et de l'ActionScript est celui de Mediabox : `http://flash.mediabox.fr/index.php`.

Concernant les publications papier, si vous avez acheté un livre pour apprendre l'AS3, c'est que vous savez déjà quels sont les avantages et les inconvénients d'une telle méthode d'acquisition. Allons tout de même plus loin et présentons l'approche de cet ouvrage.

Différences entre l'AS1, l'AS2 et l'AS3

Remarque

La première édition de ce livre est sortie en mars 2008. À l'époque, l'ActionScript 3 était un nouveau langage et l'AS2 était encore majoritaire dans les développements Flash. Les propos et interrogations soulevées dans les paragraphes ci-dessous avaient du sens. Ce n'est plus le cas en 2011, mais nous avons souhaité conserver ces quelques lignes qui permettent de comprendre l'évolution de ce langage qu'est l'AS3.

L'ActionScript 3 constitue un tournant majeur dans l'histoire de Flash et plus particulièrement du langage AS. Sans entrer dans les détails, nous pouvons tout de même dire que cette version creuse une fois encore l'écart entre les développeurs confirmés et les néophytes en matière de programmation. Pour commencer, l'un des objectifs de ce livre est de réduire cet écart.

L'approche de la programmation étant différente entre l'AS1, l'AS2, et l'AS3, nous nous retrouvons aujourd'hui avec trois populations de développeurs qui ne gèrent pas le code de la même façon pour un même langage. Si nous étions réellement objectifs, nous pourrions tout de même affirmer que l'utilisation de l'ActionScript ne fait pas de la personne qui l'utilise un développeur. Par ailleurs, il n'existe plus beaucoup d'utilisateurs qui « codent » en AS1 ; en revanche, il reste encore trop de développeurs qui ne sont pas passés à l'AS3, et ce pour deux raisons.

Pour certains développeurs, l'AS3 fait peur, car il a la réputation d'un langage complètement orienté objet ! Même s'il est toujours préférable de développer en POO pour les gros projets, il est tout à fait possible de développer dans l'interface de Flash à travers la fenêtre Actions dans le cas de développements moins ambitieux. Mais le manque de temps pour apprendre l'AS3 n'est pas l'unique raison de cette migration ratée : il est aussi difficile d'admettre que pendant un petit moment, on ne sera pas capable d'assurer le même rythme de production que celui correspondant à un développement en AS2. Les sociétés pour lesquelles travaillent les développeurs AS cherchant également à faire du profit en réduisant les délais de production, ne permettent pas, de ce fait, de laisser du temps pour apprendre l'AS3.

Avant d'expliciter davantage l'approche pédagogique de ce livre face au besoin d'apprentissage de l'AS3, revenons sur les différences qui séparent les trois versions.

La vraie différence entre l'AS1 et l'AS2

Pour certains anciens utilisateurs de Flash, la différence entre l'AS1 et l'AS2 se caractérise, à tort, par le fait de placer les scripts d'une animation sur les occurrences (AS1) ou sur l'image-clé (AS2). Les explications qui suivent vont vous démontrer le contraire.

Ce qui caractérise l'ActionScript 1, c'est le fait de placer les scripts d'une animation sur l'image-clé d'un scénario (le scénario général ou celui d'un clip) ou sur une occurrence.

La création de l'AS1 n'est pas vraiment datée, mais nous pouvons dire qu'avant 2000 (la sortie de Flash 5), le système de gestion des scripts ne représentait pas vraiment un langage à part entière. De 2000 à 2003, l'AS1 aura connu deux syntaxes pour gérer ses événements : les scripts pouvaient être placés sur les occurrences ou sur l'image-clé pour effectuer une même action.

Les utilisateurs de Flash peu expérimentés en programmation préféraient placer les scripts sur les occurrences (de bouton et de clips), cette logique étant plus compréhensible pour eux : on place le script sur le bouton sur lequel l'utilisateur cliquera pour déclencher une action. Par ailleurs, la syntaxe était plus simple. Les utilisateurs plus expérimentés en programmation préféraient placer tout le code d'une animation au même endroit, sur une image-clé du scénario principal, car le débogage est plus simple puisque le code est centralisé. Au sein de la communauté des utilisateurs de Flash, puis ensuite des développeurs Flash, cette double approche du langage a constitué deux groupes d'utilisateurs.

Utilisateurs de Flash et développeurs Flash

C'est avec le temps qu'est née la différence entre les utilisateurs de Flash et les développeurs Flash. De 2000 à 2005, un utilisateur de Flash était qualifié de flasheur, même s'il utilisait considérablement l'ActionScript. Progressivement, à partir de 2005, le marché de l'emploi a fait la différence entre les flasheurs qui sont principalement représentés par les utilisateurs de Flash au travers de son interface (utilisation importante du scénario) et les développeurs Flash qui font appel principalement au code pour contrôler une animation. On parle même aujourd'hui de développeur AS ou AS3.

À la sortie de l'AS2 en 2003 avec l'arrivée de Flash MX 2004, les développeurs qui avaient l'habitude de placer tout leur code sur une image-clé ont tout naturellement opté pour le nouveau mode de programmation : l'utilisation et l'appel d'un fichier externe pour rédiger les scripts d'une animation.

En résumé, l'AS1 se caractérise par le fait de placer ses scripts dans l'animation alors que l'AS2 fait appel à des fichiers externes. En programmation, il existe deux approches : la programmation orientée objet et la programmation séquentielle appelée aussi programmation structurée. La première approche est donc tout naturellement associée à l'AS2 et la deuxième à l'AS1.

Création d'un nouveau document dans Flash CS5

Aujourd'hui, lorsque vous demandez la création d'un nouveau document dans Flash CS5, vous pouvez opter pour un fichier Flash AS2 ou un fichier Flash AS3. Il est important de comprendre qu'Adobe a décidé de regrouper les versions AS1 et AS2 sous une même étiquette car elles utilisent des syntaxes différentes mais un même langage. Les termes sont identiques, seules quelques spécificités les différencient. L'ActionScript 3 n'utilise plus du tout la même syntaxe, mais surtout, le langage est complètement différent (par exemple, la propriété `._x` en AS2 s'écrit `.x` en AS3). Lorsque la société Adobe fait référence à l'AS2, elle veut marquer la différence entre les deux registres de vocabulaires de l'AS1/AS2 et de l'AS3 (les approches de programmation orientée objet et séquentielle étant sous-entendues pour l'AS2).

Mais pourquoi tous les utilisateurs ne sont-ils pas passés à l'AS2 ? C'est effectivement la question que nous devons nous poser, car la réponse que nous allons formuler d'ici quelques lignes va, par la même occasion, nous faire comprendre la différence qui existe entre l'AS3 et l'AS2 !

Il est important de comprendre que la programmation orientée objet, qui est donc associée à l'AS2 et l'AS3, nécessite un bon niveau en programmation. Parallèlement, Flash est un logiciel qui connaît depuis cinq à six ans un énorme succès auprès de publics très divers, y compris des utilisateurs novices en programmation et n'ayant pas un grand besoin en termes de développement informatique. Pourquoi un imprimeur aurait besoin d'apprendre un langage complexe pour réaliser une animation avec une légère interactivité ? Bien évidemment, nous avons pris ce corps de métier pour notre exemple, mais il en existe bien d'autres. Par ailleurs, pourquoi des personnes talentueuses dans leur domaine devraient-elles apprendre un langage de programmation pour réaliser une animation interactive ?

Flash, un logiciel à tout faire

D'une façon générale, tous les métiers de la communication ont trouvé dans ce logiciel un moyen facile et rapide de diffuser une information sur Internet. Flash proposant aussi de publier des animations pour les supports off-line, un marché supplémentaire s'est même créé, remplaçant par la même occasion la plupart des productions réalisées par le logiciel Director.

Répondons simplement à ces deux questions en affirmant que l'AS2, avec une approche de programmation séquentielle (ou structurée), va malheureusement perdurer encore quelques années, car les besoins en programmation ne sont pas les mêmes pour tout le monde. Pourquoi utiliser le terme malheureusement dans la phrase précédente ? Est-ce un jugement subjectif ? Non, cela signifie simplement que la différence que nous avions entre les deux communautés d'utilisateurs de l'ActionScript va non seulement perdurer mais aussi se renforcer entre l'AS2 (à l'approche de programmation séquentielle) et l'AS3.

L'ActionScript 3 est un langage complètement orienté objet, car la rédaction des lignes d'instructions qui composent un script ou un programme est très précise. La structure d'un document a complètement été repensée ; le développeur manipule à présent des objets imbriqués et non plus des occurrences de type Clip ou Bouton, comme c'était le cas dans les versions antérieures à Flash CS3. Cette version du langage n'est-elle donc pas plus accessible aux développeurs novices ? Pour ces derniers, la réponse est simple : vous pouvez développer en AS3 avec une approche séquentielle !

L'un des objectifs de ce livre est non seulement de présenter en parallèle les deux syntaxes (programmation orientée objet et programmation séquentielle), mais aussi de démontrer qu'il est possible d'apprendre l'AS3 si on débute en programmation.

Pourquoi proposer à des développeurs de programmer en AS3 avec une syntaxe séquentielle ? Ces dernières années, l'enseignant que je suis a été quelque peu excédé face aux comportements de certains collègues ou développeurs confirmés. Non, la programmation orientée objet n'est pas accessible à tout le monde et ce n'est pas non plus une question de pédagogie. Il a été assez agaçant de constater, avec la montée en puissance de l'AS3,

à quel point certains individus peuvent être si obtus et penser que tout le monde possède les mêmes facilités. J'aurais envie de dire à ces gens-là : « Comment, vous ne savez pas préparer une charlotte au chocolat ? Il suffit pourtant de mélanger des ingrédients en suivant une recette… » Ah bon, chef cuisinier c'est un métier ?

La mort de Flash et de l'AS3 ?

Depuis que l'industrie existe, des techniques naissent pour disparaître quelques années ou décennies plus tard. L'informatique n'échappe pas à cette règle : elle a connu et connaît le même cycle. La mort d'une technique ou d'une technologie est liée dans la plupart des cas à l'arrivée d'une nouvelle, qui détrône l'existante. Aujourd'hui, au premier trimestre 2011, il est vrai que les développements Flash sont moins nombreux qu'en 2009-2010 pour deux raisons, mais va-t-on assister pour autant à la disparition du couple Flash/ActionScript ? Le fait que l'iOS (l'OS de l'iPad, de l'iPhone et de l'iPod Touch) ne supporte pas le plug-in Flash et que l'arrivée du HTML 5 change les habitudes de certains développeurs suffira-t-il à rendre le format SWF obsolète ? Par ailleurs, utilise-t-on Flash uniquement sur Internet ?

Concernant l'iOS, il semble improbable qu'Apple ouvre soudainement son OS au plug-in Flash, mais l'histoire de l'informatique démontre depuis des années que les ennemis d'hier peuvent aujourd'hui collaborer sur des projets communs. Apple n'a pas voulu supporter le plug-in signé Adobe pour deux raisons ; revenons sur celles-ci.

À la sortie de l'iPhone, en juin 2007, on note que l'OS du smartphone ne supporte pas le plug-in Flash, mais il n'y a pas encore de polémique, on déplore simplement cette situation. Deux ans plus tard, Adobe et Apple entrent en conflit à ce sujet, ce dernier assurant que la technologie Flash ralentit l'iPhone. Même si cela n'est plus tout à fait vrai aujourd'hui, il est indéniable que si Apple avait accepté un tel support, l'AppStore n'aurait pas connu un succès aussi rapide ! Les utilisateurs d'iPhone dépenseraient-ils autant pour des applications, depuis 2007, s'ils trouvaient sur le Web celles qu'ils achètent sur l'AppStore ?

En 2010, Apple à sorti l'iPad (qui tourne sous l'iOS). Les consommateurs se sont tournés massivement vers ce nouvel appareil et l'absence du support de Flash explique en partie la diminution des développements réalisés traditionnellement en Flash.

Remarque

À l'heure où j'écris ces lignes, l'iPad 2 vient de sortir. Il ne se différencie pas tellement de celui de la première génération. Une chose est sûre, l'iPad est idéal pour les loisirs mais guère utilisable pour le travail. À titre personnel, je l'utilise depuis le jour de sa sortie et la saisie d'un texte dans un e-mail est toujours fastidieuse dès lors qu'il est nécessaire de faire des corrections (absence de flèches de déplacement au clavier). Tant que le système de pointage sur une ligne de texte sera celui que nous connaissons aujourd'hui, par une pression du doigt sur une ligne, il sera impossible d'utiliser cet outil pour le travail. À ce jour, des tablettes sortent chez tous les constructeurs (certaines supportent le plug-in Flash), mais attendons encore un à deux ans avant de déclarer que de tels appareils vont changer les habitudes. Dans certains corps de métiers, ce type d'appareil est bien adapté, mais l'ordinateur portable tel que nous le connaissons aujourd'hui a encore de beaux jours devant lui. Je crois davantage en un appareil de la taille d'un SmartPhone qu'on brancherait sur une station qui transformerait un moniteur et un clavier en un vrai poste de travail.

Revenons au HTML 5. Il est indéniable que les capacités du trio HTML 5/CSS/JavaScript vont jouer un rôle important dans la baisse des productions réalisées en Flash. Il suffit de parcourir le Web pour découvrir qu'il existe déjà de très belles réalisations interactives. Dans ce cas, pourquoi n'en rencontre-t-on pas encore sur les plus grands sites de la Toile ? À ce jour, le problème se pose encore en terme de navigateur, car ils ne supportent pas tous le HTML 5 de la même façon et les mêmes formats de vidéo. Espérons simplement que ces derniers propos ne se vérifient plus dans les mois à venir.

Flash et l'ActionScript 3, sont-ils dans ce cas bientôt obsolètes ? Je ne pense pas et ma réponse ne se base pas uniquement sur un attachement historique au logiciel ou une quelconque nostalgie. Pour commencer, il va falloir que de nombreux développeurs se forment au HTML 5 et au JavaScript. Tout ne va pas être réalisable en HTML 5/CSS/JavaScript aussi rapidement que cela peut l'être en AS. Les environnements de développement ne sont pas les mêmes, la portabilité des réalisations Flash est bien plus grande et indépendante des navigateurs. Ce sont autant d'éléments qui permettent de dire que si Flash devait massivement disparaître du Web, cela ne pourra pas se faire avant 2 ou 3 ans.

Approche pédagogique de ce livre

La plupart des ouvrages dédiés à l'apprentissage d'un langage informatique proposent une table des matières basée sur les notions élémentaires pour maîtriser l'algorithme. C'est une excellente logique : il est ainsi plus facile de passer d'un langage à un autre. Une telle approche pédagogique permet également une montée en puissance de la difficulté en phase d'apprentissage. J'ai pu constater au long de ma carrière qu'il existe de nombreuses approches pédagogiques pour transmettre un savoir. Elles s'adressent toutes à des publics aux profils différents.

Un étudiant qui se forme aux métiers du développement aura besoin de connaître les bases de la programmation. Il devra savoir construire un algorithme, mais aussi comprendre et maîtriser les concepts de la programmation orientée objet. Lorsque tout cela est acquis, il peut alors passer d'un langage à un autre assez facilement : il doit simplement se familiariser avec les spécificités de chacun.

Si vous connaissez déjà les bases de la programmation, ce livre va vous sembler évident et simple. En revanche, si vos connaissances en programmation sont réduites, je tiens à vous rassurer : la progression du livre s'appuie sur les besoins rencontrés en production. Il sera parfois nécessaire de faire quelques allers-retours d'un chapitre à un autre mais, dans l'ensemble, l'ordre des chapitres a été choisi de sorte que vous puissiez programmer une animation Flash, même si vous ne maîtrisez pas le contenu du chapitre 2 de ce livre.

Depuis plus de dix-huit ans, mon métier est de transmettre mes connaissances, initialement dans le domaine de la PAO, puis dans les métiers du multimédia depuis 1996. En plus de dix ans, quelque 1 200 à 1 500 personnes sont passées dans les différentes formations Flash que j'ai assurées pour le compte de divers établissements privés et publics. Ces apprenants ont tous buté sur les mêmes difficultés et sont généralement tombés dans les mêmes pièges à chaque version du langage !

Aujourd'hui, pour apprendre un langage, il est très simple d'accéder à une formation, mais cela reste coûteux. De nombreuses publications papier restent alors abordables : le prix d'un livre est compris entre 15 et 60 euros. Pour celles et ceux qui veulent investir du temps, mais pas d'argent, il reste le Web qui regorge de sites, mais il faut passer un certain temps à dénicher les bonnes adresses. En 1990, lorsque j'ai dû apprendre par moi-même à utiliser mes premiers logiciels, puis mes premiers langages de programmation, j'ai rencontré les mêmes difficultés d'apprentissage que mes étudiants aujourd'hui (d'autant que le Web n'existait pas encore, il n'y avait pas de forums et, en dehors de la bibliothèque de la Cité des Sciences à Paris, les bibliothèques municipales proposaient très peu de publications dans ces domaines). Je me souviens encore de ces soirées et nuits passées à essayer de comprendre certaines notions élémentaires... Cela me permet donc maintenant de prévoir des approches pédagogiques différentes selon les techniques à transmettre.

Chaque nouvelle explication présentée dans ce livre s'accompagnera d'un exemple simple. Seulement une ou deux notions nouvelles seront utilisées à la fois afin que vous compreniez bien. Notre approche ne sera pas systématiquement exhaustive : trop d'informations ou des informations trop précises empêchent bien souvent la compréhension d'une nouvelle notion.

Un livre dédié à la programmation orientée objet ou séquentielle ?

Qui peut le plus peut le moins ! Cela résume l'approche globale de ce livre…

À la lecture de la table des matières de cet ouvrage, vous pouvez découvrir qu'une part importante est consacrée à l'apprentissage de la programmation orientée objet, non seulement au travers de chapitres dédiés à ce mode de programmation, mais également par le biais de nombreux exemples.

Lorsque vous apprendrez une nouvelle notion ou technique, nous vous expliquerons le mécanisme général, puis s'en suivra généralement un exemple de script à placer sur une image-clé. Lorsque cela s'avèrera nécessaire, nous vous présenterons également le script d'un fichier .as.

Tous les exemples de ce livre sont téléchargeables à l'adresse suivante :

```
http://www.yazo.net/eyrolles
```

Nous ne privilégions aucun mode de programmation car nous utilisons les deux dans la plupart des cas, mais nous mettons simplement en avant le script, qui est plus facile à appréhender. Pour les plus expérimentés d'entre vous, il sera très simple de copier-coller les lignes d'instructions dans un fichier externe lorsqu'un script en POO ne sera pas présenté. Rassurez-vous, pour les scripts les plus complexes de ce livre, nous vous présenterons un exemple dans chacun des deux modes de programmation.

Quelques termes à connaître

Si vous ne connaissez pas les termes employés en programmation, il est alors indispensable que vous lisiez les définitions ci-après. Attention, elles ne sont volontairement pas conformes à des définitions officielles que vous auriez pu trouver dans des livres spécialisés sur la programmation ou des sites dédiés, dans la mesure où la volonté d'être compris de tous constitue notre principal objectif. Des sites comme `http://www.commentcamarche.net` vous donneront sûrement les informations que vous pourrez rechercher.

Lorsque nous allons expliquer toutes les notions et techniques abordées dans ce livre, nous emploierons certains termes dont voici le sens.

Instance : il existe deux types d'instances qui résultent de deux opérations différentes. Si vous placez un symbole sur la scène, vous obtenez une occurrence, mais nous pourrions également dire qu'il s'agit d'une instance du symbole. En ActionScript, vous utiliserez parfois le mot `new` qui permet d'obtenir une instance de classe. Une instance est donc représentée par un mot (et/ou une occurrence sur la scène). Lorsque vous aurez besoin d'écouter un son dans une animation, vous créerez une instance de la classe `Sound()` que vous manipulerez (charger un son dans cette instance, jouer le son de cette instance, régler le volume de cette instance).

Propriété : imaginez une instance (ou occurrence) sur la scène. Si vous souhaitez contrôler sa position, sa transparence ou sa rotation, il faudra alors écrire une ligne d'instruction qui fera référence au nom de l'instance et à la propriété à modifier. La transparence d'une occurrence constitue l'une des propriétés d'une instance, au même titre que son échelle horizontale, sa rotation, etc. En programmation orientée objet, ce terme s'étend à d'autres termes d'un script que nous n'évoquerons pas ici pour l'instant.

Méthode : dans un script ou un programme, vous écrirez certains termes avec des parenthèses à la fin. Ces derniers sont qualifiés de fonctions ou méthodes et nous pourrions les comparer aux verbes d'une phrase. Leur rôle est d'agir dans un programme ; par exemple : créer un rectangle sur la scène, changer la casse d'un texte, régler le volume d'un son, charger une image sur la scène, etc. Toutes les méthodes sont des fonctions, mais toutes les fonctions ne sont pas des méthodes. Dès que vous écrivez…

```
function marcher() {
  //ligne d'instruction à exécuter
}
```

il s'agit d'une fonction. Mais, dans certains cas, si cette fonction est précisément écrite à l'intérieur d'une classe, on la qualifie alors de méthode. Dans l'exemple précédent, nous venons de créer une méthode : autrement dit, vous expliquez au programme ce qu'il devra faire lorsque le mot `marcher()` sera écrit seul dans une ligne d'instruction.

Événement : pour temporiser un script, vous devez utiliser un événement. Le fait de survoler ou de cliquer sur une occurrence constitue un événement. De la même façon, si vous voulez exécuter une action lorsque la lecture d'un son est terminée, vous ferez appel à l'événement `soundComplete`. Un événement est un mot qui sera placé à un endroit fixé dans une ligne d'instruction. Pour être plus précis, cette ligne d'instruction est générale-

ment placée dans un gestionnaire d'événement. Nous n'expliquerons pas ce terme car le chapitre 3 de ce livre est consacré intégralement à cette notion.

Ligne d'instruction : elle est comparable à la phrase d'un paragraphe dans un texte écrit en français. Au même titre qu'une phrase contient un sujet, un verbe, un complément…, une ligne d'instruction peut contenir un nom d'occurrence, une propriété, une méthode, des mots-clés, etc. Une ligne d'instruction se termine toujours par un point-virgule (qui est très souvent omis par les développeurs Flash néophytes).

Classe : c'est un regroupement de lignes d'instructions saisies avec une syntaxe précise pour structurer une partie d'un programme. Cette notion est pour l'instant trop abstraite pour lui donner davantage de sens.

Package : c'est une structure du code qui englobe une classe. Cette notion est également trop abstraite pour l'instant pour lui donner davantage de sens.

L'affichage de résultats sur la scène, dans les exemples de ce livre

Pour afficher un texte dans une animation Flash, il existe deux solutions, mais seule la première des deux techniques que nous allons aborder ci-après permet de visualiser un contenu textuel sur la scène, dans une animation au format SWF. La deuxième technique n'est valable qu'à partir du moment où vous travaillez dans l'IDE de Flash.

Un texte dynamique sur la scène

À de nombreuses reprises, nous aurons besoin d'afficher des textes sur la scène, nous utiliserons donc la méthode suivante :

1. Créer un texte sur la scène avec l'outil Texte (maintenez un clic sur la scène, faites glisser votre souris pour définir une largeur de texte, relâchez le bouton de votre souris).
2. Dans la partie supérieure de la palette Propriétés, si cela n'est pas déjà le cas, sélectionnez le type Texte dynamique (ou Texte de saisie).
3. Nommez l'occurrence obtenue (exemple : `affichageResultat`).
4. Cliquez sur la scène puis dans la fenêtre Actions afin de saisir la ligne d'instruction ci-après :

```
affichageResultat.text = 3+45;
```

Comme vous pouvez le remarquer, nous utilisons la chaîne de caractères `.text` derrière le nom de l'occurrence, car nous faisons référence à la propriété `text` de l'instance pour stocker l'information (le résultat du calcul). Celles et ceux qui avaient l'habitude d'utiliser un nom de variable en AS1/AS2 noteront que cette technique n'est plus valable.

La fonction trace()

Il existe aussi une deuxième solution qui consiste à afficher une information dans la fenêtre Sortie de Flash.

Placez la ligne d'instruction ci-après pour tester cette fonctionnalité :

```
trace("Bonjour");
```

ou encore :

```
trace ("Bonjour, nous sommes le "+new Date().getDate());
```

Vous aurez peut-être remarqué que nous avons utilisé l'opérateur + dans les parenthèses de la fonction `trace()` pour pouvoir effectuer une concaténation, c'est-à-dire un regroupement de plusieurs informations.

Table des matières

2

La gestion des occurrences sur la scène

Afin d'optimiser votre code pour gérer les occurrences affichées sur la scène, il est important de comprendre le rôle et le fonctionnement de la liste d'affichage (displayList). Nous consacrons donc un chapitre à ce système de gestion. Nous présenterons ensuite la méthode `addChild()`, dont le but est de placer une instance de type objet d'affichage ou conteneur d'objets d'affichage sur la scène.

La liste d'affichage (displayList) d'une animation

Le fait de référencer tous les objets d'affichage et les conteneurs d'objets d'affichage présents dans une animation est la notion la plus élémentaire de l'ActionScript 3.

Dans les versions précédentes, nous devions indiquer le chemin d'une occurrence pour pouvoir l'utiliser. La hiérarchie qui existait entre les occurrences d'une animation découlait de l'emploi d'une succession d'imbrications d'instances de type MovieClip.

Une arborescence était tout de même constituée virtuellement dans la mémoire de l'ordinateur pour déterminer les parents et les enfants éventuels d'un objet d'affichage ou d'un conteneur d'objets d'affichage.

Avec l'arrivée de l'AS3, la gestion des occurrences affichées sur la scène est donc différente, plus souple et plus logique. Ne plus devoir cibler une instance, en précisant son chemin avec une syntaxe pointée, peut éventuellement troubler les anciens développeurs Flash. Cela dit, le développement consacré à la méthode `addChild()` à la fin de ce chapitre vous démontre la simplicité d'accès à une occurrence, même si elle se trouve imbriquée dans un ou plusieurs conteneurs d'objets d'affichage.

Par ailleurs, nous ne sommes plus limités à des clips et des boutons : il existe à présent plusieurs types d'objets regroupés en classes.

Remarque

Si ce dernier point vous paraît trop abstrait, ajoutons que chaque type d'objet possède son propre vocabulaire, c'est-à-dire un ensemble de mots réservés associé aux instances qu'il générera. Pour celles et ceux qui travaillaient avec Flash 8 et les versions antérieures, les symboles de type Bouton et ceux de type Clip constituent un bon exemple : ce sont deux objets de types différents.

Chaque fois que vous placez une occurrence sur la scène, manuellement (par glisser-déposer) ou dynamiquement (en AS, avec la méthode `addChild()` ou une fonction équivalente), vous alimentez une liste d'affichage, appelée également displayList. Celle-ci permet une gestion globale des occurrences de la scène. Il est ainsi possible de déterminer, par des instructions en ActionScript, le nombre d'imbrications d'occurrences (occurrences qui possèdent elles-mêmes d'autres occurrences), ce qui n'était pas le cas dans les anciennes versions de Flash.

Il est maintenant important de comprendre que vous pouvez disposer de deux types d'instances sur la scène : les occurrences qui peuvent en contenir d'autres (représentées par des rectangles gris sur la figure 2-1) et les occurrences qui ne peuvent contenir rien d'autre que la représentation graphique qui les symbolise (les rectangles blancs de la même figure). En d'autres termes, on distingue les occurrences qui peuvent posséder des enfants de celles qui ne le peuvent pas. La liste d'affichage des instances contenues sur la scène de Flash constitue donc une hiérarchie qui peut être gérée sous la forme de nœuds XML avec une racine (la scène d'une animation) et des nœuds enfants (les objets d'affichage ou les conteneurs d'objets d'affichage). Cela s'avère très pratique pour cibler une occurrence précise de la scène. La scène de Flash est alors le premier conteneur de la liste d'affichage.

Quelle différence doit-on faire entre les objets d'affichage et les conteneurs d'objets d'affichage ?

Celles et ceux qui ont utilisé les anciennes versions de Flash, se souviennent que les occurrences de type clip pouvaient en contenir d'autres (de même type ou de type différent). De ce fait, en ActionScript, il était possible de faire référence à une occurrence imbriquée dans une autre. En revanche, une occurrence de type graphique ne pouvait pas être contrôlée par ce biais ; il ne servait donc à rien d'y placer des occurrences de clips nommées.

Nous pourrions ainsi comparer les conteneurs d'objets d'affichage à des occurrences de type MovieClip et les objets d'affichage à des occurrences de symbole de type Graphique, à une différence près : un objet d'affichage ne peut pas contenir d'autres instances (quel que soit son type).

Plus généralement, un conteneur d'objets d'affichage est une occurrence (ou une instance) qui peut en contenir d'autres, alors qu'un objet d'affichage ne le peut pas. Dès que vous placez un symbole sur la scène ou créez une instance en ActionScript et l'ajoutez avec la méthode `addChild()` (ou un moyen équivalent), vous créez un objet d'affichage ou un conteneur d'objets d'affichage. Vous augmentez par la même occasion le nombre d'objets contenus dans la liste d'affichage.

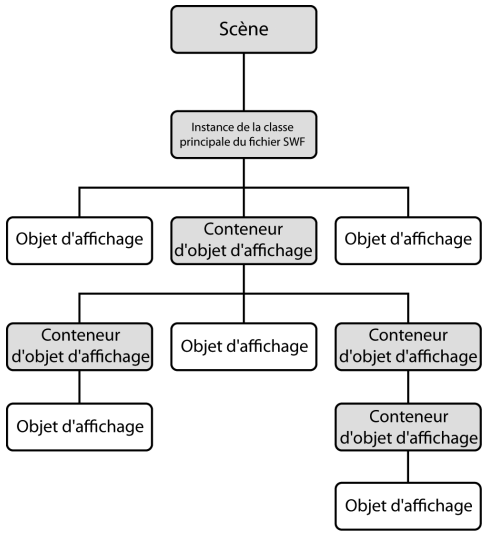


Figure 2-1

Sur la scène de Flash, vous devez distinguer deux types d'occurrences : celles qui ne contiennent pas d'autres occurrences et celles qui en contiennent.

Un objet d'affichage est toujours typé (au même titre qu'une occurrence de symbole est de type MovieClip, Bouton ou Graphique). Le schéma de la figure 2-2 présente les différents types d'objets (ou classes) disponibles dans une animation Flash. Précisons ici la notion d'héritage : un objet hérite des caractéristiques (propriétés, méthodes et événements) de l'objet qui se trouve immédiatement au-dessus de lui dans la hiérarchie.

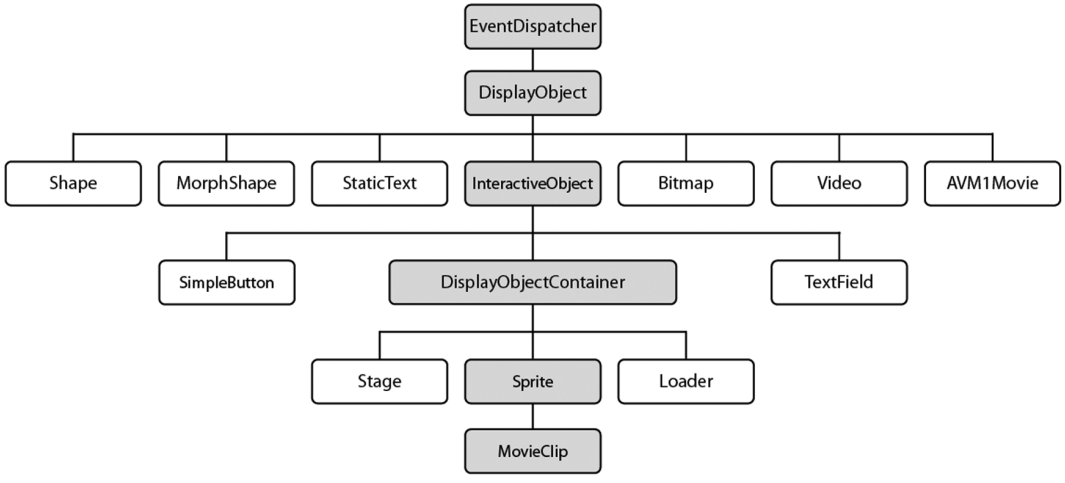


Figure 2-2

Cette arborescence vous présente l'ensemble des classes disponibles en AS3 et leurs héritages respectifs.

Les conteneurs d'objets d'affichage

Voici les différents types de conteneurs d'objets d'affichage.

- Il est conseillé d'utiliser les instances de la classe `Loader()` pour charger des images sur la scène (ou également des SWF).
- Les instances de la classe `Sprite()` servent à définir les zones principales de l'interface de votre animation. D'une façon plus générale, la fonction d'une instance de `Sprite()` est de contenir d'autres instances.
- Les instances de la classe `MovieClip()` sont comparables à celles de la classe `Sprite()` à un détail près : elles possèdent un scénario (timeline), ce qui signifie qu'elles peuvent contenir des images-clés.

> **Rappel**
>
> Avant la création de la classe `Sprite()` (apparue dans Flash CS3), le MovieClip servait de conteneur d'affichage sans avoir pour autant les mêmes avantages que les instances de la classe `Sprite()`.

La section « Structurer une mise en page avec les classes `Sprite()` ou `MovieClip()` » permet de mieux comprendre comment organiser la construction d'une image.

Les objets d'affichage

Commençons par préciser que les objets d'affichage énumérés ci-après peuvent être obtenus par l'exécution d'instructions en ActionScript ou en utilisant les outils et les commandes proposés dans l'interface de Flash.

Lorsqu'un objet d'affichage est placé sur la scène, il ne peut pas servir de conteneur d'affichage. Prenons l'exemple d'un rectangle tracé sur la scène avec un outil de dessin. Vous pouvez définir ses propriétés (couleurs de fond et de contour, épaisseur du trait, etc.), mais pas lui ajouter une image ou un texte.

L'instruction « ajoute une image bitmap dans le rectangle que tu viens de dessiner sur la scène » ne signifierait d'ailleurs rien. En revanche, si nous précisions « ajoute une image bitmap dans le clip que tu viens de créer à partir du rectangle que tu avais préalablement dessiné », cela aurait plus de sens : vous comprendriez qu'il faut éditer le symbole pour lui ajouter une image bitmap. Le clip est alors, dans ce cas, un conteneur d'objets d'affichage.

Voici une liste des différents types d'objets d'affichage.

- Shape : permet de tracer dynamiquement des droites, des courbes et des formes géométriques sur la scène (à l'aide des outils de l'interface ou en faisant appel à des instructions en AS3).
- Bitmap : permet d'afficher une image bitmap sur la scène (en important une image à partir du menu Fichier>Importer ou à l'aide d'instructions en AS3).

- TextField : permet de créer un texte dynamique sur la scène (à l'aide de l'outil Texte, disponible dans la barre d'outils de l'interface, ou d'instructions en AS3).
- Video : permet d'afficher une vidéo sur la scène (à l'aide du symbole de type Video ou d'instructions en AS3).

Les instances des classes `MorphShape`, `StaticText` et `SimpleButton` ne peuvent pas être créées dynamiquement à partir d'instructions ActionScript, mais elles constituent tout de même des objets d'affichage.

Un TextField

Il s'agit d'une zone de texte dont le contenu peut être modifié dynamiquement au cours du déroulement de l'animation. Ce changement peut être réalisé par le programme ou directement via l'interface de l'animation par une saisie de l'utilisateur.

Vous aurez compris qu'il faut distinguer deux types d'objets d'affichage : ceux qui servent à regrouper plusieurs objets d'affichage en un seul (les conteneurs) et ceux qui ne servent à contenir qu'un seul élément (un texte, une image, une vidéo, une forme).

La difficulté relative aux objets d'affichage n'est pas de différencier ces deux types pour savoir quand utiliser l'un ou l'autre : ce choix découlera directement de vos besoins. Il s'agit plutôt d'être capable de définir judicieusement l'ensemble des conteneurs de votre animation.

Différence entre un objet d'affichage et une occurrence

En AS1/2, les notions d'objet d'affichage et de conteneur d'objets d'affichage n'existaient pas : nous parlions alors, à tort, uniquement d'occurrence et d'instance.

Généralement, le terme occurrence faisait référence à la représentation graphique d'un symbole sur la scène, alors qu'une instance résultait de l'instanciation d'une classe. Malheureusement, ce que peu de personnes savaient, c'est qu'un glisser-déplacer d'un symbole sur la scène revenait à instancier celui-ci. Nous pouvions donc parler d'instance pour évoquer une occurrence. Certains développeurs maîtrisant correctement la programmation orientée objet utilisaient et utilisent toujours ce terme. Aujourd'hui, avec la nouvelle méthode qui permet de placer un symbole sur la scène à partir de l'ActionScript, la notion d'instance prend tout son sens.

Remarque

Le terme occurrence se traduit en anglais par instance !

Partons du postulat qu'une occurrence et une instance désignent la même chose, la représentation d'un symbole ou un exemplaire d'une classe.

Lorsque, par exemple, nous plaçons un symbole de type Clip sur la scène, parle-t-on d'objet d'affichage (et de quel type) ou d'occurrence/instance ?

Il s'agit en fait des deux, ce que résume le script suivant.

```
var monMessage = new TextField();
monMessage.text = "Bonjour";
addChild(monMessage);
monMessage.x=50;
monMessage.y=50;
```

Pour créer un texte dynamique ou de saisie sur la scène, nous créons une instance de la classe `TextField()`. Nous ajoutons ensuite cette instance à la liste d'affichage : elle fait donc partie de cette liste en tant qu'objet d'affichage. En définissant les propriétés `x` et `y` de l'instance `monMessage`, nous utilisons le fait que l'objet d'affichage est encore une instance.

Structurer une mise en page avec les classes Sprite() ou MovieClip()

Pour construire et gérer les différentes parties d'une animation, les utilisateurs de Flash, dont le profil est plutôt graphique, se servent généralement du scénario proposé dans l'interface du logiciel. Le déplacement de la tête de lecture fait alors apparaître les différents écrans de l'animation associés à des images-clés. Nous allons découvrir ici une autre technique, basée sur l'ActionScript, qui, bien qu'elle soit légèrement plus complexe et beaucoup plus abstraite, est plus efficace.

En effet, vous avez tout intérêt à gérer les changements d'affichage des différentes zones de votre animation en ajoutant et en supprimant, ou en affichant et en masquant, les objets d'affichage présents sur la scène. Cela sous-entend que l'analyse de l'interface de votre animation a permis de définir des zones principales et des zones secondaires.

Dans l'exemple ci-contre, vous pouvez constater que la mise en pages du site macgeneration est basée sur la grille que nous mettons en évidence dans la figure 2-4. Sans parler d'ergonomie, qui reste tout de même un objectif inéluctable lors de la construction d'une interface, vous noterez que la simplicité de lecture de cette page est due à une bonne structure.

Pour celles et ceux d'entre vous qui utilisent les CSS, vous comprendrez toute l'importance des développements suivants.

Avec l'arrivée de l'ActionScript 3, l'utilisation de la classe `Sprite()` nous permet d'organiser une mise en page sous la forme de blocs, c'est-à-dire des zones délimitant les différentes parties d'une animation. Par exemple, le découpage proposé dans la figure 2-4 (qui permet d'obtenir le résultat de la figure 2-3) contient des instances de la classe `Sprite()`.

Lorsque vous ferez appel à la classe `Sprite()` pour définir le point d'ancrage des différentes zones du quadrillage de votre mise en page, vous définirez ses coordonnées x et y.

Point d'ancrage d'une zone

Il s'agit du coin supérieur gauche d'un rectangle délimitant une zone.

Figure 2-3

La mise en forme de cette page est composée de zones.

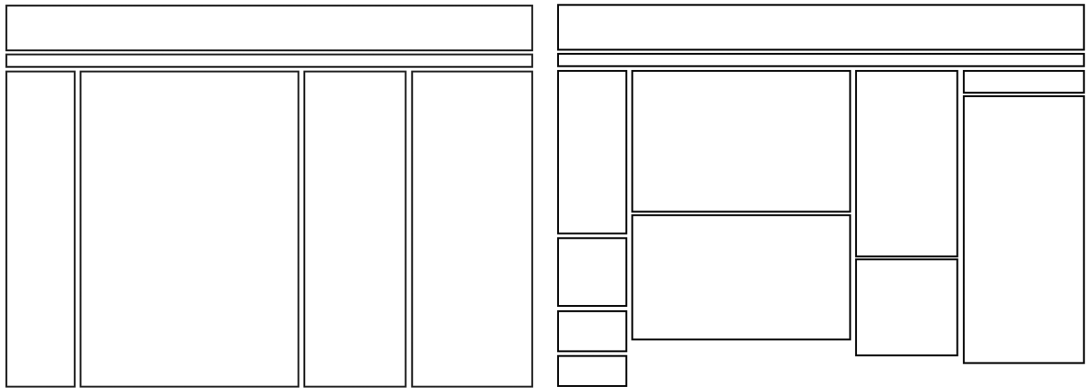


Figure 2-4

Le quadrillage de gauche met en évidence les zones principales du site macgeneration, à l'intérieur desquelles figurent des zones secondaires, présentées dans le quadrillage de droite.

La méthode addChild()

Cette méthode constitue l'action élémentaire pour placer dynamiquement un objet d'affichage ou un conteneur d'objets d'affichage sur la scène (ou dans une instance déjà présente sur la scène). En conséquence, l'objet ou le conteneur est ajouté à la liste d'affichage.

> **Note aux anciens développeurs en AS1/AS2**
>
> Les méthodes `createEmptyMovieClip()` et `attachMovie()` en AS1/AS2, ne peuvent plus être utilisées. Elles ont été partiellement remplacées par la méthode `addChild()`.

Lorsque vous aurez besoin de placer un symbole (qui possède un nom de classe) de la bibliothèque sur la scène, vous utiliserez la méthode `addChild()`.

Celle-ci vous servira également à placer dynamiquement un texte sur la scène, après avoir créé une instance de la classe `TextField()`.

Elle sera encore utile à l'instanciation d'une classe personnelle, dans le but de construire un objet d'affichage sur la scène.

Nous pourrions continuer ainsi l'énumération des cas où vous emploieriez la méthode `addChild()` ; non seulement, elle ne serait pas exhaustive, mais cela ne servirait à rien. Vous comprendriez vite que cette méthode a toujours le même rôle : rendre visible sur la scène des instances préalablement créées.

Maintenant que nous savons définir les objets d'affichage et les conteneurs d'objets d'affichage et que nous connaissons leur rôle, nous devons apprendre à les gérer dans une animation et donc faire appel à la méthode `addChild()`.

Tout d'abord, distinguons les instances que vous placerez directement sur la scène de celles que vous ajouterez dans des instances (des conteneurs d'objets d'affichage) déjà présentes sur la scène. En considérant l'exemple de la figure 2-5, nous devons commencer par créer directement 6 instances de la classe `Sprite()` sur la scène. Nous pourrons ensuite y ajouter des objets d'affichages.

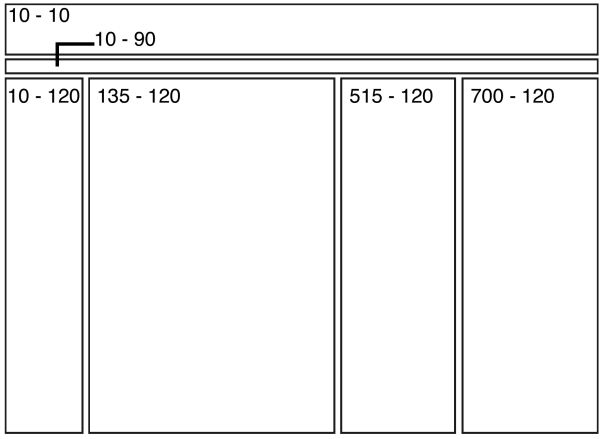


Figure 2-5

Les six zones de cette mise en page sont définies par des instances de la classe `Sprite()`.

Fichier de référence : `Chapitre2/addChildQuadrillage.fla`

```
var spEnTete:Sprite = new Sprite();
var spChapeau:Sprite = new Sprite();
var spMargeGauche:Sprite = new Sprite();
var spCorpsGauche:Sprite = new Sprite();
var spCorpsDroite:Sprite = new Sprite();
var spMargeDroite:Sprite = new Sprite();

addChild(spEnTete);
addChild(spChapeau);
addChild(spMargeGauche);
addChild(spCorpsGauche);
addChild(spCorpsDroite);
addChild(spMargeDroite);
```

Nous avons ajouté le préfixe `sp` devant chaque nom d'occurrence. Cela ne constitue en aucun cas une obligation, mais un simple repère visuel pour reconnaître qu'il s'agit d'une occurrence de la classe `Sprite()`.

Comme nous n'avons précisé aucune position, les instances sont placées par défaut en haut à gauche de la scène. Ajoutons les instructions suivantes pour obtenir la mise en page de la figure 2-5.

```
spEnTete.x =10;
spEnTete.y =10;

spChapeau.x =10;
spChapeau.y =90;

spMargeGauche.x =10;
spMargeGauche.y =120;

spCorpsGauche.x =135;
spCorpsGauche.y =120;

spCorpsDroite.x =515;
spCorpsDroite.y=120;

spMargeDroite.x =700;
spMargeDroite.y =120;
```

Les zones nommées dans notre interface étant définies, il est alors facile de programmer la suite du script. En nous basant sur la figure 2-4, nous allons ajouter deux zones (et non 4) pour pouvoir placer des textes.

```
var partieHaut:Sprite = new Sprite();
var partieBas:Sprite = new Sprite();
partieBas.y=100;

spMargeGauche.addChild(partieHaut);
spMargeGauche.addChild(partieBas);
```

Il est important de comprendre que la valeur 100, spécifiée à la troisième ligne du script, positionne l'instance à 220 pixels du haut de la scène, l'instance parent de l'instance `partieBas` se trouvant déjà à 120 pixels du haut de la scène.

> **Remarque**
> L'instruction `partieBas.y=100` peut être placée avant ou après la méthode `addChild()`.

Nous pouvons terminer cet exemple en ajoutant deux textes dans les instances `partieHaut` et `partieBas`.

```
var boutonAccueil:TextField = new TextField();
boutonAccueil.text="Accueil";
partieHaut.addChild(boutonAccueil);

var boutonContact:TextField = new TextField();
boutonContact.text="Contact";
partieBas.addChild(boutonContact);
```

Voici à présent une illustration du principal avantage de la liste d'affichage de l'AS3. Pour contrôler ou lire les propriétés de l'instance `boutonAccueil`, il est inutile de se référer à cette dernière en spécifiant un chemin à base de syntaxe pointée. Il suffit tout simplement d'écrire le nom de cette occurrence dans une instruction. Comme elle se trouve dans la liste d'affichage, sa position dans l'arborescence est parfaitement connue.

```
trace(boutonAccueil.text);
trace(boutonContact.text);
```

Ce script provoque l'affichage des mots « Accueil » et « Contact » dans la fenêtre Sortie de l'interface de Flash.

Dans la figure 2-1, nous vous avions présenté un exemple d'arborescence de la liste d'affichage. Voici à présent la représentation de notre animation.

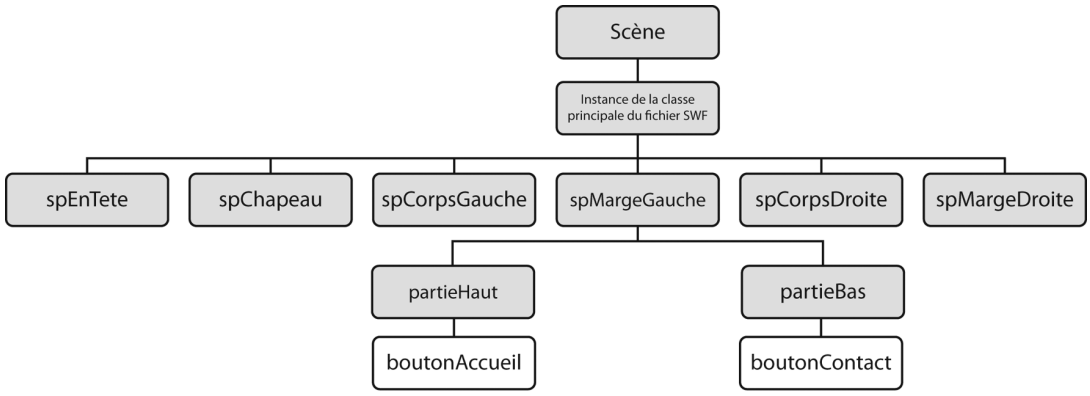


Figure 2-6
Arborescence de la liste d'affichage de notre animation

Rappelons tout de même qu'une mise en page d'instances sur la scène de Flash n'est pas toujours basée sur des grilles comme celles que nous venons de présenter. Voici un deuxième exemple, l'interface d'un DVD-Rom, où nous avons créé des instances de la classe `Sprite()` pour définir des zones qui se trouvent à différents endroits sur la scène. La mise en page obtenue est présentée dans la figure 2-7.

Figure 2-7

Cette mise en page ne peut pas s'appuyer sur un quadrillage, mais nous utilisons tout de même des instances de la classe `Sprite()` pour définir des zones.

La structure correspondante est exposée dans la figure 2-8.

Dans l'exemple de la figure 2-8, vous observerez que nous avons tracé des zones sous la forme de rectangles blancs transparents. En réalité, la largeur et la hauteur de ces blocs n'ont pas été définies car nous n'avons spécifié que l'emplacement des instances de la classe `Sprite()`. Ces positions sont représentées par les carrés noirs de la figure 2-8. Vous noterez que le rectangle situé en bas à gauche de l'interface possède son point d'ancrage à droite. En effet, nous avons attaché un texte dynamique à cette zone, mais avec un alignement de texte à droite.

Figure 2-8

Les blocs de cette mise en page facilitent le développement, notamment pour ajouter ou supprimer un objet d'affichage.

Voici à présent un dernier exemple qui résume tout ce que nous venons de voir avec une approche plus simplifiée.

Fichier de référence : `Chapitre2/addChild.fla`

```
var marge:Sprite = new Sprite();
addChild(marge);
var corps:Sprite = new Sprite();
addChild(corps);
corps.x=100;

var texte1:TextField = new TextField ();
texte1.text="Zone de\rla marge";
corps.addChild(texte1);

var texte2:TextField = new TextField ();
texte2.text="Zone du corps\rde la page";
marge.addChild(texte2);
```

Contrôler l'ajout d'objets d'affichage avec l'événement ADDED

Lorsque vous ajouterez un objet d'affichage à la liste d'affichage, vous aurez la possibilité d'exécuter une ou plusieurs actions telles qu'un test, un comptage, le réglage d'une propriété, etc. La technique est extrêmement simple car il suffit d'utiliser l'événement `ADDED`.

Dans l'exemple suivant, à chaque ajout d'un objet d'affichage dans l'instance `spMarge`, nous affichons, dans la fenêtre Sortie, le nombre d'enfants de cette instance.

```
var spMarge:Sprite = new Sprite();
addChild(spMarge);
spMarge.addEventListener(Event.ADDED,afficherInfo);

function afficherInfo(evt:Event) {
  trace(spMarge.numChildren);
}

var commande1:TextField = new TextField();
commande1.text="Accueil";
spMarge.addChild(commande1);


var commande2:TextField = new TextField();
commande2.text="Contact";
spMarge.addChild(commande2);
```

Cet événement s'avère très pratique pour mettre à jour certaines informations lors de chaque ajout d'un objet d'affichage à la liste d'affichage.

La propriété stage

Comme le précise très clairement le titre de cette section, la scène d'une animation est une propriété et non un objet directement accessible.

> **Remarque**
> Si le terme objet vous gêne et si vous avez déjà écrit des scripts en AS1/AS2, remplacez-le par le terme instance ou occurrence.

Avec l'AS1/AS2, il était possible de faire référence à la scène en tant qu'instance : elle représentait alors la racine d'un document. Rappelons que la classe `Stage()` était une classe de niveau supérieur. Lorsque nous écrivions le mot `Stage` (avec un S majuscule), nous faisions alors référence à la scène de l'animation Flash.

Pour afficher la largeur d'une occurrence dans la fenêtre Sortie de Flash nous devions écrire l'instruction suivante :

```
trace(nomDuneOccurrence._width);
```

La largeur de la scène pouvait être affichée à l'aide de l'instruction :

```
trace(Stage.width);
```

En ActionScript 3, toutes les instructions suivantes renvoient le même résultat, c'est-à-dire le chiffre 600 qui correspond à la largeur de la scène.

```
trace(stage.stageWidth);
trace(this.stage.stageWidth);
trace(rond.stage.stageWidth);
trace(carre.stage.stageWidth);
```

Pourquoi rencontre-t-on le mot stage en début de ligne dans le premier des quatre exemples ?

Aux 3e et 4e lignes de code, nous faisons référence à une instance particulière. Lorsque ce n'est pas le cas, comme à la première ligne, l'instance considérée est `this`.

Une animation désignée par `this` à la racine d'un script, possède ainsi une propriété `stage`. Toutes les occurrences placées sur la scène, que nous devons qualifier d'objet d'affichage (ou `DisplayObject`), possèdent également la propriété `stage` qui se réfère à la celle de l'animation.

En conséquence, si nous essayons d'afficher `this` à partir d'une image-clé de l'animation, nous obtenons :

```
trace(this);
[object MainTimeline]
```

Ce résultat peut se traduire par « objet de type timeline principale ».

Pour celles et ceux qui ont bien compris la notion de programmation objet, voici ce que renvoie le script suivant, qui se trouve dans la classe du document d'une animation.

```
package {
  import flash.display.Sprite;
  public class main extends Sprite {
    function main() {
      trace(this);
      trace(this.stage);
      trace(this.stage.stageWidth);
    }
  }
}
[object main]
[object Stage]
600
```

Remarque

La classe `Stage()` était une classe de niveau supérieur en AS1/AS2, ce qui n'est plus le cas en AS3.

Classes de niveau supérieur en AS1/2 :

`Accessibility, Array, AsBroadcaster, Boolean, Button, Camera, Color, ContextMenu, ContextMenuItem, CustomActions, Date, Error, Function, Key, LoadVars, LocalConnection, Math, Microphone, Mouse, MovieClip, MovieClipLoader, NetConnection, NetStream, Number, Object, PrintJob, Selection, Selection, SharedObject, Sound, Stage, String, System, TextField, TextFormat, TextSnapshot, Video, XML, XMLNode, XMLSocket, XMLUI.`

Classes de niveau supérieur en AS3 :

`ArgumentError, Array, Boolean, Class, Date, DefinitionError, Error, EvalError, int, Math, NameSpace, Number, Object, QName, RangeError, ReferenceError, RegExp, SecurityError, String, SyntaxError, TypeError, uint, URIError, Vector, VerifyError, XML.`

Supprimer un objet de la scène à l'aide de la méthode removeChild()

Remarque

Une occurrence qui a été obtenue sur la scène à partir d'un glisser-déplacer peut être supprimée à partir de la méthode `removeChild()` contrairement la méthode `removeMovieClip()` que nous utilisions en AS1/2.

Il est parfois nécessaire de supprimer un objet de la liste d'affichage.

Avant de vous proposer un exemple faisant appel à cette technique de suppression d'une instance sur la scène, il est important de préciser ou de rappeler que, dans certains cas, il sera plus judicieux de faire appel à la propriété `visible` que vous réglerez à `false` pour masquer temporairement une occurrence, plutôt que de la supprimer de la liste d'affichage.

La procédure de suppression d'un objet d'affichage de la liste d'affichage est extrêmement simple : elle se résume à l'exécution d'une seule instruction.

```
removeChild(cache1);
```

Remarque

L'occurrence ne sera plus visible sur la scène, mais elle conservera ses propriétés.

L'appel de cette méthode, sous-entend bien sûr l'exécution préalable des lignes d'instructions suivantes :

```
var cache1:Carte;
cache1 = new Carte();

addChild(cache1);
```

Que devient l'instance `cache1` après l'exécution de la méthode `removeChild()` ? En fait, elle existe toujours ; si vous demandez d'ajouter à nouveau cette instance à la liste d'affichage, elle est automatiquement replacée. Voici un exemple de synthèse des explications précédentes.

Fichier de référence : `Chapitre2/removeChild1.fla`

```
var cache1:Carte;
cache1 = new Carte();

addChild(cache1);

cache1.x = 250;
cache1.y =130;

btSuppression.addEventListener(MouseEvent.MOUSE_DOWN,retirerCache);

function retirerCache(evt:MouseEvent) {
  removeChild(cache1);
}
btReplacement.addEventListener(MouseEvent.MOUSE_DOWN,replacerCache);

function replacerCache(evt:MouseEvent) {
  addChild(cache1);
}
```

Si vous souhaitez supprimer l'objet d'affichage et plus précisément l'instance de la classe `Carte`, vous devez lui attribuer la valeur `null`, après l'avoir retirée de la liste d'affichage.

Contrôler la suppression d'objets d'affichage avec l'événement REMOVED_FROM_STAGE

Fichier de référence : `Chapitre2/removeChild4.fla`

Il est intéressant et important de savoir qu'à partir du moment où un objet d'affichage est supprimé de la liste d'affichage, l'événement `REMOVED_FROM_STAGE` est déclenché. Ainsi, dans l'exemple ci-après, cela permet de tenir une comptabilité des occurrences sur la scène.

```
var nbrCartes:Number = 3;
affichageNbrCartes.text= nbrCartes.toString();

bt1.addEventListener(Event.REMOVED_FROM_STAGE ,comptabiliserCarte);
bt2.addEventListener(Event.REMOVED_FROM_STAGE ,comptabiliserCarte);
bt3.addEventListener(Event.REMOVED_FROM_STAGE ,comptabiliserCarte);

bt1.addEventListener(MouseEvent.MOUSE_DOWN ,supprimerCarte);
bt2.addEventListener(MouseEvent.MOUSE_DOWN ,supprimerCarte);
bt3.addEventListener(MouseEvent.MOUSE_DOWN ,supprimerCarte);

function comptabiliserCarte(evt:Event) {
  nbrCartes--;
  affichageNbrCartes.text= nbrCartes.toString();
}

function supprimerCarte(evt:Event) {
  removeChild(DisplayObject(evt.currentTarget));
}
```

Remarque

Afin qu'il n'y ait aucune confusion, si vous souhaitez connaître le nombre d'objets d'affichage contenus dans un conteneur ou sur la scène, vous pouvez utiliser la propriété `numChildren`.

Ainsi, l'instruction suivante (à placer sur une image-clé) permet de connaître le nombre d'objets d'affichage contenus sur la scène.

```
trace(this.numChildren);
```

Un tel gestionnaire d'événement facilite la gestion de la suppression de certaines catégories d'occurrences dans des jeux.

Nous allons découvrir, dans un deuxième exemple, que la méthode `removeChild()` doit être précédée du nom du conteneur d'objets d'affichage qui contient l'objet d'affichage que nous souhaitons supprimer. Au cours de ce chapitre, nous avons découvert qu'il est judicieux de regrouper certaines occurrences au sein d'un même conteneur. Ainsi, vous devez préfixer la méthode `addChild()` d'un nom de conteneur. Ce nom devra également être placé devant `removeChild()`.

Dans cet exemple, nous créons une instance de la classe `Sprite()` pour accueillir 3 objets d'affichage. Puis, nous programmons un bouton qui, lorsqu'on clique dessus, retire tous les objets d'affichage contenus dans l'instance `tableJeu`.

Fichier de référence : `Chapitre2/removeChild2.fla`

```
var tableJeu:Sprite;
tableJeu = new Sprite();

addChild(tableJeu);

var cache1:Carte;
var cache2:Carte;
var cache3:Carte;

cache1 = new Carte();
cache2 = new Carte();
cache3 = new Carte();

tableJeu.addChild(cache1);
tableJeu.addChild(cache2);
tableJeu.addChild(cache3);

cache1.x = illustrationTrain.x;
cache1.y =130;

cache2.x = illustrationBateau.x;
cache2.y = 130;

cache3.x = illustrationAvion.x;
cache3.y = 130;
```

```
btSuppression.addEventListener(MouseEvent.MOUSE_DOWN,retirerCache);


function retirerCache(evt:MouseEvent) {
  tableJeu.removeChild(cache1);
  tableJeu.removeChild(cache2);
  tableJeu.removeChild(cache3);
}
```

Nous allons découvrir, dans un troisième et dernier exemple, une mauvaise surprise. Pour mieux la comprendre examinons la signature de la méthode `removeChild()`.

removeChild() méthode

```
public function removeChild(child:DisplayObject):DisplayObject
```

Figure 2-9

La méthode removeChild() prend pour paramètre un nom de type DisplayObject, c'est-à-dire un objet d'affichage (ou celui d'un conteneur d'objets d'affichage).

Observez bien le type d'information contenu entre les parenthèses ! La méthode prend en paramètre le nom d'un objet d'affichage. Dans ce cas, lorsque nous programmons une fonction de rappel (callBack) qui fait référence à la propriété `currentTarget` pour identifier un objet d'affichage à supprimer, comment préciser qu'il s'agit d'un objet d'affichage ?

Dans l'exemple suivant, la ligne d'instruction ne peut pas être exécutée.

```
removeChild(evt.currentTarget);
```

Vous devez donc faire appel à la classe `DisplayObject()` pour préciser au compilateur que le paramètre `evt.currentTarget` est bien un objet d'affichage.

```
removeChild(DisplayObject(evt.currentTarget));
```

Voici un exemple complet qui illustre notre propos.

Fichier de référence : `Chapitre2/removeChild3.fla`

```
var cache1:Carte;
var cache2:Carte;
var cache3:Carte;

cache1 = new Carte();
cache2 = new Carte();
cache3 = new Carte();

addChild(cache1);
addChild(cache2);
addChild(cache3);

cache1.x = illustrationTrain.x;
```

```
cache1.y =130;

cache2.x = illustrationBateau.x;
cache2.y = 130;

cache3.x = illustrationAvion.x;
cache3.y = 130;

cache1.addEventListener(MouseEvent.MOUSE_DOWN,retirerCache);
cache2.addEventListener(MouseEvent.MOUSE_DOWN,retirerCache);
cache3.addEventListener(MouseEvent.MOUSE_DOWN,retirerCache);

function retirerCache(evt:MouseEvent) {
  removeChild(DisplayObject(evt.currentTarget));
}
```

removeChildAt()

Cette dernière méthode vous permettra de supprimer un objet d'affichage en spécifiant un numéro d'index. Afin de mieux comprendre le fonctionnement des index liés aux objets d'affichage, consultez le chapitre 4 de ce livre et plus particulièrement la section « Gérer les plans entre deux ou plusieurs occurrences ».