Les éditions Eyrolles
présentent

LA VIDÉO DANS FLASH

par

David Tardiveau

EYROLLES

Remerciements

Pour illustrer les possibilités vidéo dont il est question dans ce livre, j'ai préféré partir d'une vidéo agréable à visionner plusieurs fois plutôt que d'utiliser un film de vacances personnel, voire ennuyeux. Mon choix s'est donc arrêté sur la vidéo d'un film d'animation de Gobelins, l'école de l'image.

Lionel, *un film d'animation que nous utiliserons tout au long de cet ouvrage.*

Lionel est un film réalisé par quatre jeunes créateurs dans le cadre de leurs études dans cette école. Si vous souhaitez les contacter pour des raisons professionnelles, voici leurs coordonnées :

Gabriel Gelade : gabien@hotmail.fr

Anthony Menard : authong@hotmail.com

Matthieu Poirey : mpoirey@framestore-cfc.com

Mehdi Leffad : mehdi_@msn.com

Encore merci à tous les quatre de m'avoir autorisé à promouvoir votre travail en utilisant votre animation pour démontrer les possibilités en vidéo proposées dans Flash.

Je tiens également à remercier Tom, le petit frère de Noé, pour sa collaboration et son jeu d'acteur magnifique ! C'était la *guest star* de ce livre avec son passage éclair sur fond bleu.

Avant-propos

À propos des animations décrites dans cet ouvrage

Vous avez sûrement consulté quelques pages de ce livre avant de prendre la décision de l'acheter. Vous avez donc remarqué qu'il présente de nombreux exemples courts et précis, car il vaut mieux étayer notre propos par des animations simples, qui ne font appel qu'à une seule notion à la fois. En effet, des exemples trop complexes nuiraient à la démonstration : lorsque vous ouvrez un fichier `.fla`, il est toujours fastidieux de rechercher la ou les lignes de code qui illustrent la fonctionnalité dont il est question…

À qui s'adresse cet ouvrage ?

Cet ouvrage s'adresse à divers types de publics que nous pourrions classer ainsi :

- les professionnels de la vidéo qui souhaitent diffuser une vidéo sur Internet ;
- les utilisateurs (intégrateurs et développeurs) de Flash qui souhaitent étendre leurs compétences dans le domaine de l'ActionScript en matière de vidéo ;
- les entreprises et autres entités du secteur de la communication qui souhaitent développer leurs services en matière de diffusion de contenu multimédia sur Internet ou sur support off-line.

D'une manière générale, il est conseillé à chaque lecteur de connaître ces notions élémentaires du langage ActionScript :

- l'emplacement d'un script dans une animation ;
- la notion de gestionnaire d'événement ;
- la notion de propriété ;
- la notion de variable.

Si vous ne maîtrisez pas ces notions, de nombreux sites vous proposent des tutoriaux pour vous mettre à niveau (les premières pages du site *www.yazo.net* abordent ainsi toutes ces connaissances élémentaires).

Autre question : est-ce que votre apprentissage de la vidéo dans Flash va se faire en AS1, AS2, AS3 ? Quelle version du langage doit-on utiliser ? Nous n'allons pas entrer une fois

de plus dans la polémique, mais voici mon point de vue. En tant qu'auteur, et avant tout enseignant, je sais à quel point il est difficile d'apprendre un langage de programmation. Si vous achetez ce livre, c'est parce que vous souhaitez connaître les différentes techniques permettant d'insérer et contrôler une vidéo sur la scène. Vous désirez gagner du temps et éviter d'effectuer vous-même des recherches sur la gestion de ce média dans Flash. Il se peut également que vous fassiez vos premiers pas en matière de développement avec ce langage qu'est l'ActionScript et que le « vocabulaire » qui entoure le composant vidéo vous effraie un peu.

Pour être très honnête, la gestion de la vidéo dans Flash se fait principalement par le biais d'un composant qui fait appel à l'AS2. Vous ne connaissez rien à l'AS2 ? Pas de craintes à avoir, nous allons volontairement simplifier nos scripts et utiliser une syntaxe en AS1 pour que vous compreniez l'ensemble des scripts présentés dans cet ouvrage. Pour celles et ceux qui utilisent l'AS2 ou l'AS3, vous n'aurez qu'à utiliser les lignes de code proposées dans tous les exemples de ce livre pour les replacer dans les classes que vous créerez vous-même. Vous typerez également vos instances et autres variables.

Se faire assister par Flash ou programmer soi-même ?

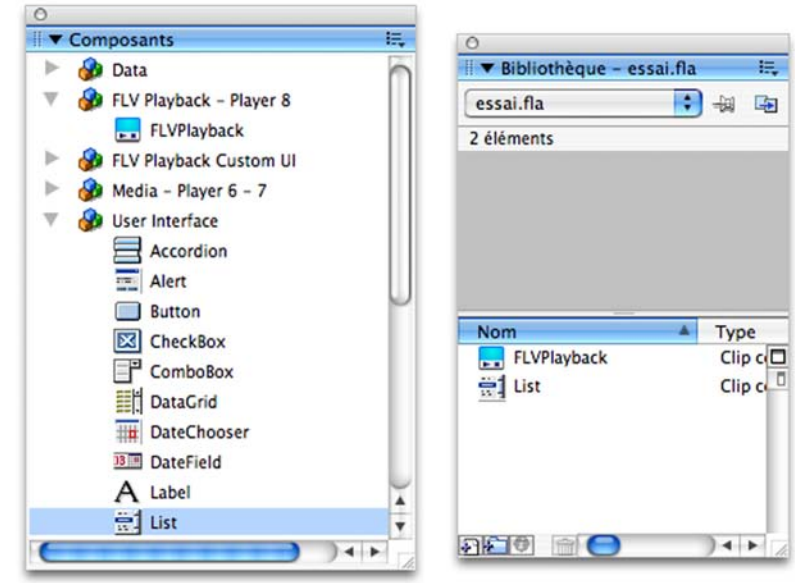
Pour placer une vidéo sur la scène, il existe plusieurs techniques, dont une qui ne nécessite pas une grande connaissance de Flash. À la fin du chapitre 1, vous découvrirez en effet une procédure dans laquelle le logiciel vous assiste intégralement pour le placement d'une vidéo sur la scène. Précisons tout de même que si vous optiez pour cette solution, vous vous restreindriez dès le départ dans le réglage de certaines options. Plus précisément, cette technique présente des limites que vous ne pourrez pas dépasser si vous ne faites pas appel à l'ActionScript.

Pourquoi utiliser le composant FLV Playback plutôt qu'un symbole vidéo pour intégrer une vidéo ?

Il existe deux méthodes pour intégrer une vidéo dans Flash, chacune présentant bien sûr des avantages et des inconvénients. Il est vrai qu'il est très simple de copier-coller ou d'écrire les cinq lignes de code nécessaires à la lecture d'une vidéo au travers d'un symbole de même nom, mais dès que vous allez vouloir contrôler la lecture de votre séquence, vous allez découvrir qu'il existe certaines limites. Certes, au premier abord et sans explications, l'emploi du composant FLV Playback peut rebuter car vous vous retrouvez face à un panneau Paramètres contenant justement de nombreux réglages dont les intitulés ne sont pas forcément explicites. De plus, en consultant l'aide en ligne du fonctionnement de ce composant, on accède très rapidement à des lignes de code parfois difficiles à comprendre pour les non-initiés. Cette notion même de composant en Flash qui se veut simple et justement destinée aux novices n'est donc pas si facile à appréhender si on ne comprend pas l'avantage qu'offrent les composants. C'est l'occasion de définir ici le fonctionnement d'un composant.

Que ce soit dans la bibliothèque de votre animation (car vous pouvez vous-même en créer une) ou dans celle qui s'intitule Composants et qui s'affiche par le biais du menu Fenêtre, vous constaterez que certains symboles ne sont pas de type Clip, Bouton ou Graphique : ce sont des composants dont l'icône diffère de celle des types précités (à l'exception des images, sons et polices de caractères). Le principe d'un composant est simple puisqu'il s'agit d'un symbole préprogrammé (qui contient du code sur sa *timeline*), permettant ainsi d'être simplement glissé puis déplacé sur la scène. Il ne reste plus qu'à renseigner quelques paramètres dans la palette Paramètres. Un composant est donc destiné à un usage répétitif et simplifie la gestion du symbole qu'il représente. Il offre aussi la possibilité de répartir le travail entre le développeur qui programme un symbole (le composant) et l'intégrateur qui l'utilise.

Les composants possèdent une icône particulière.



Le composant FLV Playback simplifie donc l'insertion d'une vidéo dans une animation, mais également son contrôle : lecture en boucle, automatique, choix du type du contrôleur, redimensionnement automatique… Vous découvrirez au travers de ce livre que le vocabulaire (méthodes et propriétés) lié à ce composant est d'une grande richesse et permet de contrôler parfaitement ce média.

Typage et déclaration des variables

Vous découvrirez certains scripts qui posséderont des lignes de code plus complexes que d'autres qui aboutissent au même résultat. En écrivant ce livre, le souci permanent d'être compris par un maximum de lecteurs m'a en effet poussé à alterner la rédaction de scripts

contenant des déclarations de variables et des typages, avec d'autres scripts ne présentant ni les unes ni les autres. Pour les développeurs plus expérimentés, gardez à l'esprit que la déclaration d'une variable qui vous semble logique et évidente (nous ne parlons pas de l'initialisation) ne l'est pas forcément pour des débutants.

Table des matières

Introduction

Depuis les débuts d'Internet, les possibilités techniques d'insertion d'une vidéo dans une page Web n'ont pas été très nombreuses. Avant de vous présenter l'évolution qu'a connue la vidéo sur le Web, rappelons qu'un navigateur n'est pas capable de lire nativement un flux vidéo. L'internaute a besoin de télécharger un plug-in pour étendre les capacités de son logiciel de navigation. Il s'agit d'un ou plusieurs fichiers qui vont relayer le navigateur pour lire le flux vidéo qui arrive sur un ordinateur connecté à Internet.

> **Remarque**
>
> N'oubliez pas qu'une vidéo Flash peut être aussi lue en local, c'est-à-dire sur votre ordinateur ou sur un DVD-Rom (ou sur tout autre support amovible).

La société Apple a été une des toutes premières à proposer un plug-in afin de pouvoir lire des séquences QuickTime en ligne. Malheureusement, les débits d'Internet, qui à la fin des années 1990 n'étaient pas très élevés, ont constitué une première difficulté, insurmontable à l'époque, à la consultation de séquences en ligne.

Les sociétés Microsoft et Real sont très rapidement entrées dans la course à la diffusion de flux audio et vidéo sur le Web. Au début des années 2000, le plug-in Real-audio a connu un très grand succès, et ce dernier est même devenu un standard pendant quelque temps. Cependant, comme il a été très difficile durant plusieurs mois de découvrir le lien de téléchargement de ce plug-in, la société Real a perdu de nombreuses parts de marché. Aujourd'hui, ce point a été simplifié de sorte qu'il ne faut pas plus de trois écrans avant d'accéder au lien du téléchargement de l'archive à partir de la page d'accueil (son accès n'est toutefois pas mis en évidence sur cette page).

La société Microsoft a quant à elle tenté d'imposer durant plusieurs années ses différents formats lisibles avec son *player* mais elle n'a que partiellement réussi. Relativisons notre propos en précisant que de nombreux fichiers aux extensions WMA et WMV circulent sur Internet, mais qu'il s'agit généralement de vidéos à télécharger, bien plus rarement à consulter en ligne. Dans ce dernier cas, l'intégration du média dans l'interface du site n'est pas des plus esthétiques.

Les problèmes de débit et le manque de standardisation suffisent donc à expliquer les débuts difficiles de la vidéo sur Internet. Aujourd'hui, on la trouve partout ! Dans les

spots publicitaires des pages d'accueil des sites à fort trafic, en encart ou en plein écran, incrustée (détourée), dans les pages d'aide, sur les sites de diffusion de flux, etc. De Microsoft, Adobe (anciennement Macromedia) et Apple, qui a réussi à s'imposer ?

Il suffit de parcourir le Web pour découvrir qu'à chaque clic droit sur une vidéo, un menu local déroulant apparaît et affiche « À propos de Macromedia Flash Player 9 » ou « À propos de Macromedia Flash Player 8 » (tout dépend de la version de votre player Flash). Comment expliquer un tel succès ?

Flash ne gère correctement la vidéo que depuis 2002, et pourtant, lorsque nous voyons une publicité vidéo sur Internet, il s'agit neuf fois sur dix d'un fichier FLV. Comment expliquer que le format vidéo de Flash ait pu détrôner ceux des deux géants de l'informatique Apple et Microsoft, sans parler de Real ?

Cette suprématie peut s'expliquer par la nature du codec VP6, proposé par la société On2, qui permet de générer des fichiers vidéo lisibles dans Flash d'une très bonne qualité (laquelle s'évalue en comparant la qualité de l'image par rapport au poids du fichier). Par ailleurs, le taux d'implantation du plug-in Flash est nettement supérieur à ceux d'Apple et de Real. Depuis quelques années, lors de l'installation d'un système d'exploitation estampillé Apple ou Windows, l'implantation du player Flash est systématiquement proposée. Les pages Web faisant appel à cette technologie sont très nombreuses sur la toile, et de ce fait, les internautes sont très souvent sollicités et finissent généralement par installer le plug-in Flash, s'il ne l'est pas par défaut. Adobe annonce un taux d'équipement du player de plus de 97 %. Bien sûr, il faut relativiser ce chiffre et la société elle-même précise que ce pourcentage tient compte des différentes versions installées. Il serait absurde de vous présenter des statistiques aujourd'hui, car lors de la publication de ce livre, elles auront bien évidemment évolué. Généralement, il faut compter 6 à 9 mois après la sortie d'une nouvelle version de Flash pour que plus de 50 à 70 % des machines connectées à Internet voient leur plug-in mis à jour.

Partant du constat que la vidéo en Flash est très utilisée aujourd'hui et qu'elle est généralement placée au milieu d'une page comme simple média comparable à une image fixe, l'idée nous est venue de publier un ouvrage démontrant et décrivant toutes les possibilités de manipulation d'une vidéo dans Flash.

1

Préparer et encoder une vidéo au format FLV

Vous allez découvrir plus loin dans ce livre que la technique d'encodage d'une vidéo s'exécute très facilement et rapidement. Cependant, cela ne se fait pas n'importe comment et nécessite donc un travail de préparation. C'est pourquoi nous allons commencer par apprendre à :

- lire les informations relatives à une vidéo ;
- effectuer un montage rapide d'une vidéo ;
- connaître les contraintes d'encodage.

Lorsque vous saurez maîtriser ces étapes, vous pourrez alors passer à l'étape finale : l'encodage de la vidéo.

Si vous êtes un professionnel de la vidéo, vous pouvez passer directement au chapitre 2, car nous allons aborder dans les pages qui vont suivre, les techniques de montage simplifiées d'une séquence vidéo ainsi que les contraintes d'encodage.

Les formats standards vidéo

Avant d'apprendre à manipuler un fichier informatique contenant de la vidéo, vous devez acquérir quelques bases dans ce domaine qui existe depuis de nombreuses années, bien avant la démocratisation des micro-ordinateurs. Pour illustrer notre propos, nous pourrions en effet vous poser les questions suivantes :

- Vous savez manipuler un caméscope et enregistrer une séquence, mais savez-vous combien pèse 1 mn de film au format DV ou HDV ?

- Quelles sont les dimensions (largeur et hauteur) d'une séquence vidéo ?
- Combien d'images contient une seconde de vidéo ?

Nous pourrions continuer notre énumération, mais développons plutôt tous ces points.

Pour commencer, gardez toujours à l'esprit qu'une seconde de vidéo au format DV occupe un espace de 3,43 Mo/s en PAL, c'est-à-dire avec des dimensions d'image de 720 × 576 pixels. Une minute de vidéo au format DV génère donc un fichier de 206 Mo.

Au même titre qu'il existe à travers le monde, plusieurs langues, plusieurs systèmes monétaires ou unités de mesure, il existe également deux formats vidéo. Le PAL est utilisé en Europe alors que le NTSC l'est aux États-Unis et au Japon. Cela entraîne une différence de taille d'image et de cadence.

Tableau 1-1 Les formats PAL et NTSC présentent des caractéristiques différentes, ce qui entraîne des réglages différents pour la compression d'une vidéo.

Format de l'image	Taille de l'image	Nombre d'images par seconde (Cadence)
PAL (Europe)	720 × 576	25
NTSC (USA, Japon)	720 × 480	29,97

Si vous lisez ce livre alors que vous vous trouvez sur le sol européen, lorsque vous numériserez vos vidéos, vous obtenez des fichiers avec une image de 720 × 576 pixels avec 25 images par seconde.

Le format auquel nous faisons référence est celui qui est utilisé en vidéo, mais lorsque vous aurez capté vos séquences, il faudra que vous vous posiez la question suivante :

« Quel sera le support final de mon montage ? Sur bande ou un fichier informatique ? »

Si vous numérisez une cassette vidéo, c'est que vous avez sûrement besoin de travailler sur ordinateur pour créer un film. Ensuite, deux solutions s'offrent à vous. Ou bien vous retransferez votre création sur bande vidéo ou bien vous générez un fichier informatique. D'ici quelques années, la bande devrait être amenée à disparaître, mais nous n'en sommes pas encore là. Si votre montage est destiné à être diffusé sur un écran d'ordinateur, vous ne pourrez pas conserver le format 720 × 576. En effet, la lecture d'une séance avec un tel format (ou rapport) dans les logiciels de montage vidéo ne présente aucune déformation car ils compensent l'anamorphose. En revanche dans certains cas, vous observerez un étirement de vos images. Si vous faites un tel constat alors que vous n'utilisez pas un logiciel de montage vidéo, c'est qu'il faut corriger le rapport 720 × 576 par 768 × 576. Ainsi, la consultation d'une séquence vidéo numérisée ne sera plus anamorphosée.

Remarque

Numériser une vidéo signifie transférer le contenu d'une bande d'une cassette DV (ou autre) sur un support informatique (disque dur ou tout autre support). Ce verbe a été retenu car c'est celui qu'on a initialement utilisé pour numériser une image, c'est-à-dire passer de l'état image sur papier à celui de fichier informatique. Ce sont principalement les personnes qui viennent du monde du multimédia ou de la photo qui utilisent ce terme. Les spécialistes de la vidéo utilisent quant à eux le verbe « capter ».

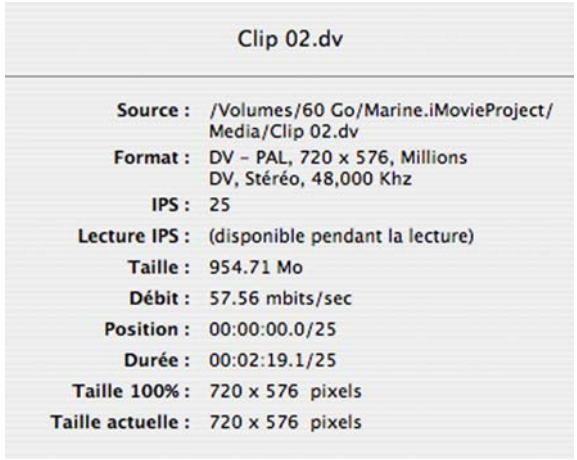


Figure 1-1

La fenêtre Informations sur la séquence du logiciel QuickTime Player Pro nous permet d'obtenir les caractéristiques d'un fichier.

Lorsque vous numériserez vos vidéos ou récupérerez des fichiers DV, quel poids peut-on espérer obtenir après compression ? Comme vous allez pouvoir le constater dans le développement suivant, tout dépend du codec. Gardons à l'esprit que nous travaillons la compression de nos vidéos dans le but de les intégrer dans Flash. Il n'existe dans ce cas que deux codecs de compression. Le Sorenson Spark et le On2 VP6. Le premier des deux commence à présent à dater un petit peu et propose une qualité d'image trop mauvaise en comparaison du second. Dans le cas où vous devriez conserver une compatibilité avec les anciens players Flash 6 et 7, vous n'aurez malheureusement pas le choix et serez obligé d'utiliser ce codec.

À l'occasion de la rédaction de ce livre, nous avons fait des essais à partir de 4 séquences (avec des rythmes différents). D'une façon générale, nous pouvons constater qu'avec ces deux codecs, la compression est impressionnante : le rapport de l'espace occupé entre le fichier compressé et le fichier original varie de 1,3 à 2 %. En partant d'un fichier de 80 Mo, nous arrivons à un poids d'un méga-octets. Il n'y a pas de différence significative entre les deux codecs. Proportionnellement, les poids sont relativement identiques. En revanche, la différence de qualité est flagrante. En analysant deux séquences compressées avec les codecs Sorenson Squeeze et On2 VP6, on constate qu'il n'y a pas de pixellisation dans les mouvements avec ce dernier.

Les valeurs du tableau ci-après ne vous sont présentées qu'à titre indicatif et ne constituent en aucun cas une référence de calculs. L'objectif est simplement de vous donner un ordre de grandeur des poids de fichiers.

Tableau 1-2 Le poids des vidéos compressées représente en moyenne 2 % du fichier d'origine.

Taille DV (720 × 576)						
	Fichier DV		VP6		Sorenson pour Flash	
	Mo	Temps	Mo	Poids/DV	Mo	Poids/DV
Fond vert/interview	84,7	24 secondes	1,3	1,53 %	1,2	1,42 %
Défilé de mode (caméra sur pied)	80,9	23 secondes	1,2	1,48 %	1,3	1,61 %
Enfant en mouvement	480,1	139 secondes	7,4	1,54 %	7,2	1,50 %
Travelling sur une foule	50,4	14 secondes	0,74	1,47 %	1,1	2,18 %

Le format HDV

Dans les mois et années à venir, nous allons de plus en plus entendre parler du format HDV. À l'heure où nous écrivons ce livre, ce format est proposé à la télévision et par certains FAI (fournisseurs d'accès à Internet), mais les caméras en HDV ne sont pas très nombreuses et sont peu répandues. Elles ne coûtent pourtant pas très cher car les premiers prix pour le grand public avoisinent les 1 500 euros. Comme vous le savez peut-être le format HDV propose non seulement une meilleure qualité d'image, mais surtout une taille plus importante. Depuis plusieurs mois, ce format est représenté par deux standards : le 1080i et le 720p. Nous ne présenterons pas en détail la différence qui les oppose dans le monde audiovisuel, mais il semblerait, paradoxalement, que le 1080i s'impose progressivement en France.

Une image peut avoir deux dimensions selon les caméras : en 720p (p pour progressif), la résolution est réellement de 1 280 × 720 pixels. En 1080i (i pour *interlaced* ou entrelacé), elle est de 1 440 × 1 080 pixels. La largeur est anamorphosée afin d'obtenir un affichage de 1 920 × 1 080 pixels. Si ce format d'avenir vous intéresse, nous vous invitons à consulter le site http://www.dvforever.com.

Les autres formats

Parfois, vous aurez peut-être des séquences aux dimensions différentes de celles que nous avons évoquées. N'oubliez pas que la vidéo n'est pas la seule technique de diffusion d'une série d'images. Le cinéma existe encore ! Toute plaisanterie mise à part, nous vous invitons à consulter les liens ci-dessous pour en savoir davantage sur les différents formats existants.

Le garage de la vidéo sur Mac

La page de ce site vous présente non seulement les formats qui existent en vidéo, au cinéma, à la télévision et sur informatique, mais vous découvrirez également les techniques de recadrage.

http://www.garage-video.com/articles/format.html

TDT3D

Ce site vous propose une série de tableaux sur les différents formats avec de nombreuses données chiffrées.

http://www.tdt3d.be/articles.php?art=videos_field

H264, Sorenson Spark et On2 VP6

Quels sont les critères qui permettent de juger de la bonne qualité d'un codec ? La réponse à cette question est très simple, car le constat est frappant : la pixellisation. Plus vous diminuez le débit exprimé en Kbits/s plus vous obtiendrez des carrés « grossiers » pour la représentation de l'image.

Figure 1-2

La copie d'écran ci-dessous à gauche est extraite d'une séquence avec une faible compression. Un zoom démontre la netteté. À droite, on peut constater qu'après une compression plus forte (un débit plus faible) les formes sont moins précises.

Comment expliquer une telle différence de qualité ? D'une façon générale, nous pouvons dire que plus un codec est ancien, plus la pixellisation est forte. À l'heure où ce livre est écrit, nous sommes en août 2006, les deux derniers codecs qui présentent un très bon rapport qualité/poids sont le H264 de QuickTime et le VP6 de On2 développé pour Flash 8. Avant eux, à l'époque de Flash MX 2004, le codec Sorenson Squeeze proposait une très bonne qualité, mais nettement inférieure à celle de On2 VP6.

La qualité d'une séquence ne se résume pas uniquement à la qualité graphique, mais également à sa fluidité. Plus le nombre d'images est important, plus la séquence sera fluide, mais également volumineuse.

Un bon codec est donc un algorithme de compression et de décompression capable de réduire le poids d'un fichier sans pour autant détériorer la qualité de l'image. Il doit également être capable de lire de façon fluide les images compressées.

Le codec H264 proposé par Apple est celui qui propose tout de même la meilleure qualité de compression. Pour un fichier de même poids, on constate tout de même une différence

de rendu entre ce codec et le VP6 de On2, notamment dans les mouvements de caméra et les fondus. Ces deux derniers critères sont ceux que vous devez retenir pour comparer la qualité de différentes séquences.

Lire les informations relatives à une vidéo

Avant de compresser votre vidéo, vous aurez besoin de connaître certaines de ses caractéristiques telles que les dimensions et la cadence. En effet, nous allons devoir effectuer certains réglages de compression en fonction de ces informations.

Comme il existe différents types de formats vidéo, tous les logiciels ne peuvent pas lire tous les fichiers vidéo. Pour ce premier chapitre, nous allons partir du principe que vous travaillez la vidéo dans un processus (une chaîne de production) professionnel, c'est-à-dire avec Première, FinalCut, Avid ou encore AfterEffects. Tous ces logiciels ont besoin de QuickTime pour fonctionner et c'est justement ce player que nous allons utiliser. Il est conseillé d'utiliser la version Pro ; enregistrez donc votre version si celle-ci est en standard.

> **Remarque**
>
> Pour enregistrer votre version de QuickTime, sélectionnez l'option Enregistrement dans les préférences du logiciel. Vous devez alors spécifier un nom, un nom d'entreprise (facultatif) et le numéro de série qui vous a été communiqué après avoir payé votre licence auprès d'Apple.

Pour obtenir des informations sur une séquence (une vidéo), suivez la procédure ci-dessous :

1. Ouvrez un fichier (Fichier/Ouvrir...).
2. Dans le menu Fenêtre, activez la commande Information sur la séquence.
3. La fenêtre ci-dessous apparaît alors.

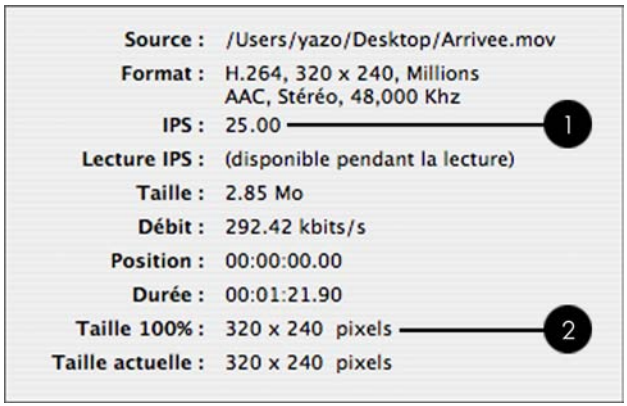


Figure 1-3

Les informations relatives à la cadence (IPS) et à la taille (dimensions) du fichier doivent être connues avant la phase d'encodage.

Comme vous l'indique cette copie d'écran, nous allons avoir besoin de la cadence et des dimensions de la vidéo pour procéder à l'encodage.

Montage express d'une vidéo avec QuickTime Pro

> **Remarque**
> Vous devez enregistrer QuickTime en version Pro pour pouvoir suivre cette partie.

Comme nous vous le précisions en introduction à ce chapitre, si vous connaissez déjà un logiciel tel que Première ou FinalCut, vous pouvez passer cette partie du livre car vous maîtrisez déjà les techniques nécessaires pour monter une séquence.

Si vous avez la chance de récupérer de la part de votre commanditaire (ou une autre source) une vidéo prête à être encodée, c'est un gain de temps non négligeable. Dans le cas contraire, vous allez devoir procéder à quelques manipulations.

> **Important**
> Ne travaillez jamais sur les fichiers originaux qui vous ont été fournis. Effectuez une copie et utilisez-la pour vos montages.

Imaginons que vous possédiez trois séquences vidéo, une image, un texte et un fichier son que vous souhaitiez monter. Nous allons apprendre à réaliser un montage au travers des étapes suivantes :

- Effacer le début et la fin d'une séquence.
- Ajouter une image sur une vidéo.
- Supprimer une piste d'une vidéo.
- Extraire une piste.
- Ajouter un texte.
- Ajouter un son.
- Ajouter une vidéo par-dessus une autre.
- Faire pivoter la vidéo.
- Monter plusieurs séquences les unes après les autres.
- Utiliser un fichier texte pour ajouter un texte à une séquence.

Toutes ces étapes pourraient être réalisées directement dans Flash, mais dans certains cas, il est plus pratique de le faire directement dans la vidéo.

Pourquoi avoir choisi QuickTime pour ce genre de manipulations ? Tout simplement parce que QuickTime Pro est la solution la plus simple et la plus rapide pour réaliser proprement un montage.

Commençons donc par voir comment il est possible de supprimer les premières et dernières secondes d'une séquence. En effet, lorsque vous filmez et/ou numérisez des sources vidéos, vous ne commencez pas et ne vous arrêtez pas sur les « bonnes » images de début et de fin. Vous devez donc définir, ce qu'on appelle en vidéo, le *point d'entrée* et le *point de sortie*. Pour notre exemple, nous sommes partis d'une séquence d'un enfant qui a été filmé en train de jouer. Les premières images du fichier à partir duquel nous avons travaillé correspondaient aux dernières images de la séquence précédente sur la bande de la cassette. À la fin de la séquence, le cadreur a souhaité se rapprocher du sujet en changeant de focale (en zoomant) plutôt que de déplacer la caméra, mais la fin de la prise de vue a été ratée, et il faut donc supprimer le mouvement final. Les copies d'écran suivantes vous présentent la séquence sous forme de pellicule.

Figure 1-4

Les premières secondes de la séquence correspondent aux images de la séquence précédente sur la bande d'origine. Il faut donc supprimer les premières secondes.

Figure 1-5

La séquence finale doit correspondre au milieu de la séquence actuelle.

Figure 1-6

Les dernières secondes contiennent également des images qu'il faut supprimer.

Effacer le début et la fin d'une séquence

Nous allons apprendre à supprimer les images de début et de fin de séquence. Gardez toujours à l'esprit que cette technique est valable également dans le cas où vous souhaiteriez supprimer une partie de la séquence qui se trouve au milieu de la vidéo.

Pour supprimer une série d'images dans une séquence QuickTime, vous devez suivre la procédure suivante :

1. Ouvrez un fichier (Fichier/Ouvrir…).

2. Commencez par effectuer une première lecture.

Astuce

Utilisez la barre d'espace de votre clavier pour lancer la lecture et la mettre en pause. Chaque pression sur cette touche alterne la lecture et la pause d'une séquence.

3. Replacez votre curseur de lecture au début de la séquence.

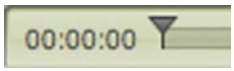


Figure 1-7

La tête de lecture est représentée par le petit triangle noir qui défile de gauche à droite de la fenêtre de la séquence.

4. Maintenez la touche Majuscule (Shift) enfoncée et appuyez sur la barre d'espace. Dès que vous arrivez sur l'image que vous souhaitez conserver, appuyez à nouveau sur la barre d'espace. Utilisez alors les touches fléchées gauche et droite de votre clavier pour allonger ou raccourcir la sélection.

Figure 1-8

La sélection de la zone à supprimer apparaît sous la forme d'une barre grise foncée.

5. Appuyez sur la touche Suppr de votre clavier pour supprimer la sélection en cours.

Pour supprimer la fin de la séquence la technique est la même, mais détaillons-la tout de même :

1. Déplacez votre curseur en le faisant glisser avec votre souris ou laissez-le se placer en effectuant une nouvelle lecture de votre séquence.

Figure 1-9

La tête de lecture doit se trouver à droite de la jauge de lecture.

2. Maintenez la touche Majuscule (Shift) enfoncée et appuyez sur la barre d'espace. Lorsque la tête de lecture arrivera à la fin de la séquence, elle s'arrêtera automatiquement.
3. Appuyez sur la touche Suppr de votre clavier pour supprimer la sélection en cours.
4. Enregistrez votre séquence.

Pour supprimer les extrémités de notre séquence, nous aurions également pu retenir la procédure suivante :

1. Placez votre curseur au début de la séquence à conserver.
2. Maintenez la touche Majuscule (Shift).
3. Lancez la lecture de la séquence ou déplacez le curseur de la tête de lecture en le faisant glisser avec votre souris à l'endroit désiré.
4. Votre sélection est à présent effectuée, sélectionnez la commande Ne conserver que la sélection du menu Édition.

Figure 1-10

La sélection se traduit par une zone plus foncée de la jauge de lecture du contrôleur.

6. Enregistrez enfin votre document sous un nom différent en faisant très attention de cocher le bouton radio Enregistrer comme séquence autonome.

Astuce

Utilisez le raccourci clavier Ctrl+B (PC) ou Cmd+B (Mac) pour désélectionner la sélection en cours.

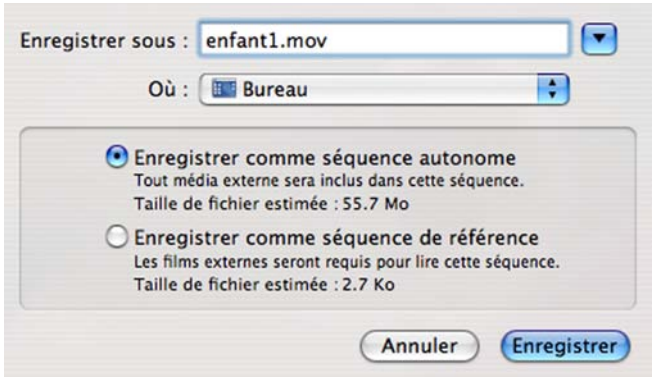


Figure 1-11

Si vous n'enregistrez pas en séquence autonome, vous devrez conserver le ou les fichiers d'origine pour que celui que vous vous apprêtez à enregistrer soit lisible.

Maintenant que nous avons appris à « nettoyer » notre séquence, voyons comment travailler sur les pistes de celle-ci.

Astuce

Agrandissez la fenêtre de votre vidéo pour effectuer des sélections plus précises.

Ajouter une image sur une vidéo

Si vous souhaitez ajouter votre logo sur une vidéo, ou n'importe quel autre type d'image, la technique est très simple car elle relève du copier-coller. Gardez à l'esprit qu'en ajoutant une image, vous ajoutez une piste.

Dans la procédure ci-dessous, nous allons partir d'une séquence ne contenant que deux pistes (une vidéo et un fichier son). Nous allons également utiliser une image au format PNG, avec un fond transparent. Le format n'a pas d'importance, QuickTime étant capable de reconnaître tous les formats graphiques standards.

Figure 1-12

L'image que vous allez placer dans la séquence ne sera plus nécessaire après le copier-coller.

1. Dans QuickTime, ouvrez l'image souhaitée.
2. Cliquez sur Ctrl+A (PC) ou Cmd+A (Mac) pour tout sélectionner, puis sur Ctrl+C ou Cmd+C pour copier cette sélection.

> **Remarque**
>
> Si vous utilisez un navigateur, vous pouvez effectuer un clic droit sur une image et sélectionner Copier dans le menu contextuel qui s'affiche. Respectez bien évidemment les droits d'auteurs de ce visuel. Dans un logiciel de traitement graphique, vous pouvez également copier la sélection en cours.

3. Ouvrez la séquence vidéo dans laquelle vous souhaitez ajouter une image.
4. Sélectionnez une partie de la séquence durant laquelle l'image sera visible. Utilisez le raccourci clavier Ctrl+A (PC) ou Cmd+A (Mac) si vous souhaitez sélectionner l'intégralité de la séquence.

> **Remarque**
>
> Pour apprendre à sélectionner une partie de la séquence, référez-vous à la procédure qui se trouve avant la figure 1-9.

5. Dans le menu Édition, sélectionnez la commande Ajouter à la sélection et mettre à l'échelle.
6. Dans le menu Fenêtre, sélectionnez la commande Propriétés de la séquence.

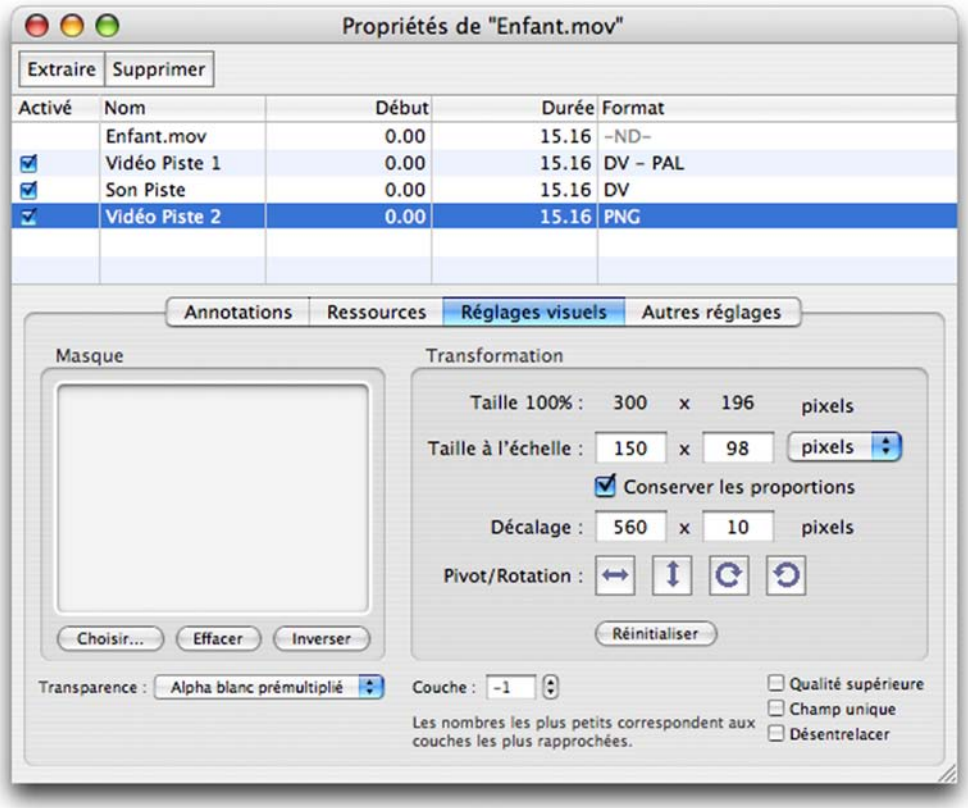


Figure 1-13

La palette Propriétés de la séquence permet non seulement d'obtenir des renseignements sur la séquence, mais également d'exécuter des opérations de montage relativement basiques.

7. Dans la palette qui apparaît, sélectionnez la piste qui s'est ajoutée, il s'agit généralement de la piste Vidéo Piste 2. Pour vérifier s'il s'agit de celle que vous voulez traiter, cliquez sur la case à cochée située à gauche de la piste sélectionnée (dans la colonne Activé). Cliquez alors sur l'onglet Réglages visuels afin d'accéder aux différentes transformations possibles :

- Taille à l'échelle, pour modifier la largeur et la hauteur de l'image ajoutée.
- Décalage, pour positionner précisément votre image sur la vidéo.
- Pivot/Rotation, pour manipuler l'image.
- Couche, pour changer le niveau de plan de la piste. N'utilisez ce paramètre que si vous devez gérer le plan entre deux images ou deux vidéos. Vous risquez de voir disparaître une image si vous la placez derrière une vidéo (valeur élevée).

Remarque

Lorsque vous saisissez une valeur de décalage, tapez sur la touche Entrée de votre clavier pour visualiser le résultat. Si ce dernier ne vous convient pas, tapez un autre nombre, sans même sélectionner de nouveau la valeur que vous venez de saisir.

En bas à gauche de cette palette, un menu déroulant dont le préfixe est Transparence, vous permet de choisir un mode d'incrustation. Choisissez en un en fonction de vos besoins.

Voilà, vous avez non seulement ajouté une image à votre séquence, mais vous avez également contrôlé sa taille et sa position sur la vidéo. Maintenant, vous souhaitez peut-être supprimer cette image, c'est-à-dire la piste sur laquelle elle se trouve.

Figure 1-14

Un logo a été ajouté à la vidéo, puis redimensionné avant d'être placé dans le coin supérieur droit.

Supprimer une piste

Il est assez rare de rencontrer une séquence avec plusieurs pistes (autres que la piste vidéo et son), mais si vous en avez ajouté une nous devons apprendre à la supprimer.

1. Ouvrez une séquence qui contient plusieurs pistes.
2. Dans le menu Fenêtre, sélectionnez la commande Propriétés de la séquence.
3. Dans la palette qui apparaît (figure 1-13), sélectionnez la piste à supprimer. Pour vérifier si vous vous apprêtez à supprimer la bonne, cliquez sur la case à cocher située à gauche de la piste sélectionnée (dans la colonne Activé).

4. Cliquez sur le bouton Suppr qui se trouve en haut de la palette.

Cette technique est très simple et vous évite donc de passer par un logiciel de montage vidéo. À présent, découvrons une autre technique de manipulation des pistes.

Extraire une piste

Si vous possédez une séquence dont la piste vidéo vous intéresse (ou la piste audio), vous avez la possibilité de créer automatiquement un nouveau document QuickTime avec la piste souhaitée. Cela vous permettra ainsi de l'utiliser pour l'intégrer dans un autre document.

1. Ouvrez la séquence vidéo à traiter.
2. Dans le menu Fenêtre, sélectionnez la commande Propriétés de la séquence.
3. Dans la palette qui apparaît (figure 1-13), sélectionnez la piste à extraire. Pour vérifier si vous vous apprêtez à extraire la bonne, cliquez sur la case à cocher située à gauche de la piste sélectionnée (dans la colonne Activé).
4. Cliquez sur le bouton Extraire qui se trouve en haut de la palette.

Vous obtenez un nouveau fichier que vous n'avez plus qu'à enregistrer en séquence autonome. Vous pouvez aussi le copier-coller dans une autre séquence.

Ajouter un texte

Nous avons souhaité consacrer une section à cette technique car il s'agit d'une opération semblable à l'ajout d'une image avec quelques manipulations supplémentaires. Commencez donc par apprendre à ajouter une image à une séquence si vous ne savez pas le faire, puis continuez la lecture au paragraphe suivant.

Au lieu de copier une image comme vous venez de le faire ou savez le faire, copiez un texte en le sélectionnant préalablement. Avant de tenter de coller le contenu de votre Presse-Papiers dans votre vidéo, vous avez sûrement tout intérêt à sélectionner uniquement une partie de votre séquence et non la totalité. À présent, comme vous l'avez fait avec l'image, activez la commande Ajouter à la sélection et mettre à l'échelle.

Après avoir copié-collé votre texte, vous observerez que la taille du texte n'est peut-être pas celle souhaitée. Vous devez donc effectuer les réglages de mise en forme de votre texte avant de le copier.

Vous noterez également que le texte vient toujours se placer sous la vidéo. Vous devez donc le replacer en modifiant le deuxième paramètre de l'option Décalage de la fenêtre Propriétés.

> **Remarque**
>
> Dans le cas où vous auriez un sous-titrage complet d'une séquence à réaliser, optez plutôt pour une technique combinant une vidéo au format FLV, un SWF et un fichier XML. Consultez le sommaire de cet ouvrage pour découvrir les différentes pages consacrées à cette technique.

Dans certains cas, vos besoins de mise en forme du texte seront tels qu'il sera nécessaire d'opter pour une autre technique. Nous abordons cette dernière à la fin de ce chapitre.

Utilisez donc le texte ci-dessous dans un fichier avant de le copier-coller à votre séquence.

```
{QTtext}
{justify:Center}{size:18}{Plain}
{textColor:65535,65535,65535}{backColor:0,0,0}
{width:768}{height:576}
{Anti_Alias:on}{Language:1}
{timeStamps:Absolute}{timeScale:600}
[00:00:00.0]









Luna la petite soeur de Marine
[00:00:03.0]
```

Vous observerez qu'il est possible de choisir précisément la position verticale du texte en ajoutant des retours à la ligne avant votre ou vos lignes de texte. Pour régler plus finement l'affichage de vos textes, référez-vous à la dernière partie de ce chapitre.

Ajouter un fichier son

La technique d'ajout d'un son à une séquence est une fois encore très simple et comparable à l'ajout d'une image. Commencez donc par apprendre cette technique si vous ne la maîtrisez pas, puis continuez la lecture au paragraphe suivant.

1. Ouvrez dans QuickTime la séquence qui contient une piste audio à copier et cliquez sur Ctrl+A (PC) ou Cmd+A (Mac).
2. Utilisez le raccourci clavier Ctrl+C (PC) ou Cmd+C (Mac) pour copier dans le Presse-Papiers le fichier son à coller.
3. Ouvrez la séquence vidéo sur laquelle vous souhaitez ajouter une sonorisation (une piste son).
4. Placez votre curseur sur l'image à partir de laquelle la piste son doit démarrer.
5. Dans le menu Édition, sélectionnez la commande Ajouter à la séquence.

Faites très attention à la durée de la piste son que vous ajoutez à votre séquence. Elle doit être inférieure ou égale à la durée de votre vidéo, sinon vous générerez un blanc à la fin de votre séquence (un écran blanc d'une durée plus ou moins longue apparaîtra le temps que la piste son se termine).

Ne choisissez surtout pas la commande Coller du menu Édition pour ajouter un son à votre séquence. Vous obtiendriez une vidéo composée de trois parties : le début de votre séquence, un écran blanc avec la musique que vous venez d'ajouter, puis la fin de votre séquence.

Ne choisissez pas non plus la commande Ajouter à la sélection et mettre à l'échelle, car vous obtiendriez les résultats suivants :

- Une accélération du son si celui-ci est plus long que la séquence dans laquelle vous le copiez.
- Une écoute au ralenti de votre son s'il est plus court que la séquence dans laquelle vous le copiez.

La technique d'ajout d'une piste son est donc très simple, mais cela sous-entend que vous avez pris connaissance des durées de vos séquences (audio et vidéo, séquence dans laquelle vous allez ajouter le son).

Ajouter une vidéo par-dessus une autre

Avant d'apprendre à ajouter une vidéo dans une autre, vous devez être conscient que le résultat peut être différent de celui auquel vous vous attendiez. Avez-vous pensé à tout ? Lors la visualisation du passage, durant lequel deux vidéos vont se jouer en même temps, quelle piste audio doit être retenue ? Souhaitez-vous que les deux soient audibles en même temps ? Est-ce que cela ne va pas donner un brouhaha inaudible ?

Si vous pensiez couper le son de la séquence principale (celle qui va recevoir le copier-coller) durant la lecture de la vidéo que vous allez y insérer, cela est tout simplement impossible. Si vous souhaitez uniquement ajouter une piste vidéo, pensez à supprimer la piste audio de la séquence que vous allez utiliser pour l'insert. Dans notre exemple, nous allons donc partir d'une séquence qui ne contient pas de piste audio et que nous allons placer dans une séquence qui contient deux pistes (vidéo et audio).

1. Ouvrez la séquence qui ne contient pas de piste son.
2. Cliquez sur Ctrl+A (PC) ou Cmd+A (Mac), puis un Ctrl+C ou Cmd+C.
3. Ouvrez la séquence vidéo dans laquelle vous souhaitez effectuer votre insert.
4. Sélectionnez l'image à partir de laquelle votre insert doit se faire.
5. Dans le menu Édition, sélectionnez la commande Ajouter à la séquence.
6. Dans le menu Fenêtre, sélectionnez la commande Propriétés de la séquence (figure 1.15).
7. Ajustez la position et les dimensions de la piste vidéo que vous venez de coller (figure 1.16).

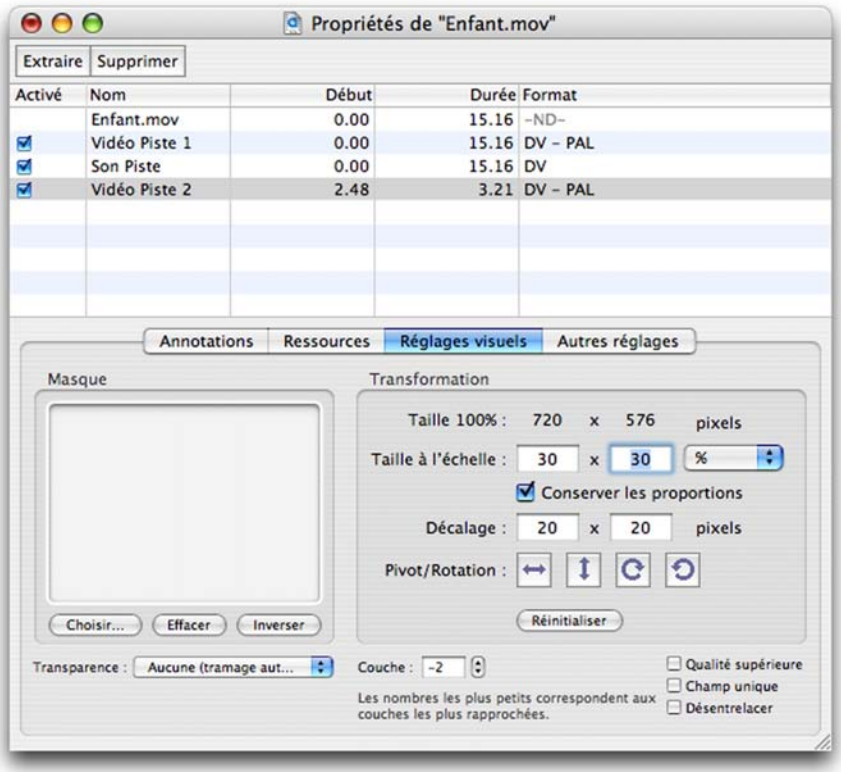


Figure 1-15

La piste Vidéo Piste 2 doit être plus petite que la séquence principale.

Figure 1-16

Dans cet exemple, nous avons incrusté la vidéo sur elle-même, juste pour un instant.

Faire pivoter une vidéo

Aujourd'hui, de nombreux appareils, notamment les appareils photo, permettent de capturer une séquence vidéo. Ces derniers ne sont pas toujours tenus dans le bon sens lors de la prise de vue. Résultat, à la lecture de la séquence sur l'ordinateur, la vidéo est inclinée à 90 ou –90°. Une fois encore, vous n'avez pas besoin d'un logiciel de montage vidéo pour effectuer cette opération. Vous pouvez simplement utiliser QuickTime.

1. Ouvrez la séquence à traiter.
2. Dans le menu Fenêtre, sélectionnez la commande Propriétés de la séquence.
3. Sélectionnez la piste à manipuler.
4. Via l'onglet Réglages visuels, cliquez sur l'un des boutons de rotation.

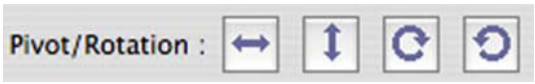


Figure 1-17

Cliquez sur l'un des deux boutons de droite pour faire tourner votre vidéo dans le sens horaire ou anti-horaire.

Monter plusieurs séquences les unes après les autres

Vous pouvez assembler plusieurs séquences pour n'en faire qu'une. La technique qui consiste à placer des vidéos les unes derrière les autres est extrêmement simple car elle se résume à un copier-coller. Partons de l'exemple suivant :

1. Ouvrez trois fichiers dans QuickTime.
2. Créez un nouveau document (Ctrl+N sur PC ou Cmd+N sur Mac).
3. Sélectionnez une des trois vidéos.
4. Cliquez sur Ctrl+A (PC) ou Cmd+A (Mac), puis un Ctrl+C (PC) ou Cmd+C (Mac). Vous venez de placer dans le Presse-Papiers de votre ordinateur le contenu d'une des trois séquences.
5. Sélectionnez la fenêtre du nouveau document.
6. Utilisez le raccourci clavier Ctrl+V (PC) ou Cmd+V (Mac) pour y placer la première séquence. Votre curseur, qui indique la position de la tête de lecture, devrait normalement se trouver à la fin. Si ce n'est pas le cas, faites-le glisser jusqu'au bout.

Figure 1-18

Avant d'effectuer un deuxième collage, votre curseur doit se trouver au bout de la jauge du contrôleur. Lorsque le collage est effectué, il est représenté par une zone grisée dans la jauge.

Remarque

Vous aurez souvent besoin d'annuler la sélection en cours ; utilisez alors le raccourci clavier Ctrl+B (PC) ou Cmd+B (Mac).

7. Exécuter un deuxième collage, la jauge de votre contrôleur doit ressembler à la deuxième copie d'écran de la figure ci-avant. Votre tête de lecture se trouve au bout de la jauge, vous pouvez effectuer un troisième collage et ainsi de suite.

Comme vous venez de le découvrir, l'ajout de séquences bout à bout se fait par le biais de simples copier-coller. Si vous souhaitez combiner des images et des pistes sons en incrustation, référez-vous aux points développés dans les pages précédentes.

Vous aurez donc compris, à l'aide des procédures élémentaires que nous venons de décrire, comment réaliser un montage simplifié Il va de soi que si vous êtes amené à réaliser de nombreux montages, le player QuickTime ne pourra pas répondre à tous les besoins dans ce domaine. De plus, il est indispensable d'apprendre les techniques de montage, ce qui ne peut se résumer à une suite de copier-coller. Consultez la page suivante pour en savoir un peu plus : http://www.zeropoint.fr/metier-monteur.htm.

Utiliser un fichier texte pour ajouter un texte à une séquence

« Copier-coller » et « sélections de commandes dans des menus », voilà comment nous pourrions résumer jusqu'à présent les points que nous venons d'aborder.

Pour ce qui suit, nous allons avoir besoin d'utiliser un éditeur de texte car nous allons créer des fichiers textes. Commencez donc tout d'abord par lancer votre programme favori (NotePad sous Windows et TextEdit sous Mac suffisent) et recopiez la ligne ci-dessous.

```
{QTtext}
[00:00:00.0]Réalisation : David TARDIVEAU [00:00:02.000]
```

Enregistrez, puis ouvrez ce fichier dans QuickTime. Vous serez surpris d'obtenir une séquence qui affiche le texte compris entre les dates qui l'encadrent dans le fichier. À quoi correspond cette syntaxe ? Pour répondre à cette question, revenons en détail sur la construction d'un tel type de fichier.

Attention

Vous devez impérativement enregistrer votre fichier au format texte et non en RTF ou tout autre format.

Dès que vous aurez besoin de réaliser un sous-titrage, un encart de texte, un écran de couleur avec ou sans texte, un générique défilant verticalement, un bandeau défilant horizontalement, vous utiliserez la technique qui consiste à décrire des séquences temporisées au travers d'un fichier texte.

Vous devez toujours commencer la rédaction de votre description par la balise `{QTtext}`. À l'ouverture du fichier par QuickTime, ce dernier sait qu'il doit interpréter les lignes qui suivent cette première information.

À partir de ce moment-là, vous pouvez définir un certain nombre de paramètres que devra utiliser QuickTime pour l'affichage des textes et autres fonds de couleurs. Dans l'exemple ci-dessus, nous avons utilisé une seule ligne pour rédiger notre instruction, mais nous aurions pu saisir le texte sous la forme suivante :

```
{QTtext}
[00:00:00.0
Réalisation : David TARDIVEAU
[00:00:02.000]
```

Si vous devez ajouter d'autres textes à des temps différents, voici un deuxième exemple :

```
{QTtext}
[00:00:00.000]
Réalisation : David TARDIVEAU
[00:00:02.000]
[00:00:02.100]
Copyright (c) 2007
[00:00:03.100]
[00:00:03.150]
Fin
[00:00:04.300]
```

Vous pouvez aussi utiliser la syntaxe suivante :

```
{QTtext}
[00:00:00.100]Réalisation : David TARDIVEAU [00:00:02.000]
[00:00:02.100]Copyright (c) 2007 [00:00:03.100]
[00:00:03.150]Fin [00:00:04.300]
```

Attention, dans le cas où vous opteriez pour la méthode qui propose l'insertion des temps avant et après votre ligne de texte, vous devrez toujours vous assurer qu'une espace est comprise entre la fin de votre message et le temps de fin d'affichage. Ainsi, la ligne ci-dessous provoquera une erreur car le « n » du mot « fin » se trouve directement accolé au crochet.

```
[00:00:03.150]Fin[00:00:04.300]
```

La temporisation

Si vous êtes observateur, vous aurez fait la relation entre les valeurs affichées entre crochets et la durée d'affichage des textes. En effet, les nombres séparés par deux points correspondent aux informations suivantes :

heure:minute:seconde:intervalle de temps.

Pourquoi la dernière valeur n'est-elle pas un nombre compris entre 0 et 25, c'est-à-dire le nombre maximal d'images contenues dans une seconde de vidéo ? En fait, vous pouvez choisir n'importe quelle échelle qui va désigner l'unité de division comprise en une seconde. Par convention, 600 est une valeur habituelle car 24, 25 et 30 en sont des multiples. Si vous êtes sûr de ne travailler qu'avec des vidéos de 25 images par seconde, vous pouvez alors choisir un timecode comme celui-ci :

```
[00:02:17.24]
```

Ce temps correspond à 2 minutes, 17 secondes et 24 images. Les deux temps suivants seraient :

```
[00:02:17.25]
[00:02:18.00]
```

Mise en forme du texte

Vous avez la possibilité d'insérer dans votre fichier les attributs ci-dessous à n'importe quel moment pour changer l'apparence du texte.

Balises (attributs) élémentaires

- `{font: police }` Police de caractères
- `{size: taille }` Taille du caractère (exprimée en points)
- `{textColor: Rouge , Vert, Bleu }` Couleur du texte
- `{anti-alias: on/off }` Lissage du texte

Pour le choix des couleurs, utilisez des valeurs comprises entre 0 et 65535, et référez-vous au tableau ci-après pour connaître les plus courantes.

Tableau 1-3 Couleurs en mode RVB

Couleurs	Valeur des couleurs
Rouge	65535, 0, 0
Vert	0, 65535, 0
Bleu	0, 0, 65535
Cyan	0, 65535, 65535
Magenta	65535, 0, 65535
Jaune	65535, 65535, 0
Noir	0, 0, 0
Blanc	65535, 65535, 65535
Gris clair	49000, 49000, 49000
Gris	32000,32000,32000
Gris foncé	16000,16000,16000

Styles de texte

- `{plain}` Texte standard, normal
- `{bold}` Texte en gras
- `{italic}` Texte en italique *

* Pour un affichage de texte sur une vidéo, évitez d'utiliser ces styles, car cela diminue la lisibilité du texte, sauf si vous utilisez une taille assez élevée. N'abusez pas des styles de texte et réservez l'italique pour les dialogues ou citations.

- `{underline}` Texte souligné
- `{outline}` Texte détouré *
- `{shadow}` Texte ombré *
- `{condense}` Caractères rapprochés
- `{extend}` Caractères espacés

* Pour un affichage de texte sur une vidéo, évitez d'utiliser ces styles, car cela diminue la lisibilité du texte, sauf si vous utilisez une taille assez élevée. N'abusez pas des styles de texte et réservez l'italique pour les dialogues ou citations.

Alignement du texte à l'écran

- `{justify:left}` Texte aligné à gauche
- `{justify:right}` Texte aligné à droite
- `{justify:center}` Texte centré

Cadre de fond du texte

- `{backColor: rouge , vert , bleu }` Couleur du cadre de fond de texte
- `{height: hauteur }` Hauteur du cadre de fond de texte
- `{width: largeur }` Largeur du cadre de fond de texte
- `{textBox: haut , gauche , bas , droite }` Position et dimensions de la piste texte
- `{keyedText: on ou off }` Transparence du fond

> **Remarque**
> Utilisez la valeur 0 avec la propriété `width` pour une taille verticale automatique.

Balises liées à la séquence

Tableau 1-4 Balises de langue et d'échelle de temps

Balises	Description
`{language: nombre }`	Indique la langue du texte. 0 pour l'anglais, 1 pour le français et 11 pour le japonais.
`{timeStamps: valeur }`	`Absolute` ou `relative`.
`{timeScale: nombre }`	Échelle temporelle contenue dans une seconde de vidéo.

Exemples

`{timestamps:absolute}` Les marqueurs font référence au début de la séquence. Ainsi, 00:04:10.000 indique 4 minutes et 10 secondes à partir du début de la vidéo.

`{timeStamps:relative}` Les marqueurs font référence à la marque précédente. Ainsi, 00:04:10.000 indique 4 minutes et 10 secondes après le marqueur précédent.

Les trois derniers zéros indiquent l'échelle temporelle pour une seconde de vidéo. Généralement, on utilise la valeur 600. Cette dernière présente l'avantage d'être à la fois un multiple de 24, 25 et 30, qui représentent respectivement le nombre d'images par seconde d'une séquence de cinéma, de vidéo en PAL et NTSC.

Ainsi avec 600, le repère 00:00:01.300 indique 1,5 seconde.

Balises liées au texte

Tableau 1-5 Balises relatives à l'affichage du texte

Balises	Valeurs	Descriptions
`{doNotDisplay:on}`	`on` ou `off`	Permet d'activer (`on`) ou de désactiver l'affichage du texte de la piste.
`{doNotAutoScale:on}`	`on` ou `off`	Permet d'activer ou de désactiver la mise à l'échelle automatique lorsque l'utilisateur agrandit ou rétrécit la taille de la fenêtre où se lit la séquence.
`{clipToTextBox:on }`	`on` ou `off`	Permet de réduire la zone de fond (si elle est opaque) aux dimensions minimales du texte qui est affiché.
`{useMovieBackColor:on }`	`on` ou `off`	Permet de préciser que la piste texte doit (ou ne doit pas) utiliser le fond de la séquence (ou son propre fond).
`{shrinkTextBox:on }`	`on` ou `off`	Permet de réduire la zone d'affichage au strict minimum permettant ainsi d'afficher un texte plus grand.

Balises liées au défilement du texte

Tableau 1-6 Balises relatives au défilement du texte

Balises	Valeurs	Descriptions
`{scrollDelay:600}`	nombre	Permet de régler la durée de pause avant le lancement du defilement du texte. Il s'agit d'une valeur exprimée en subdivision de la seconde. Par exemple, 600 pour 1 seconde si la timeScale a été définie avec cette valeur.
`{scrollIn:on }`	`on` ou `off`	Permet d'activer le mode Défilement du texte. Le texte part de l'extérieur de écran (`on`).
`{scrollOut:on }`	`on` ou `off`	Permet de spécifier si le texte doit quitter intégralement l'écran.
`{horizontalScroll:on }`	`on` ou `off`	Permet d'activer le défilement horizontal du texte.
`{reverseScroll:on }`	`on` ou `off`	Permet d'inverser le sens de defilement du texte.
`{continuousScroll:on}`	`on` ou `off`	Permet d'enchaîner les défilements de textes sans être oblige d'attendre.

Astuce

Utilisez le texte ci-dessous pour insérer un fond de couleurs (ou noir et blanc) entre deux images. Vous pouvez bien sûr définir une durée d'affichage.

```
{QTtext}
{justify:Center}{size:18}{Plain}
{textColor:0,0,0}{backColor:0,0,65535}
{width:768}{height:576}
{Anti_Alias:on}{Language:1}
{timeStamps:Absolute}{timeScale:600}
[00:00:00.0]
[00:00:00.100]
```

Les contraintes d'encodage

Vous devez garder à l'esprit que deux fichiers présentant deux séquences différentes, mais de même durée, ne feront jamais le même poids car la compression est calculée différemment selon les mouvements internes de la vidéo et la composition de chaque image.

La compression d'une vidéo se fait sur une suite d'images. Plus une image possède d'aplats de couleurs et/ou peu de changements d'une image à l'autre, plus la compression donnera un poids faible.

Ainsi, une vidéo présentant une interview sera toujours mieux compressée qu'une bande annonce de film qui contient de nombreux plans, tous aussi différents les uns des autres.

En résumé, nous pouvons dire que la compression d'une vidéo sera meilleure si :

- La séquence contient des mouvements de caméra lents et/ou peu de plans.
- La séquence contient plutôt des plans fixes.
- La séquence présente une bonne lumière.

Malheureusement, dans la plupart des cas, nous ne pouvons pas assurer le tournage et les séquences nous sont généralement fournies.

À partir de cet instant, nous devons nous poser deux questions avant de passer à l'encodage.

- Quelles sont les dimensions (largeur × hauteur) de la séquence ?
- Quelle est sa cadence en ips (images/seconde) ?

Ce sont deux informations capitales que nous devons récupérer car la compression va se produire en respectant ces données initiales. Vous devez toujours choisir des rapports de calcul tels que nous vous les présentons dans le tableau ci-après.

Partons de l'exemple suivant : vous récupérez une séquence vidéo de 25 ips en 720 × 576, elle dure 2 minutes et 12 secondes. Voici les options que vous avez le droit d'utiliser.

Tableau 1-7 Paramètres à utiliser pour une compression

	Données initiales	Rapport 1/2	Rapport 1/3	Rapport 1/4
Largeur	720 (768)	384	256	192
Hauteur	576	288	192	144
Cadence	25 ips	12 ou 12,5	8 (à éviter)	Ne pas utiliser

Remarque

Si vous utilisez le codec Sorenson pour le player Flash 6 ou 7, vous pourrez opter pour 12,5 images/seconde. En revanche, avec le codec On2 VP6, vous devrez retenir un débit de 12 ips lorsque vous choisirez de diviser par deux les 25 ips initiales.

Apportons tout de même quelques commentaires à ce tableau. Nous pouvons dire d'une façon générale que vous devez toujours essayer de conserver des valeurs proches de celles d'origine. Malheureusement, diffuser une vidéo en 768 pixels de largeur par 576 pixels de hauteur à une vitesse de 25 images par seconde, semble difficile aujourd'hui par Internet. En revanche, si vous projetez de réaliser un DVD-Rom ou une application qui utilise des données enregistrées sur disque dur, dans ce cas, cela ne posera pas de problèmes.

Le tableau vous présente des valeurs qui peuvent être combinées ; ainsi, vous pouvez tout à fait encoder une vidéo en 384×288 à 25 ou 24 ips. En revanche, l'inverse (768×576 à 12 ips) présente moins d'intérêt.

Comme nous l'évoquions au début de ce chapitre, la vidéo HD va peu à peu s'imposer comme standard, remplaçant ainsi le format DV. Il faudra cependant attendre encore de nombreuses années avant que cela se produise. Quoi qu'il en soit, le problème que nous connaissons aujourd'hui restera le même. Il ne sera pas toujours possible d'afficher une image en plein écran. Nous aurons alors recours à différents calculs pour connaître les dimensions d'une séquence à diffuser.

Encoder une séquence vidéo au format FLV

Nous allons à présent procéder à l'encodage, c'est-à-dire la dernière étape avant de passer dans Flash. Dans les développements précédents, nous avons préparé notre vidéo, nous avons pris connaissance des informations qui la caractérisent et dont nous allons avoir besoin, nous pouvons maintenant l'encoder en utilisant l'une des trois techniques ci-dessous :

- Importer une vidéo dans Flash et suivre la procédure proposée par l'assistant.
- Utiliser l'application Flash Video Encoder.
- Utiliser un autre logiciel dédié.

Importer une vidéo dans Flash

Si vous n'avez besoin d'encoder qu'une seule vidéo et que vous ne maîtrisez pas encore la procédure d'intégration d'une vidéo dans une animation Flash, cette solution est pour vous.

Cette technique va non seulement vous proposer une assistance pour l'encodage de votre vidéo, mais elle gère également le placement du composant sur la scène et son apparence (également appelée *skin*).

1. Commencez tout d'abord par enregistrer votre animation si vous ne l'avez pas encore fait.
2. Sélectionnez la commande Importer du menu Fichier puis la sous-commande Importer de la vidéo...

Figure 1-19

Pour l'incrustation simplifiée d'une vidéo dans une animation, un assistant vous propose de répondre à ses questions.

3. En cliquant sur le bouton Parcourir, sélectionnez le fichier vidéo que vous souhaitez utiliser. Si vous indiquez une vidéo déjà encodée au format FLV, passez les étapes 4 à 6, sinon, continuez la lecture de cette procédure pas à pas.

Comme vous le montre la copie d'écran ci-dessous, vous pouvez aussi indiquez une URL où se trouve déjà un fichier encodé. À ce stade de vos connaissances, nous ne détaillerons pas la possibilité de pointer vers un fichier XML.

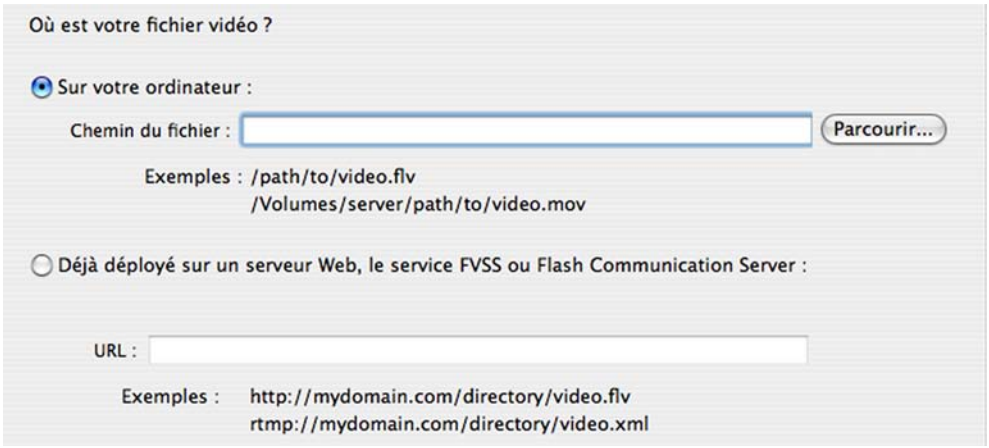


Figure 1-20

Nous pouvons choisir un fichier qui est déjà encodé ou qui ne l'est pas encore.

4. Si vous avez acheté ce livre, c'est que vous aurez besoin d'utiliser la première ou la troisième solution de la copie d'écran ci-dessous. Si vous devez utiliser un serveur Flash Media Server, consultez le chapitre 6 de ce livre. D'une façon générale, vous sélectionnerez toujours la première option qui est d'ailleurs la solution proposée par défaut.

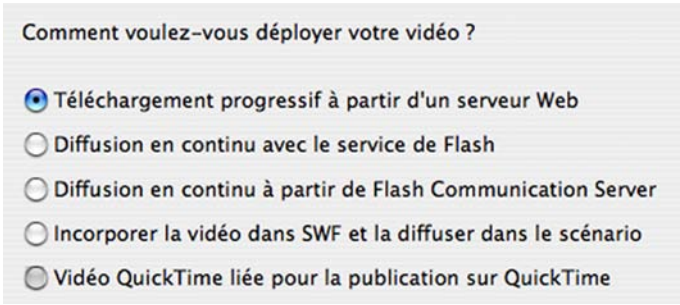


Figure 1-21

Le choix d'un téléchargement progressif ne nécessite pas l'usage d'un serveur de flux. Cette option vous permet donc de proposer à l'internaute la consultation de vidéos sur une page Web sans aucune difficulté.

5. Pour l'étape qui va suivre, vous avez le choix entre deux possibilités :

   Solution A : vous sélectionnez un profil de codage dans le menu déroulant situé en haut à gauche de la fenêtre (en optant pour Flash 8 - Qualité moyenne (400 Kbps)) et vous indiquez un nom de fichier qui sera celui du fichier FLV que vous allez obtenir.

   Solution B : vous cliquez sur le bouton Afficher les paramètres avancés pour des réglages plus précis.

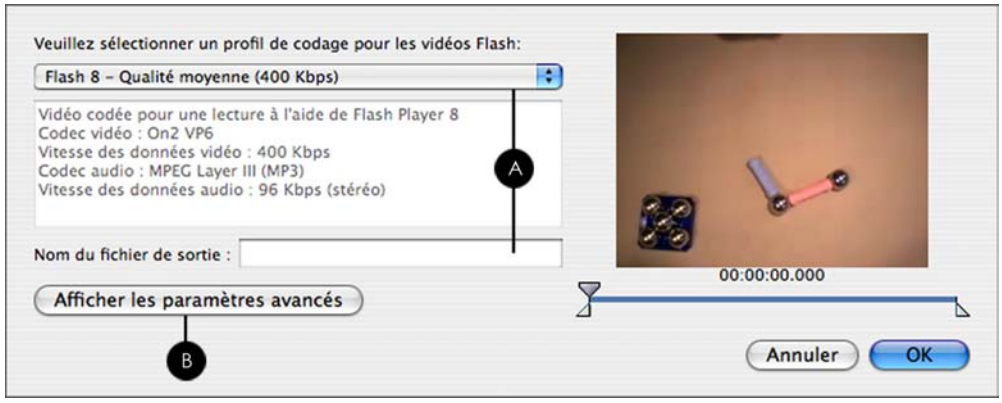


Figure 1-22

Nous vous conseillons fortement d'opter pour la solution B pour encoder vos vidéos.

Quel que soit votre choix, détaillons la deuxième solution.

6. La copie d'écran de la figure 1-23 vous présente une série de réglages à effectuer avant de lancer votre encodage.

   Choisissez entre le codec On2 VP6 ou Sorenson Spark. Gardez à l'esprit que la première vous permet d'obtenir une meilleure qualité de vidéo (rond noir n° 1).

   Définissez une qualité Moyenne ou Personnalisée via le menu déroulant situé sous le rond noir n° 2. Attention, le débit est exprimé en kilo-bits et non en kilo-octets (kilo-bytes) comme le montre la figure 1-23. Si vous choisissez une valeur inférieure à 250, la qualité de votre vidéo ne sera pas très bonne. Au-delà de 400, vous obtiendrez une bonne qualité, mais le poids du fichier vidéo sera important. Nous vous conseillons donc de faire des tests compris entre 250 Kbits/s et 450 Kbits/s. Il y a de fortes chances pour que votre choix s'arrête sur une valeur de 350 à plus ou moins 20.

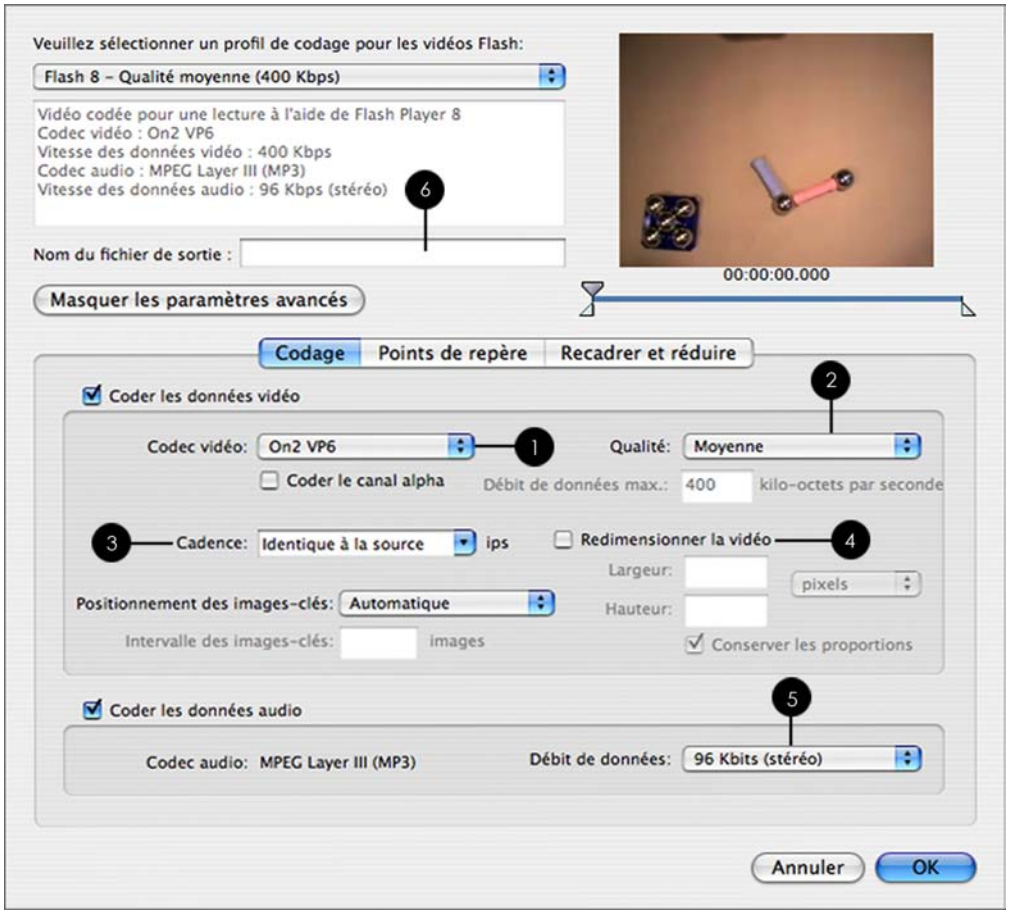


Figure 1-23

Tous ces réglages vous permettront d'optimiser le rapport qui existe entre la qualité de compression de vos vidéos et le poids obtenu du fichier.

Remarque

Pour connaître le nombre de kilo-octets lorsque vous définissez un nombre en kilo-bits, divisez la valeur retenue par 8. Ainsi 320 kilo-bits équivalent à 40 Ko (320/8). En protocole HTTP, un tel débit représente déjà une bonne moyenne.

Pour les réglages relatifs à la cadence de la séquence et ses dimensions (ronds noirs n° 3 et n° 4), nous venons de voir dans la section précédente, Les contraintes d'encodage, les calculs que vous devez faire par rapport à vos fichiers sources.

Au niveau du son (rond noir n° 5), nous vous conseillons de conserver la valeur 96 par défaut. Au-delà de cette valeur, vous augmenterez le poids de vos fichiers sans pour autant obtenir une meilleure qualité audio perceptible. En dessous de cette valeur, la qualité du son se dégrade très vite.

Terminez cette étape en précisant le nom du fichier .flv que vous allez obtenir (rond noir n° 6).

7. Sélectionnez un type de contrôleur qui accompagnera votre séquence vidéo sur la scène de votre animation Flash. Le tableau ci-après vous présente les différentes possibilités de configuration.

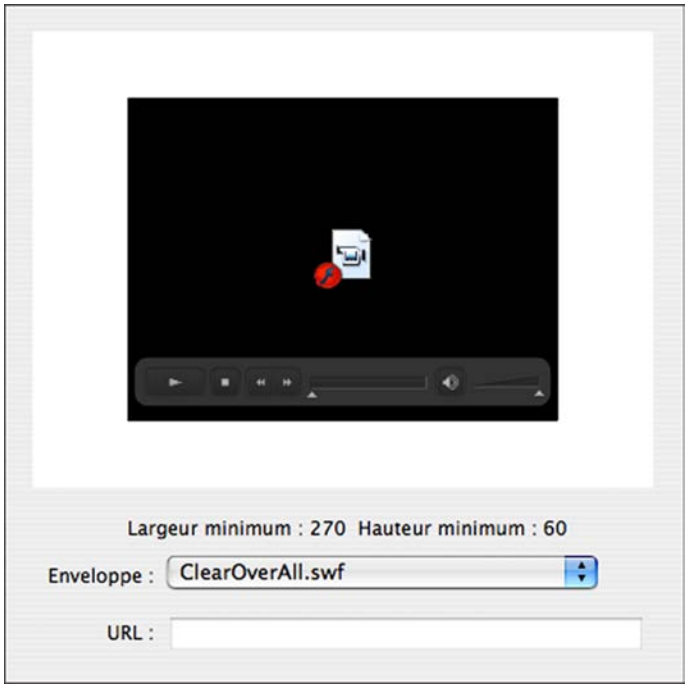


Figure 1-24

Le menu déroulant vous propose quatre séries de huit contrôleurs de quatre couleurs différentes.

Tableau 1-8 Le contrôleur d'une séquence peut se placer sur l'image ou sous elle.

Contrôleurs sous la séquence	Contrôleur sur la séquence
 Largeur minimum : 270 Pas de hauteur minimum Enveloppe : SteelExternalAll.swf	 Largeur minimum : 270 Hauteur minimum : 60 Enveloppe : SteelOverAll.swf
 Largeur minimum : 230 Pas de hauteur minimum Enveloppe : SteelExternalNoVol.swf	 Largeur minimum : 230 Hauteur minimum : 60 Enveloppe : SteelOverNoVol.swf
 Largeur minimum : 96 Pas de hauteur minimum Enveloppe : SteelExternalPlayMute.swf	 Largeur minimum : 96 Hauteur minimum : 60 Enveloppe : SteelOverPlayMute.swf
 Largeur minimum : 155 Pas de hauteur minimum Enveloppe : SteelExternalPlaySeekMute.swf	 Largeur minimum : 155 Hauteur minimum : 60 Enveloppe : SteelOverPlaySeekMute.swf

La série que nous vous présentons ci-dessus est la Steel (couleur acier) ; il en existe trois autres dont les noms sont Artic (bleu clair), Clear (blanc) et Mojave (kaki clair).

Au terme de ces sept étapes, Flash procède à l'encodage de la vidéo que vous lui avez indiquée, ce qui peut prendre plus ou moins de temps. Durant ce laps de temps, vous ne pouvez pas travailler sur Flash, ce qui implique que si vous avez de nombreuses vidéos à encoder, vous avez sûrement intérêt à utiliser une autre technique (par exemple l'une des deux qui vont suivre). Lorsque l'encodage est terminé, Flash place automatiquement un composant FLVPlayer sur la scène et le configure de façon standard. Il est probable que tous les réglages ne vous conviennent pas : passons-les en revue.

Configurer la vidéo

1. Sélectionnez l'occurrence de votre vidéo (composant FLV Playback) sur la scène (un simple clic sur l'occurrence suffit).
2. Cliquez sur l'onglet Paramètres qui se trouve au même endroit que la palette Propriétés. Six réglages apparaissent alors, ainsi que six autres auxquels vous pouvez accéder en utilisant l'ascenseur.

   – `autoplay` : réglé sur `true`, la lecture de la vidéo démarrera au chargement du .swf. L'utilisateur n'aura pas à cliquer sur le bouton Lecture. Dans certains cas, il sera plus judicieux de régler ce paramètre sur `false`, laissant ainsi l'utilisateur choisir l'instant auquel il souhaite démarrer la lecture de la vidéo.

   – `autoRewind` : réglé sur `true`, le rembobinage de la vidéo se fera automatiquement. Lorsque la lecture d'une séquence est terminée, la première image de la vidéo s'affiche à nouveau dans l'occurrence du composant FLV Playback présente sur la scène. Cette option réglée à `true` donne l'impression d'un sursaut d'image à la fin de la séquence.

   – `autoSize` : laissez ce paramètre réglé sur `true` si vous êtes amené un jour à changer vous-même le paramètre contentPath.

   – `bufferTime` : ce paramètre, dont la valeur est comprise entre 0 et 1, permet de spécifier la durée de la mémoire tampon. Lorsque vous arrivez sur une page Web qui contient une animation Flash comprenant une vidéo, cette dernière ne peut démarrer sa lecture directement. Il faut attendre un temps minimum de chargement. Si la durée de votre séquence est de 1 minute et que vous réglez le paramètre `bufferTime` à 0,1 (soit 10 %), il faudra attendre un minimum de 6 secondes avant le début de la lecture de la vidéo. Si vous saisissez la valeur 0,5 (50 %) dans ce champ, , il faudra attendre 30 secondes avant le lancement de la vidéo. Pour une même valeur de `bufferTime`, le temps d'attente varie donc en fonction de la durée de la séquence. La valeur idéale doit être comprise entre 0,1 et 0,4 au grand maximum.

   – `contentPath` : évitez d'utiliser ce paramètre, il est source de nombreux problèmes. Saisissez de préférence la ligne d'instruction ci-dessous sur une image clé de la timeline, celle qui contient de préférence l'occurrence du composant FLV Playback.

   ```
   ecranVideo.contentPath = "nomDeMaVideo.flv";
   ```

Note

`ecranVideo` est le nom que vous avez donné à l'occurrence du composant FLV Playback. Par exemple, la copie d'écran de la figure 1-25 ne contient pas encore de nom d'occurrence, car le champ est vide.

   – `skin` : en effectuant un clic sur ce paramètre, vous reconnaîtrez l'interface que nous vous avons présentée ci-avant pour sélectionner un contrôleur.

   – `skinAutoHide` : ce paramètre sera très pratique si vous avez choisi un contrôleur sur la vidéo. Cette option permet en effet de n'afficher le contrôleur qu'à partir du moment où le curseur de la souris se trouve sur la vidéo.

– `volume` : utilisez une valeur comprise entre 0 est 100 pour régler le niveau initial de la piste audio de votre séquence.

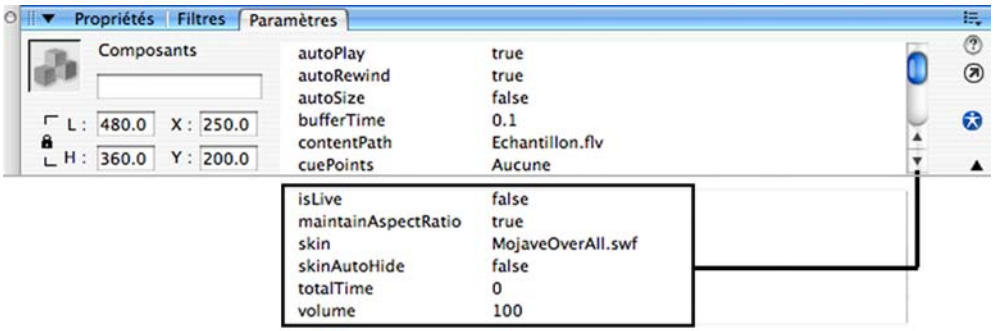


Figure 1-25

Les réglages accessibles via la palette Paramètres vous permettent d'éviter de passer par l'ActionScript pour configurer votre vidéo sur la scène.

La technique d'insertion d'une vidéo dans une animation Flash que nous venons de voir est simple, abordable pour les néophytes qui ne connaissent pas obligatoirement Flash, mais elle présente de nombreuses limites. Nous vous invitons donc à lire les paragraphes suivants afin de découvrir qu'il existe d'autres solutions plus complexes, mais tout de même abordables.

Utiliser l'application Flash Video Encoder

Si vous avez lu les pages précédentes, vous avez sûrement retenu qu'il est impossible d'encoder une vidéo tout en continuant de travailler dans l'interface de Flash. Vous avez en effet sollicité Flash pour vous aider à intégrer une vidéo sur la scène, mais il a lui-même sollicité l'application Flash Video Encoder pour faire le travail d'encodage. Flash est donc en attente jusqu'à la fin de cette demande.

Pour pouvoir encoder des vidéos tout en continuant de travailler dans Flash, vous devez donc sous-traiter cette tâche : c'est le rôle de l'application Flash Video Encoder.

Figure 1-26

L'application Flash Video Encoder permet d'encoder une vidéo pendant que vous travaillez dans Flash.

Commencez donc par regrouper toutes les vidéos que vous souhaitez encoder dans un même dossier, puis lancer l'application Flash Video Encoder. La fenêtre de la copie d'écran ci-après apparaît. Faites glisser un ou plusieurs (ou l'ensemble) de vos fichiers vidéo dans la fenêtre. Vous pouvez aussi cliquer sur le bouton Ajouter pour sélectionner votre ou vos séquences via une fenêtre système.

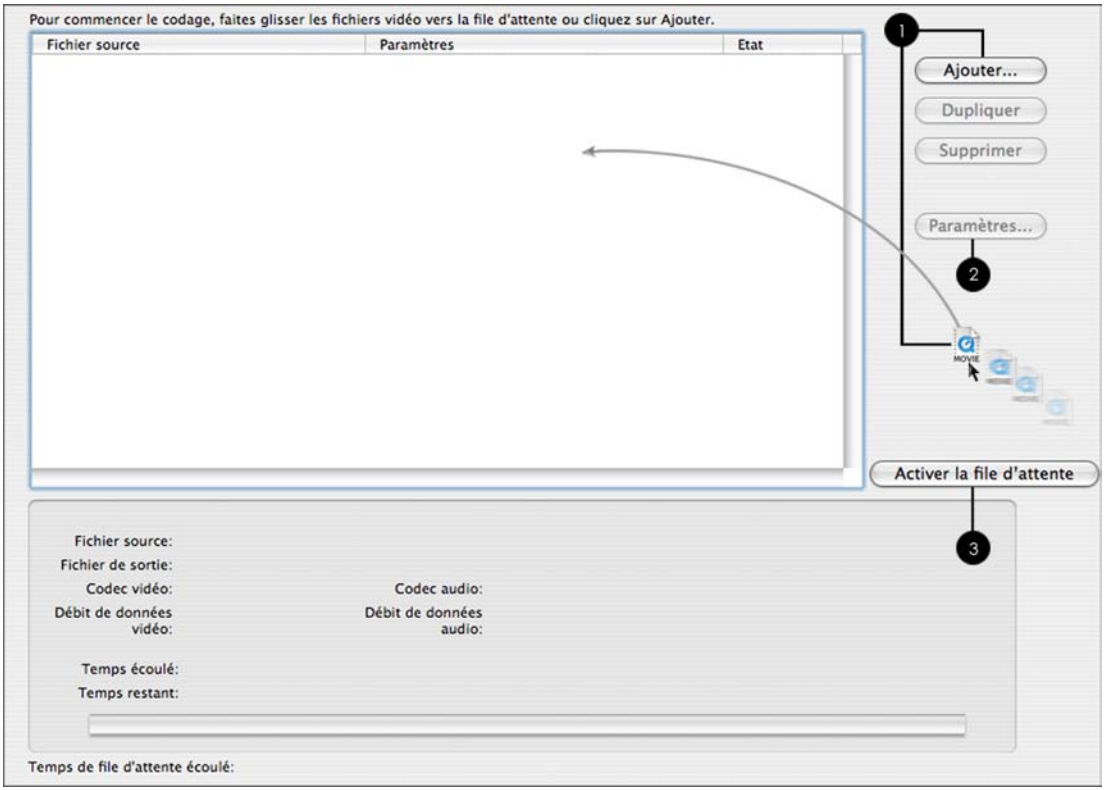


Figure 1-27

Par un simple glisser-déplacer d'un fichier dans la fenêtre, vous pouvez encoder une vidéo.

Comme vous le montre la figure 1-28, des paramètres d'encodage par défaut vous sont proposés ; vous pouvez donc si vous le souhaitez cliquer sur le bouton Activer la file d'attente pour lancer l'étape d'encodage. Si vous souhaitez modifier les dimensions de votre séquence, ou sa cadence ou encore d'autres paramètres, vous n'aurez pas le choix et serez obligé de cliquer sur le bouton Paramètres.

Vous allez alors vous retrouver face à une fenêtre qui correspond à celle que nous vous avons présentée à la figure 1-22. Consultez cette copie d'écran ainsi que la figure 1-23 et les explications qui les accompagnent. Lorsque vous atteindrez la septième étape de cette procédure, n'allez pas plus loin et revenez à cette page.

En cliquant sur le bouton OK de la fenêtre des paramètres d'encodage (figure 1-23), vous vous retrouvez face à la fenêtre d'origine (figure 1-28). Bien sûr votre fenêtre possède peut-être une liste plus longue de vidéos à modifier. Cliquez à présent sur le bouton Activer la file d'attente pour lancer l'étape d'encodage qui peut prendre plusieurs heures ou juste quelques minutes. Tout dépend du nombre de fichiers que vous devez compresser et des paramètres que vous avez spécifiés.

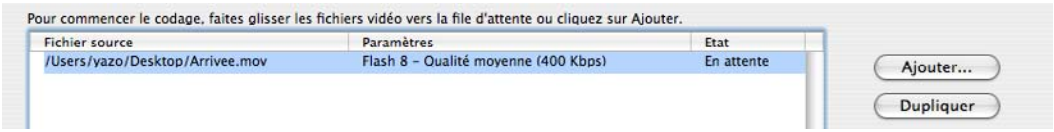


Figure 1-28

Paramètres d'encodage proposés par défaut après un glisser-déplacer

Modifier les paramètres d'encodage de plusieurs fichiers

Si vous devez encoder plusieurs fichiers avec les mêmes paramètres, nous vous conseillons de sélectionner plusieurs lignes de fichiers avant de cliquer sur le bouton Paramètres…

Utiliser un autre logiciel dédié

Flash Video Encoder est une très bonne application, mais elle ne propose pas toutes les options disponibles dans certains logiciels dédiés à l'encodage au format FLV. Cependant, certains éditeurs de logiciels ont bien pris conscience de l'importance de ce format, et ils ont donc créé des programmes tels que Sorenson Squeeze, On2 Flix Pro, Riva FLV Encoder et quelques autres moins connus, et surtout moins efficaces.

> **Remarque**
>
> À ce jour, Sorenson Squeeze et On2 Flix Pro sont les seuls logiciels à fonctionner sur les deux plates-formes OS X et Windows. Riva ne fonctionne pas encore sur Mac. Par ailleurs, seules les versions 4.3 et postérieures de Sorenson Squeeze Suite proposent le codec On2 VP6.

Il est vrai que ces logiciels sont payants alors que Flash Video Encoder ne l'est pas, mais ils ont l'avantage de proposer des options puissantes qui vous sont présentées au travers de l'interview d'Elliot Mebane publiée à l'adresse suivante : http://www.adobe.com/devnet/flash/articles/selecting_video_encoder.html

Entre On2 Flix Pro et Sorenson Squeeze 4.3, quel logiciel adopter ? C'est une question à laquelle de nombreuses personnes se retrouvent confrontées. L'article auquel nous faisons référence ci-dessus vous permettra sûrement de vous faire votre propre opinion. Nous avons fait notre choix et nous allons donc vous présenter une procédure d'encodage avec Sorenson Squeeze 4.3.

Après avoir lancé Sorenson Squeeze, vous découvrez une interface très ergonomique. En cliquant sur le bouton le plus en haut à droite de la fenêtre, vous pouvez même séparer la fenêtre en deux.

Avant de démarrer l'apprentissage de l'encodage dans ce logiciel, nous devons vous présenter le menu Aspect qui vous permettra de compenser les anamorphoses de vos fichiers vidéos. Ce menu se trouve en haut de la palette.

Comment fonctionne ce logiciel ? La réponse est très simple : suivez les étapes pas à pas que nous vous avons préparées.

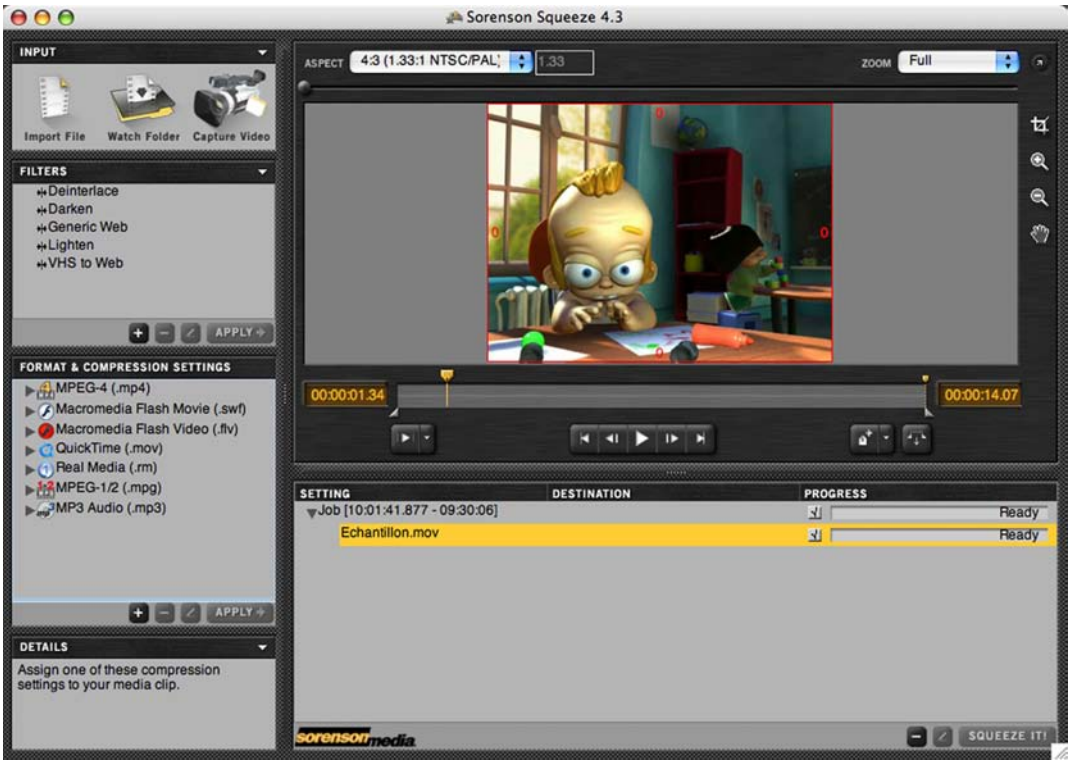


Figure 1-29

L'interface très ergonomique de Sorenson Squeeze rend sa manipulation très simple.

1. Faites glisser une ou plusieurs séquences vidéo dans la partie inférieure droite de la fenêtre (là où figure Echantillon.mov dans l'exemple de la copie d'écran de la figure 1-29). Elles vont toutes se placer les unes sous les autres. L'ajout de fichiers pour effectuer un encodage par lot est aussi simple que cela, mais il est une particularité de Sorenson Squeeze qui est extrêmement intéressante. Vous pouvez faire glisser l'icône d'un dossier complet dans la fenêtre (au même endroit que celui dans lequel vous avez placé vos fichiers) sans être obligé de prendre chaque fichier indépendamment. Alors que l'encodage aura commencé et sera même terminé, vous pourrez ajouter des séquences à ce dossier qui seront alors automatiquement encodées. L'avantage de cette technique est de n'avoir à définir qu'une seule configuration d'encodage, tous

les fichiers étant alors encodés de la même façon, avec les mêmes paramètres sans aucune intervention de votre part.

2. Pour sélectionner une préconfiguration utilisant le codec On2 VP6, cliquez sur le triangle qui figure à gauche du logo Macromedia Flash Video (figure 1-30). Apparaît alors un menu dans lequel vous pouvez sélectionner une ou plusieurs présélections de réglages (figure 1-31). Faites glisser cette sélection dans la partie inférieure droite de la fenêtre pour les associer à un fichier à compresser. Vous obtiendrez alors autant de fichiers que de réglages sélectionnés, ce qui est utile pour effectuer des essais avec plusieurs rendus.

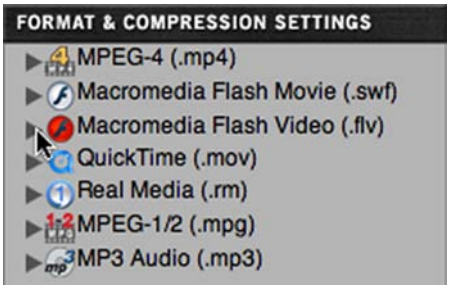


Figure 1-30

Sorenson Squeeze propose de nombreux formats d'encodage.

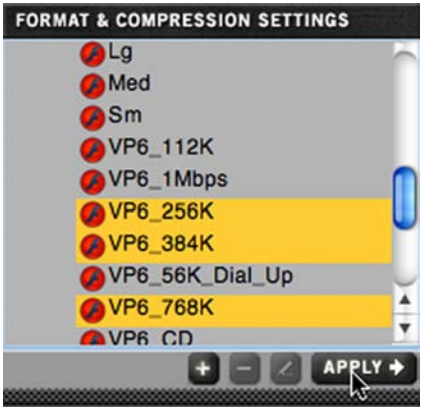


Figure 1-31

Sélection de plusieurs préréglages

3. Dans l'exemple de la copie d'écran ci-après (figure 1-32), nous avons sélectionné plusieurs préréglages afin de générer cinq fichiers que nous visualiserons pour pouvoir faire notre choix. Nous analyserons la qualité de chaque séquence, nous regarderons le poids de chaque fichier et nous prendrons une décision sur le fichier à conserver. Rappelons tout de même que pour une diffusion sur Internet, une cadence de 400 ou 512 Kbits/seconde est une bonne valeur.

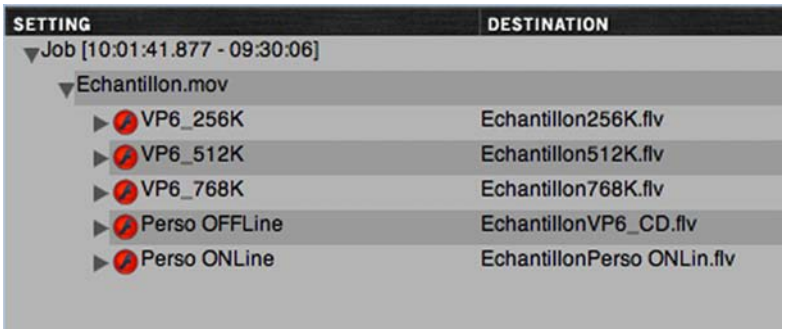


Figure 1-32

Par un simple glisser-déplacer d'un fichier dans la fenêtre, vous pouvez encoder une vidéo.

4. Nous n'allons pas demander tout de suite à Squeeze de lancer l'encodage. Nous aimerions peut-être avant cela vérifier et/ou modifier avant quelques paramètres. En cliquant sur le triangle qui se trouve à gauche de l'un des logos rouges de Flash (figures 1-32 ou 1-33) vous affichez deux options. Que vous double-cliquiez sur l'une ou l'autre, vous obtiendrez la même fenêtre, celle de la figure 1-34.

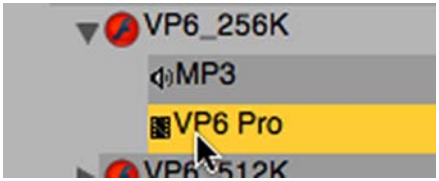


Figure 1-33

Les deux paramètres de cette préconfiguration permettent d'affiner les réglages de compression audio et vidéo.

5. Cette fenêtre (figure 1-34) est assez proche de celle que nous avons vue dans Flash Video Encoder. Vous noterez tout de même que les options y sont plus nombreuses ; observez bien, il y a même un ascenseur à droite de la fenêtre qui vous permet d'accéder à des réglages supplémentaires. Apportons quelques commentaires sur deux réglages que nous n'avons pas encore abordés.

   – `Method` : ce menu déroulant vous permet de définir le mode d'encodage. L'option d'une double passe (la première lit la vidéo, la deuxième procède à l'encodage) est un choix judicieux qui optimisera davantage le rapport qualité image/poids du fichier. La méthode VBR (*Variable Bit rate*) donne parfois de meilleurs résultats que la méthode CBR (*Constant Bit Rate*) car elle est plus évoluée. Comme son nom l'indique le débit varie en fonction du signal analogique compressé.

   – `Auto Key Frames` : vous demandez à Squeeze qu'il insère une image clé à chaque nouveau plan ou changement d'image. Cela sera ensuite plus pratique pour accéder à ces images avec la méthode `seek()` de la classe `FLVPlayback()`.

En descendant l'ascenseur, vous découvrirez des paramètres pour lesquels il est difficile de faire des commentaires. Seuls des essais vous permettront de juger d'un résultat.

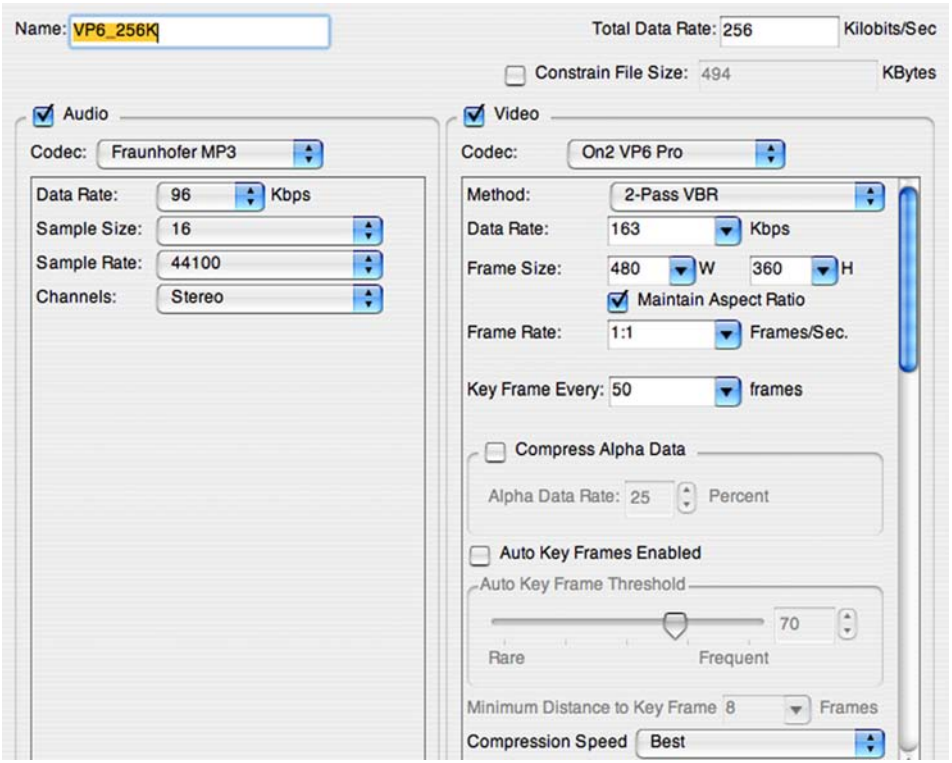


Figure 1-34

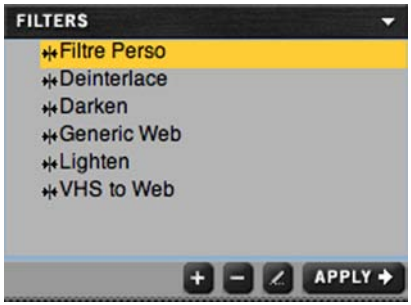
Les réglages avant encodage sont très nombreux et très précis.

> **Remarque**
>
> La case Constraint File Size s'avérera très pratique lorsque vous aurez besoin de limiter le poids d'un fichier à une valeur précise. La compression sera alors calculée en fonction.

6. Avant de lancer l'encodage, vous pouvez encore procéder à certains réglages complémentaires. En haut à gauche de l'interface (figure 1-35), des préréglages peuvent être glissés-déplacés pour les associer à un fichier ou un dossier. Ainsi, pour désentrelacer une vidéo provenant d'une cassette DV, vous choisirez le filtre Deinterlace ou un autre dans lequel il faudra activer ce paramètre (figure 1-36). De nombreuses autres options vous permettront de régler le gamma (ou les niveaux) des images de votre vidéo, la diminution d'un bruit dans les images. Vous pourrez également recadrer manuellement ou selon différents formats standards, ajouter un fondu entrant ou sortant, etc.

Figure 1-35

Des options supplémentaires permettent d'améliorer la qualité de l'image.



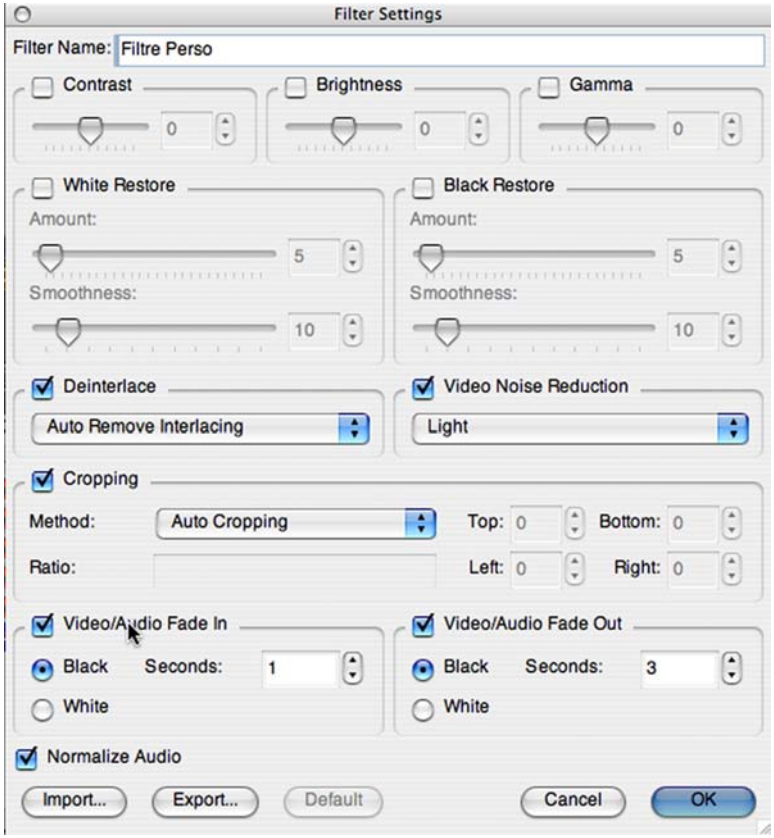


Figure 1-36

Fondus vidéos, normalisation de la piste audio, désentrelacement, etc., sont autant d'options supplémentaires que vous pouvez appliquer à une vidéo.

Rappel

Le lien vers le site TDT 3d (http://www.tdt3d.be/articles.php?art=videos_field) fait référence à la notion de trames qui occasionnent un problème d'entrelacement.

7. Nous n'avons pas abordé cette option dans Flash Video Encoder car elle fera l'objet d'un développement dans le chapitre 3 de ce livre. Attardons-nous tout de même sur cette possibilité dans Sorenson Squeeze. Dans certaines séquences, vous aurez besoin d'ajouter des images clés manuellement. Rappelons que ces dernières ont la particularité de décrire intégralement le contenu d'une image. La méthode `seek()` en ActionScript permet ainsi d'accéder à ce type d'images. Par ailleurs, vous pouvez aussi ajouter des repères appelés *cuePoints* pour détecter le passage de la tête de lecture à un instant précis de la vidéo. Le menu déroulant qui apparaît est représenté dans la copie d'écran de la figure 1-37. Il vous permet de placer des images clés ou des cuePoints pour les intégrer directement au fichier .flv. Au chapitre 3, nous verrons qu'il est préférable d'ajouter les cuePoints dynamiquement. En revanche, l'ajout d'images clés ne peut se faire qu'au travers de l'interface des logiciels de montage vidéo ou de Sorenson Squeeze.

Figure 1-37

Images clés et repères (cuePoints) peuvent être ajoutés directement à partir de l'interface de Squeeze.

Dès lors que vous aurez ajouté des repères et/ou images clés, la ligne Markers s'ajoutera sous le codec que vous avez choisi dans la partie inférieure droite de la fenêtre. Double-cliquez sur cette ligne pour faire apparaître la fenêtre représentée ci-après. Vous pouvez changer la position, le type des images clés et/ou des repères. Pour ces derniers, vous pouvez, même modifier leurs noms. Nous ne développerons pas ici la notion de variables associées à un chapitre, mais gardez à l'esprit que cela est possible dans Squeeze.

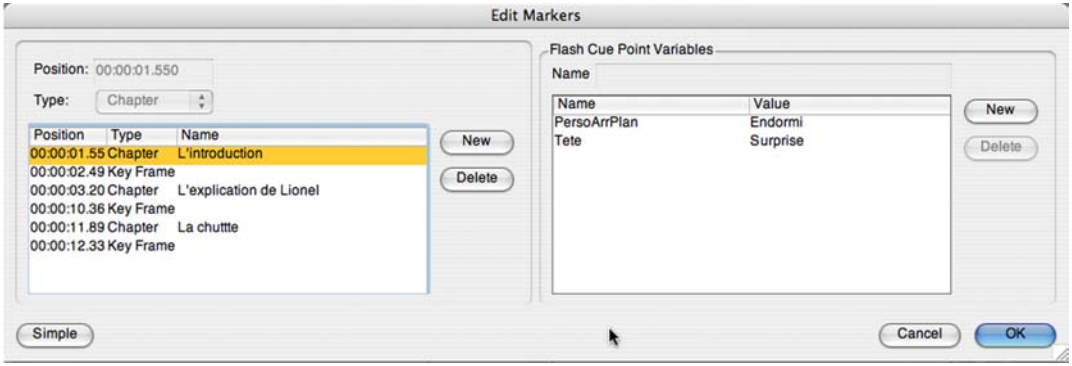


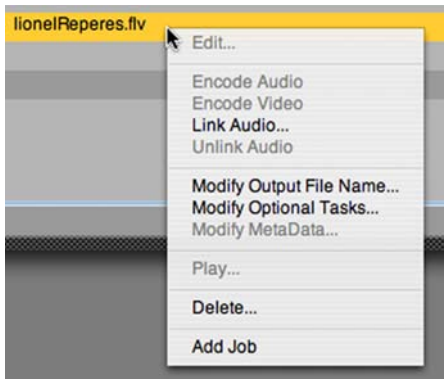
Figure 1-38

Cette fenêtre vous permet d'éditer facilement et précisément les informations relatives aux images clés et repères de votre séquence.

8. Pour finir, vous avez la possibilité de changer le nom des fichiers que vous allez obtenir au format FLV en sélectionnant la commande Modify Output File Name lors d'un clic droit sur une séquence.

Figure 1-39

Un clic droit sur une séquence vous permet d'accéder au menu qui vous propose une commande pour renommer vos fichiers.



Si vous avez besoin de transférer automatiquement vos fichiers encodés sur un serveur via un protocole FTP, vous pouvez sélectionner la commande Modify Optional Tasks du menu que nous venons d'évoquer à la figure 1-39.

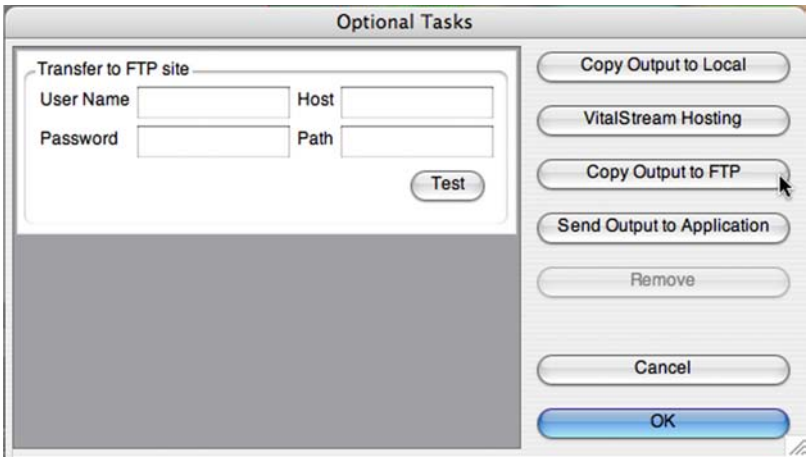


Figure 1-40

Une fois votre encodage terminé, une option vous permet de télécharger automatiquement vos fichiers au format FLV.

Voilà, vous savez à présent préparer et encoder vos vidéos pour pouvoir les manipuler dans Flash. Le chapitre suivant est dédié à l'apprentissage de la manipulation des occurrences de type FLV Playback. Si vous ne savez pas manipuler le XML dans Flash, ni gérer des listes avec le composant List, nous vous invitons à consulter au préalable le chapitre 7.

2

Insérer et contrôler une vidéo dans un SWF

Le premier chapitre de ce livre était consacré aux techniques de préparation d'une vidéo pour une utilisation dans une animation Flash. Nous allons à présent découvrir qu'il est très simple et surtout très rapide de placer un fichier au format FLV sur la scène. Pour être plus précis, nous n'allons pas importer une vidéo dans une animation, mais définir une liaison entre une partie du fichier SWF et un FLV.

Détecter la version d'un plug-in

Commençons par rappeler que le plug-in de la dernière version de Flash n'est pas forcément installé sur la machine de l'internaute qui cherche à lire votre vidéo au format SWF et que cela peut poser problème.

Au fur et à mesure de l'évolution du logiciel, chaque nouvelle version s'accompagne en effet d'un nouveau plug-in capable de lire les fichiers générés avec les nouvelles fonctionnalités. Les players antérieurs sont donc incompatibles avec les fichiers SWF les plus récents car ils sont incapables de lire les animations contenant de nouvelles instructions. Dans le cas de la vidéo, si nous prenions l'exemple d'un internaute qui posséderait le player Flash 7, il pourrait toujours lire un SWF exporté depuis Flash 8 ou 9, mais si ce fichier contient un FLV (encodé en VP6), il ne pourrait pas être lu car à l'époque du plug-in Flash 7, le codec On2 VP6 n'existait pas.

Les vidéos encodées en Sorenson Spark présentent tout de même une qualité d'image inférieure à celles qui utilisent le codec On2 VP6. Il serait donc dommage de se passer de ce dernier sous prétexte d'une meilleure compatibilité avec un plus grand nombre d'internautes. Il suffit d'effectuer un test sur la version du plug-in installé sur la machine de l'utilisateur avant de proposer la lecture d'une animation récente.

Il est très simple de détecter la version d'un player Flash. Il vous suffit d'utiliser la ligne de code ci-dessous :

```
var versionPlayer:String = new String(System.capabilities.version).substr(4,1);
```

La variable `versionPlayer` aura la valeur 9 si vous possédez le player Flash 9. Vous allez sûrement avoir besoin d'effectuer un test pour comparer la valeur obtenue afin d'inviter l'internaute à télécharger la dernière version du player s'il ne l'a pas. Rappelons que dans notre premier exemple, nous avons typé notre variable, et qu'il s'agit d'une valeur de type texte. Vous devez donc utiliser des guillemets comme vous le montre l'exemple suivant :

```
if (versionPlayer == "9") {
    messagePlayer_inst.text = "Vous possédez le dernier player";
} else {
    messagePlayer_inst.text = "Veuillez cliquer sur le lien ci-dessous...";
}
```

Nous avons sur la scène un texte dynamique dont le nom d'instance est `messagePlayer_inst`. Il va afficher (contenir) l'une ou l'autre des phrases ci-avant en fonction de la version du player Flash installée sur la machine de l'utilisateur.

Dans le cas où vous ne seriez pas familiarisé avec le typage des variables, nous pouvons simplifier la première ligne d'instruction de la façon suivante :

```
tempoPlayer = new String(System.capabilities.version );
versionPlayer = tempoPlayer.substr(4, 1);
```

Vous connaissez à présent la technique qui permet de savoir si l'utilisateur possède la dernière version du player. Passons à la partie suivante qui va vous démontrer qu'il est extrêmement simple de placer une vidéo sur la scène.

Utiliser le composant FLV Playback

Voilà le moment tant attendu ! Nous allons placer une séquence vidéo dans une animation. Vous avez préparé votre vidéo et l'avez exporté au format FLV ; voyons à présent quelles sont les manipulations à effectuer côté Flash.

1. Commencez tout d'abord par afficher la palette Composants via le menu Fenêtre. Elle contient le FLV Playback – Player 8 dont nous avons besoin.

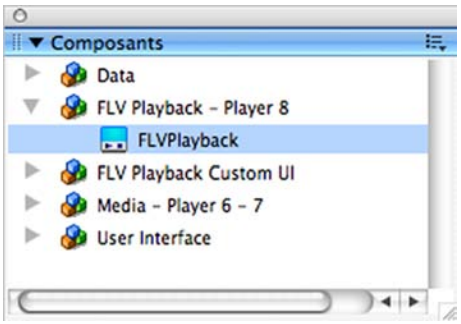


Figure 2-1

La palette Composants contient de nombreux symboles préprogrammés, communément appelés des composants (d'où le nom de cette palette) dans plusieurs domaines : Data, médias et interface.

2. Faites glisser ce composant (FLVPlayback) sur la scène. Vous obtenez alors une occurrence que vous devez nommer via la palette Propriétés (figure 2-2). Dans notre exemple, nous avons choisi `ecranVideo` car c'est un nom que nous utiliserons systématiquement dans les exemples de ce livre.

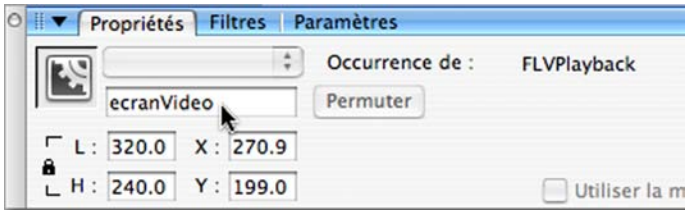


Figure 2-2

Ce nom va être utilisé en ActionScript pour contrôler le déroulement et l'interactivité relative à vos séquences vidéos. Rappelons que le nom que vous choisissez ne doit pas contenir d'espace, ni de caractères accentués ou spéciaux.

3. Cliquez sur l'image clé (du scénario) qui contient l'occurrence que vous venez de nommer.

> **Remarque**
>
> Vous n'êtes pas obligé de choisir précisément cette image clé, mais le script que nous nous apprêtons à saisir doit au moins se trouver sur la même image que celle qui contient l'occurrence du composant FLV Playback.

4. Cliquez dans la fenêtre Actions et saisissez la ligne d'instruction ci-après :

```
ecranVideo.contentPath = "lionel.flv";
```

Voilà, vous pouvez à présent tenter de publier votre animation afin de visualiser la vidéo (Cmd+Entrée sur Mac ou Ctrl+Entrée sur PC).

Si vous souhaitez contrôler la lecture de la vidéo, descendez l'ascenseur de la palette Paramètres jusqu'à la ligne Skin, cliquez à droite de son intitulé et choisissez un contrôleur. Il existe huit configurations différentes de quatre couleurs.

Si les possibilités de lecture d'une vidéo dans une animation Flash se limitaient à cette ligne d'instruction, il n'y aurait pas besoin d'écrire un livre. Découvrons donc dans les pages qui vont suivre les possibilités que nous offre Flash pour gérer la vidéo dans une animation SWF.

> **Attention**
>
> Nous pourrions définir, dans la palette Paramètres (figure 2-3), le nom (chemin) de la vidéo à diffuser au travers de l'occurrence du FLV Playback. Malheureusement et très souvent, Flash utilise une adresse absolue et non relative. De ce fait, la lecture de votre vidéo ne fonctionne plus lorsque vous mettez votre animation en ligne (sur Internet) ou changez de volume (disque dur ou amovible). Nous préférons donc passer par le code car vous ne risquez pas de commettre une erreur (en dehors d'une erreur de syntaxe).

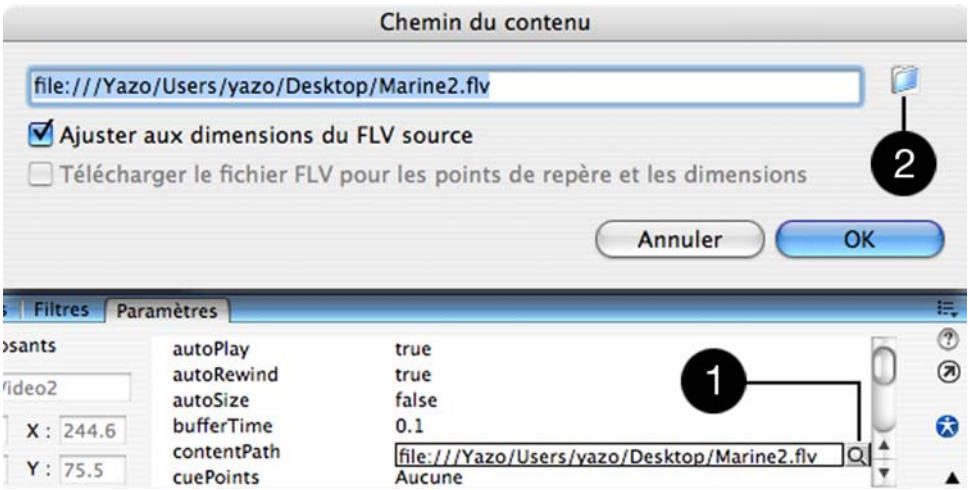


Figure 2-3

Évitez d'utiliser la palette Paramètres pour définir le chemin de votre fichier FLV, vous risqueriez d'obtenir une adresse absolue en suivant les étapes 1 et 2 (de la copie d'écran ci-après) normalement conseillées par l'aide de Flash.

Programmer le composant FLV Playback

Que doit-on comprendre au travers de ce titre, *Programmer le composant FLV Playback* ?

Jusqu'à présent, nous avons visiblement réussi à diffuser une vidéo sur la scène et nous pouvons la contrôler. Que nous manque-t-il ?

Il serait très long et inutile d'énumérer tout ce que nous allons faire en programmation pour contrôler une vidéo, mais posez-vous tout de même les questions suivantes :

Comment feriez-vous pour :

- Arrêter ou lire une vidéo sans passer par le contrôleur proposé ?
- Régler le volume de la vidéo sans passer par le contrôleur proposé ?
- Déplacer la tête de lecture à un instant précis ?
- Synchroniser la vidéo avec l'affichage de textes ou des images ?
- Changer de vidéo ?
- Enchaîner des vidéos ?

La réponse à toutes ces interrogations est la même : il suffit d'utiliser l'ActionScript, et pour cela, nous allons donc programmer le composant FLV Playback.

Nous allons commencer par vous présenter les propriétés élémentaires au contrôle d'une vidéo.

Remarque

Afin de mieux comprendre la partie du livre qui va suivre, nous vos invitons à créer une animation que vous enregistrerez sous le nom de test.fla dans un dossier qui contient une vidéo au format FLV qui s'intitule lionel.flv.

Avant le lancement de la lecture

Vous venez d'indiquer le chemin (le nom) de la vidéo que vous souhaitez diffuser sur la scène. Vous avez donc utilisé la ligne d'instruction suivante :

```
ecranVideo.contentPath = "lionel.flv";
```

Vous pourriez aussi préciser ces quatre paramètres (propriétés) supplémentaires :

```
ecranVideo.autoPlay = false;
ecranVideo.autoRewind=false;
ecranVideo.autoSize = false;
ecranVideo.maintainAspectRatio=true;
```

Paramètre autoPlay

Il indique à Flash que la séquence doit ou ne doit pas être lue automatiquement au lancement de l'animation. Si vous réglez sagement cette valeur sur `false`, l'utilisateur devra cliquer sur le bouton de lecture du contrôleur vidéo (ou sur celui que vous aurez prévu à cet effet), lui laissant ainsi le choix de l'instant où il va consulter la séquence.

Attention

Par défaut la propriété `autoPlay()` est réglée sur `true` lançant ainsi automatiquement la lecture de la vidéo.

Paramètre autoRewind

Si la valeur de cette propriété est réglée sur `false`, cela indique à Flash de ne pas replacer la tête de lecture au début de la séquence lorsque cette dernière est terminée.

Attention

Par défaut la propriété `autoRewind()` est réglée sur `true`. Cela a pour effet de donner l'impression d'un saut d'image puisque la dernière image d'une séquence et la première sont rarement les mêmes. Nous vous conseillons donc de régler la valeur de cette propriété sur `false`, d'autant plus qu'en cliquant sur le bouton Lecture, le retour à la première image de la séquence est automatique.

Paramètre autoSize

Dans certains cas, vous serez peut-être amené à charger dynamiquement des vidéos (les unes après les autres, mais pas en même temps), dans une même occurrence de composant de type FLV Playback. Si toutes les séquences possèdent la même largeur et la même hauteur, vous n'aurez pas besoin de la propriété `autoSize`. Sur la scène, vous donnerez des dimensions initiales à votre occurrence issue du symbole FLV Playback, et vous pourrez

Remarque

Dans les paragraphes ci-dessous, nous allons faire référence, aux dimensions (la taille) d'une vidéo, mais également à son rapport. La taille ou les dimensions d'une séquence se définissent par l'expression d'une largeur et d'une hauteur. Le rapport est quant à lui représenté par le calcul du ratio de la largeur par rapport à la hauteur. Une vidéo aux dimensions 320 × 240 a un rapport 4:3. Une vidéo dont la taille est de 640 × 360 a un rapport 16:9.

ensuite charger des séquences avec la méthode `load()` ou la propriété `contentPath`. Si vous devez travailler avec des fichiers FLV de tailles différentes, cette propriété sera donc le moyen de présenter toutes vos séquences à la même taille sur la scène. Elles seront en effet redimensionnées à la taille de l'occurrence en utilisant la ligne d'instruction ci-après.

```
ecranVideo.autoSize = false;
```

Comme nous vous l'avons fait remarquer au début du paragraphe précédent, la valeur par défaut de cette propriété est `false` ; vous n'aurez donc peut-être pas besoin de spécifier cette ligne d'instruction. Dans le cas où vous auriez eu besoin de régler cette propriété à `true`, il faudra alors penser à la basculer sur `false` car cela ne se fait pas automatiquement. Si vous utilisez le paramètre `true` au lieu de `false`, les séquences seront de tailles différentes, déstructurant ainsi la mise en page de votre scène, à moins que vous procédiez à un replacement de la séquence (l'occurrence du FLV Playback) avec les propriétés `_x` et `_y` (vous pouvez aussi replacer les éléments qui entourent votre vidéo).

Attention

Par défaut cette propriété est réglée à `false` afin d'éviter un étirement ou une réduction de la vidéo.

Paramètre maintainAspectRatio

Cette propriété permet d'indiquer à Flash de conserver le rapport d'une vidéo ou au contraire de l'anamorphoser. Il devrait être assez rare que vous ayez besoin d'utiliser cette propriété en réglant sa valeur sur `false`, mais si vous aviez à le faire, gardez toujours à l'esprit que la qualité et la fluidité de la séquence risquent d'en pâtir. Sur la copie d'écran de la figure 2-4, nous avons voulu vous démontrer que le chargement de deux vidéos aux dimensions différentes se fait sans problème (si ce n'est la déstructuration de la mise en page). En revanche, la figure 2-5 nous apporte la preuve qu'il ne faut pas régler cette propriété sur `false` si la taille de l'occurrence sur la scène est mal réglée.

Remarque

Si vous combinez les propriétés `autoSize` et `maintainAspectRatio`, cette dernière ne sera jamais prise en compte car la mise à l'échelle prime sur le respect du rapport.

Si vous deviez un jour travailler avec une vidéo qui présente une anamorphose, la propriété `maintainAspectRatio` réglée à `false` vous permettra de redonner le bon rapport à votre séquence.

Figure 2-4

La propriété maintainAspectRatio a été réglée sur true.

Figure 2-5

La propriété maintainAspectRatio a été réglée sur false.

Par ailleurs, certaines vidéos ont parfois des dimensions légèrement différentes, comme 384×288, 382×288, 380×286, etc. (un recadrage de la séquence a peut être été effectué au moment de l'encodage). Rappelons qu'un quart de format DV affiché sur un écran d'ordinateur correspond aux dimensions 384×288. Si vous souhaitez donc remplir complètement l'espace consacré à la vidéo sur votre scène afin qu'il n'y ait pas de blanc

à droite et/ou sous l'occurrence FLV Playback, vous devez régler la valeur de la propriété `maintainAspectRatio` sur `false`. Attention tout de même de vérifier que ce choix de votre part n'abîme pas trop la qualité de l'image comme nous l'évoquions précédemment.

Attention

Par défaut, cette propriété est réglée à `true` afin d'éviter une anamorphose de la vidéo. Il n'est donc pas obligé de la régler, c'est-à-dire de consacrer une ligne d'instruction y faisant référence dans votre script.

Remarque

Si vous placez la propriété `maintainAspectRatio` après la propriété `contentPath`, vous risquez de rencontrer des problèmes de chargement de la vidéo.

Informations sur l'état de la vidéo

Parfois, vous aurez besoin de connaître l'état de chargement d'une vidéo, ses dimensions, sa cadence ou bien d'autres informations. Consultez le paragraphe *Informations sur l'état de la vidéo* à la page 55.

Recentrer une vidéo sur la scène

Si vous chargez des vidéos dynamiquement sur la scène, il est possible que certaines d'entre elles ne possèdent pas la même largeur et ou la même hauteur. Vous allez donc devoir procéder à un repositionnement de l'occurrence du FLV Playback.

Dans la copie d'écran de la figure 2-4, vous pouvez constater que la vidéo de droite n'est pas centrée. Nous aimerions la replacer automatiquement au centre de la zone prévue, lorsque la fin de son chargement est effective, avant qu'elle ne s'affiche sur la scène. Utilisez le script ci-dessous :

```
ecranVideo.contentPath = "Marine3_240.flv";
//
var largeurEcranVideoOrigine:Number = ecranVideo._width;
var positionXEcranVideoOrigine:Number = ecranVideo._x;
//
var surveil:Object = new Object();
surveil.metadataReceived = function() {
  decalageAAjouter = (largeurEcranVideoOrigine-ecranVideo.metadata.width)/2;
  ecranVideo._x = positionXEcranVideoOrigine+decalageAAjouter;
};
ecranVideo.addEventListener("metadataReceived", surveil);
```

Retrouvez cette animation dans le dossier KTechniques, sous le nom de recentrerVideo.fla.

La première ligne d'instruction est celle qui permet de charger la vidéo dans l'occurrence du composant FLV Playback.

Les deux variables intitulées `largeurEcranVideoOrigine` et `positionXEcranVideoOrigine` stockent comme leurs noms l'indiquent, ces informations dont nous allons avoir besoin pour notre calcul de repositionnement.

Nous créons ensuite un objet afin de créer un écouteur. Si vous ne connaissez pas le fonctionnement de ce type de programmation, consultez la section *Informations sur l'état d'une vidéo* page 55 qui y fait référence.

Le calcul repose ensuite sur la différence de taille entre l'emplacement d'origine et la taille réelle de la vidéo chargée. Nous prenons la moitié de la valeur obtenue et nous repositionnons l'occurrence, sans oublier d'ajouter la valeur de la marge d'origine qui se trouve entre l'occurrence et le bord gauche de la scène.

Remarque
Si la hauteur de la vidéo chargée est plus grande que celle de l'occurrence du FLV Playback, le recentrage ne se fera pas correctement.

Gestion des skins

Commençons par rappeler qu'une *skin* permet de définir l'apparence du contrôleur d'une vidéo. Plusieurs configurations permettent d'utiliser ou de ne pas utiliser les éléments qui composent un contrôleur. Par défaut, Flash en propose 32. Nous avions vu dans le chapitre 1 qu'il existe quatre séries de huit skins différentes.

Une skin n'est ni plus ni moins qu'un simple fichier au format SWF qui contient une structure précise. Ce dernier est rattaché à l'animation qui contient votre vidéo par un simple lien qu'il est possible de modifier en ActionScript. Bien sûr, si vous faites référence à une skin (un fichier SWF), assurez-vous qu'elle soit présente au chemin que vous indiquez en paramètre de la propriété `skin`. Voici un exemple :

```
ecranVideo.play("lionel.flv");
btApparenceGrise.onPress = function() {
  ecranVideo.skin = "ClearOverAll.swf"
};
btApparenceBleue.onPress = function() {
  ecranVideo.skin = "ArcticOverAll.swf"
};
```

Deux occurrences ont été placées sur la scène, et un clic sur l'une d'entre elles permet de charger une skin, c'est-à-dire changer de contrôleur.

Pour obtenir des skins disponibles en standard, recherchez-les sur votre disque dur. Elles se trouvent dans le dossier qui contient l'application de Flash, et plus précisément dans le sous-dossier `Skins` du dossier `Configuration`. Voici la liste de toutes celles qui existent en standard.

SteelOverPlaySeekMute.swf	SteelOverPlayMute.swf	SteelOverNoVol.swf	SteelOverAll.swf
SteelExternalPlaySeekMute.swf	SteelExternalPlayMute.swf	SteelExternalNoVol.swf	SteelExternalAll.swf
MojaveOverPlaySeekMute.swf	MojaveOverPlayMute.swf	MojaveOverNoVol.swf	MojaveOverAll.swf
MojaveExternalPlaySeekMute.swf	MojaveExternalPlayMute.swf	MojaveExternalNoVol.swf	MojaveExternalAll.swf
ClearOverPlaySeekMute.swf	ClearOverPlayMute.swf	ClearOverNoVol.swf	ClearOverAll.swf
ClearExternalPlaySeekMute.swf	ClearExternalPlayMute.swf	ClearExternalNoVol.swf	ClearExternalAll.swf
ArcticOverPlaySeekMute.swf	ArcticOverPlayMute.swf	ArcticOverNoVol.swf	ArcticExternalPlaySeekMute.swf
ArcticExternalPlayMute.swf	ArcticExternalNoVol.swf	ArcticExternalAll.swf	ArcticOverAll.swf

Consultez la dernière partie de ce chapitre qui est consacrée à la personnalisation des skins.

Pendant la lecture

Au cours de la projection d'une séquence, sans passer par le contrôleur, nous pouvons non seulement contrôler sa lecture, son volume, la position de sa tête de lecture, mais également obtenir des informations relatives à l'état de la séquence (problèmes de chargement, chargement terminé, lecture en cours, en pause ou terminée, etc.). Ce sont autant d'informations dont nous avons besoin pour mieux gérer ce média dans une animation. Commençons donc par découvrir les méthodes les plus simples.

Contrôler la lecture de la vidéo

Si vous avez chargé votre vidéo dans l'occurrence du FLV PlayBack en utilisant la propriété `contentPath` sans même avoir fait référence à la propriété `autoPlay` (réglée à `true` par défaut), sachez que vous n'aurez pas besoin de demander la lecture comme nous le précisions dans les pages précédentes. En revanche, si vous avez fait appel à la méthode `load()`, il faudra cliquer sur le bouton Lecture pour lancer la séquence.

La ligne d'instruction ci-après permet donc de voir la vidéo dans l'occurrence du composant FLV Playback après un chargement avec la méthode `load()` ou une pause avec la méthode `pause()`.

```
ecranVideo.play();
```

Pour information, gardez à l'esprit que vous pouvez saisir la ligne d'instruction ci-dessous. Dans ce cas, vous n'avez plus besoin d'utiliser la méthode `load()` ni la propriété `contentPath`.

```
ecranVideo.play("lionel.flv");
```

Les paramètres que vous pouvez passer entre les parenthèses des méthodes `play()` et `load()` sont :

- le nom d'un fichier FLV en local (sur l'ordinateur) ;
- le nom d'un fichier FLV sur un serveur en utilisant le protocole http ;
- le nom d'un fichier FLV sur un serveur en utilisant le protocole rtmp.

Pour arrêter la lecture d'une vidéo, vous disposez de deux méthodes intitulées `pause()` et `stop()`. Elles assurent toutes les deux la même fonction, à savoir, bloquer la tête de lecture sur une image de la vidéo. Ce qui fait la différence entre ces deux méthodes relève de la propriété `autoRewind`. Sachez en effet que si vous avez réglé la valeur de cette dernière propriété sur `false`, les méthodes `pause()` et `stop()` fonctionne de la même façon : elles bloquent la tête de lecture. Si vous ne changez donc pas la valeur de la propriété `autoRewind`, la méthode `stop()`, interrompra la vidéo et placera la tête de lecture sur la première image de la séquence.

Contrôler le volume

La propriété qui permet de contrôler le volume d'une séquence est très singulière. Utilisez la ligne d'instruction ci-après.

```
ecranVideo.setVolume(50)
```

La valeur que vous pouvez saisir entre les parenthèses doit être comprise entre 0 et 100. Dans l'exemple ci-dessus, le volume est réglé à un niveau moyen.

Pour baisser le volume progressivement sous forme de fondu, placez un symbole sur la scène, nommez l'occurrence obtenue `btBaisserVolumeVideo`, puis utilisez le script ci-après :

```
btBaisserVolumeVideo.onPress = function() {
  this.onEnterFrame = function() {
         ecranVideo.volume -= 2;
         if(ecranVideo.volume<=0) delete this.onEnterFrame
  };
};
```

La valeur `2` peut être remplacée par une autre plus petite (y compris une valeur décimale) afin d'allonger la durée du fondu. Si vous maîtrisez la fonction `setInterval()`, privilégiez cette dernière technique.

Pour augmenter le son remplacer `-=2` par `+=2` et `<=0` par `>=100`.

Maintenant que nous avons découvert les méthodes élémentaires de la classe `FLVPlayback()`, essayons de comprendre la structure du script qui va suivre.

Informations sur l'état de la vidéo

Imaginez les situations suivantes :

- Vous aimeriez afficher *Vidéo en pause* sur la scène, lorsque l'utilisateur met effectivement la séquence en pause.
- Vous aimeriez afficher *Chargement de la vidéo en cours* sur la scène tant que l'utilisateur n'a pas encore vu la moindre image de la vidéo dont il a demandé le chargement.
- Vous aimeriez afficher sur la scène *Cliquez sur une autre vignette ci-dessous pour voir une autre vidéo*, lorsque la lecture d'une vidéo est terminée.
- Vous aimeriez afficher sur la scène *N'affichez pas la solution tant que la vidéo n'est pas terminée* durant la lecture d'une séquence.
- Vous aimeriez afficher sur la scène *Problème de connexion ou fichier FLV introuvable* si la vidéo ne peut être jouée pour l'une de ces deux raisons.

Toutes les situations auxquelles nous faisons référence ci-avant se produisent fréquemment. Vous pouvez d'ailleurs afficher des pictogrammes plutôt que d'utiliser des textes. Utilisez alors les méthodes `loadMovie()` ou `attachMovie()`.

Pour pouvoir être informé par Flash des différents états énumérés ci-après, vous devez saisir quelques lignes de code qui vont être chargées de vous alerter en temps voulu.

- Mise en mémoire cache des premières secondes de vidéo.
- Problème de connexion dû à un dysfonctionnement du réseau ou un fichier introuvable.
- Lecture en cours.
- Recherche d'un point temporel dans la vidéo.
- Pause demandée par l'utilisateur.
- Fin de la lecture (fin de la vidéo atteinte ou arrêt de la part de l'utilisateur).

Le script qui va suivre commence à se compliquer légèrement, mais sa structure est logique car il s'agit d'un écouteur.

> **Remarque**
> Si la première ligne d'instruction ci-après vous semble complexe, vous pouvez la simplifier en écrivant uniquement `surveil = new Object();`.

```
var surveil:Object = new Object();
surveil.stateChange = function(information) {
  trace(information.state);
};
ecranVideo.addEventListener("stateChange", surveil);
```

La première ligne informe Flash qu'il doit créer un objet auquel nous allons associer un événement (ligne 2).

Si cet événement se produit, la ou les lignes contenues entre les accolades vont s'exécuter. Dans la mesure où nous pourrions définir plusieurs événements et en interrompre certains, nous devons spécifier à Flash l'instant où il doit activer la surveillance de l'événement. La dernière ligne déclenche l'écouteur !

La copie d'écran de la figure 2-6 nous montre le résultat de l'exécution du script ci-dessus. Ces valeurs sont celles que renvoie la propriété `state`.

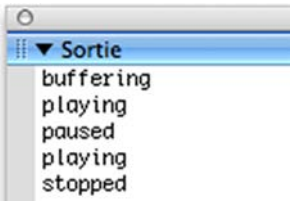


Figure 2-6

Ces messages inscrits dans la fenêtre Sortie de Flash sont obtenus grâce à la commande trace() du script ci-dessus.

L'événement `stateChange` est assez particulier car il regroupe un ensemble d'événements que nous allons vous présenter dans le tableau ci-après. Il présente tout de même

l'inconvénient de devoir tester la valeur renvoyée par la propriété `state` comme dans l'exemple suivant :

```
var surveil:Object = new Object();
surveil.stateChange = function(information) {
  if (information.state == "buffering") {
        vMessageVideo = "Veuillez patienter, chargement de la vidéo en cours";
  }
  if (information.state == "paused") {
        vMessageVideo = "Pause";
  }
  if (information.state == "playing") {
        vMessageVideo = "";
  }
};
```

Remarque

Le mot `information` contenu entre les parenthèses de la fonction puis entre les accolades devant la propriété `state` a été choisi de façon arbitraire. Vous pouvez donc utiliser n'importe quel mot tant que vous n'utilisez pas de caractères accentués, spéciaux ou des espaces.

Dans l'exemple qui suit, si nous avons simplement besoin de tester la pause d'une vidéo, il suffit d'utiliser l'événement `paused`.

```
var surveil:Object = new Object();
surveil.paused = function(information) {
  vMessageVideo = "Pause";
};
ecranVideo.addEventListener("paused", surveil);
```

Nous devons reconnaître que si plusieurs événements devaient être définis puis surveillés (écoutés) il serait préférable d'utiliser l'événement `stateChange` et de tester la valeur renvoyée par la propriété `state`.

```
var surveil:Object = new Object();
surveil.paused = function(information) {
  vMessageVideo = "Pause";
};
ecranVideo.addEventListener("paused", surveil);
//
surveil.playing = function(information) {
  vMessageVideo = "";
};
ecranVideo.addEventListener("playing", surveil);
```

La propriété `state` ne permet pas de renvoyer toutes les valeurs que nous pourrions avoir besoin de tester. Ainsi, le passage de la tête de lecture à des instants clés de la vidéo ne peut être signalé avec l'événement `stateChange`. Si vous avez besoin de détecter l'un des événements du tableau ci-dessous, utilisez-le en remplacement de ceux que nous venons de vous démontrer dans le dernier script (avec `paused` et `playing`).

Tableau 2-1 Tous ces événements peuvent être utilisés dans un écouteur.

Événement	Description
`buffering`	Cet événement permet de connaître l'instant où l'animation démarre le chargement d'une vidéo. Ne confondez pas cet événement avec `progress`.
`complete`	Cet événement sera très pratique lorsque vous souhaiterez exécuter une ou plusieurs lignes d'instructions à la fin d'une séquence. C'est donc celui qu'il vous faut pour pouvoir enchaîner des vidéos les unes après les autres.
`cuePoint`	Cet événement vous ouvre les portes à la synchronisation entre une vidéo et l'exécution de lignes d'instructions. Référez-vous au chapitre 3 qui traite de la problématique de la synchronisation.
`fastForward`	Utilisez cet événement lorsque vous souhaitez exécuter des lignes d'instructions au moment où la tête de lecture est avancée vers une image précise de la vidéo.
`metadataReceived`	Cet événement vous permet d'obtenir des informations sur une vidéo, avant même la fin de son chargement. Ces informations telles que la largeur, la hauteur, le nombre d'images/seconde et bien d'autres, vous permettent d'anticiper sur certaines manipulations à effectuer avant l'affichage de la séquence sur la scène. Retrouvez l'ensemble de ces propriétés dans le tableau suivant.
`paused`	Lorsque l'utilisateur met une vidéo en pause, utilisez cet événement pour l'en informer ou exécuter d'autres lignes d'instructions.
`playheadUpdate`	Utilisez cet événement si vous devez exécuter des lignes d'instructions durant la lecture d'une vidéo. Par exemple, pour afficher le timecode d'une vidéo, gérer et animer la progression d'une jauge de lecture, déplacer une occurrence pour représenter la tête de lecture (souvent sous forme d'un triangle pointant vers le haut).
`playing`	Utilisez cet événement lorsque vous souhaitez exécuter une ligne d'instruction au lancement de la vidéo. Consultez l'événement précédent si vous ne l'avez pas fait (`playheadUpdate`).
`progress`	Utilisez cet événement pour gérer et animer une barre (jauge) de progression du chargement d'une vidéo.
`ready`	Lorsque le chargement d'une vidéo est terminé, cet événement vous prévient.
`resize`	Si l'occurrence (du FLV Playback) contenant la vidéo est redimensionnée, cet événement vous prévient du changement. Utilisez les propriétés `width` et `height` pour connaître les nouvelles valeurs.
`rewind`	Utilisez cet événement lorsque vous souhaitez exécuter des lignes d'instructions au moment où la tête de lecture est reculée vers une image précise de la vidéo.
`scrubFinish`	Lorsque vous utiliserez le curseur de la barre de lecture du contrôleur (en fonction de la skin choisie), cet évènement vous renverra le timecode de l'image sur laquelle vous vous êtes arrêté.
`scrubStart`	Lorsque vous utiliserez le curseur de la barre de lecture du contrôleur (en fonction de la skin choisie), cet événement vous renverra le timecode de l'image à partir de laquelle vous démarrez votre recherche.
`seek`	Événement comparable à `fastForward` et `rewind`.
`skinError`	Si le fichier de la skin de votre vidéo ne peut être chargé, cet événement vous en informe.
`skinLoaded`	Si le fichier de la skin de votre vidéo a été correctement chargé, cet événement vous en informe.
`stateChange`	Si vous n'avez pas lu le développement qui précède ce tableau, consultez-le.
`stopped`	Dès que la vidéo est arrêtée au moyen de la méthode `stop()`, cet événement vous en informe.
`volumeUpdate`	Dès que le volume de la vidéo est modifié, cet événement vous en informe.

Remarque

Ne confondez pas les événements `playing` et `playheadUpdate` ainsi que `buffering` et `progress`. Il y a de très fortes chances que vous utilisiez `playheadUpdate` et `progress` pour animer des barres (jauges) de lecture ou de chargement.

Le tableau ci-après liste toutes les propriétés que vous pouvez employer avec la propriété `metadata` que vous devez utiliser avec l'événement `metadataReceived`.

Il est important de rappeler que, grâce à cet événement et ses propriétés, vous obtenez des informations avant même le démarrage du chargement de la vidéo. En effet, la propriété `contentPath`, et les commandes `load()` et `play()` chargent d'abord les métadonnées du fichier vidéo avant les images et la bande son qu'il contient.

Tableau 2-2 Propriétés de la propriété metadata

Propriété	Description
height	Utilisez cette propriété pour connaître la hauteur du fichier FLV en cours de chargement.
width	Utilisez cette propriété pour connaître la largeur du fichier FLV.
duration	Utilisez cette propriété pour obtenir la durée du fichier FLV en secondes.
framerate	Utilisez cette propriété pour obtenir le nombre d'images par seconde du fichier FLV.
videodatarate	Utilisez cette propriété pour connaître la vitesse de transmission vidéo du fichier FLV.
audiocodecid	Utilisez cette propriété pour connaître la version (sous forme de nombre) du codec utilisée pour coder l'audio.
audiodelay	Nombre qui indique l'heure time 0 précisé dans le fichier FLV d'origine. Le contenu vidéo doit être légèrement retardé pour synchroniser correctement l'audio.
audiodatarate	Utilisez cette propriété pour connaître le nombre de Ko/s de la piste audio.
videocodecid	Utilisez cette propriété pour connaître la version du codec utilisée pour coder la vidéo.

Dans le cas où vous n'auriez pas lu les pages qui précèdent ces deux tableaux, un exemple sur les propriétés de la propriété `metadata` est disponible dans ce chapitre à la section *Recentrer une vidéo sur la scène*.

En voici un deuxième qui a pour fonction de stocker dans deux variables intitulées `largeurVideo` et `hauteurVideo` les dimensions de la vidéo en cours de chargement.

```
surveil.metadataReceived = function() {
  largeurVideo = ecranVideo.metadata.width;
  hauteurVideo = ecranVideo.metadata.height;
};
ecranVideo.addEventListener("metadataReceived", surveil);
```

Pour l'événement `cuePoint`, référez-vous au chapitre 3 qui traite du problème de la synchronisation au sein d'une vidéo.

Pour les techniques des trois points ci-après, nous avons souhaité consacrer des développements spécifiques (même si nous venons d'y faire référence au travers des tableaux ci-avant), car vous risquez d'en avoir fréquemment besoin.

Afficher le timecode de la vidéo sur la scène

Placez tout d'abord sur la scène un texte dynamique dont le nom de variable est `vMessageVideo`. Utilisez ensuite le script ci-dessous :

```
var surveil:Object = new Object();
surveil.playheadUpdate = function() {
  vMessageVideo = ecranVideo.playheadTime;
};
ecranVideo.addEventListener("playheadUpdate", surveil);
```

Grâce à cet écouteur, la variable `vMessageVideo` prend, tous les 1/4 de seconde, la valeur de position en secondes et millisecondes de la tête de lecture de la vidéo. Si vous souhaitez augmenter ou diminuer l'intervalle du 1/4 de seconde, utilisez la propriété `playheadUpdateInterval` en définissant une durée exprimée en millisecondes.

```
ecranVideo.playheadUpdateInterval = 1500;
```

Cette ligne d'instruction permet de n'exécuter le code contenu dans le gestionnaire d'événement `playheadUpdate` que toutes les secondes et demie.

Déceler la fin de la vidéo

Utilisez le script ci-dessous :

```
ecranVideo.contentPath = "Marine3.flv";
var surveil:Object = new Object();
surveil.complete = function() {
  ecranVideo.contentPath = "lionel.flv";
};
ecranVideo.addEventListener("complete", surveil);
```

Ce script a pour effet de relancer une deuxième vidéo intitulée lionel.flv lorsque la première est terminée.

Afficher une barre de chargement sur la scène

Si vous faites appel à des vidéos relativement lourdes, quel que soit le débit de l'internaute, rappelons que ce dernier a besoin de savoir si le chargement de la séquence est terminé. Il peut ainsi mieux comprendre d'éventuels sauts de l'image (lorsque le chargement va moins vite que la lecture).

Dans notre exemple, nous avons placé un clip sur la scène, dont le point d'alignement est placé à gauche de la barre horizontale. Nous avons nommé l'occurrence `jauge`.

Remarque

N'utilisez pas ce genre de technique pour des consultations de vidéos off-line.

```
ecranVideo.contentPath = "http://www.yazo.net/clone.flv";
//
var surveil:Object = new Object();
```

```
surveil.progress = function(infos) {
  jauge._xscale = (infos.bytesLoaded/infos.bytesTotal)*100;
};
ecranVideo.addEventListener("progress", surveil);
```

Cet écouteur nous permet d'obtenir deux informations capitales pour l'animation de la barre de chargement : le nombre total d'octets que représente le poids du fichier FLV et le nombre d'octets déjà chargés.

Personnaliser une skin

Nous allons à présent mettre le code de côté pour revenir à des manipulations d'occurrences avec les outils de création et d'édition de la barre d'outils de Flash.

Une fois encore, essayez de suivre les explications ci-après tout en vous exerçant sur un fichier. Pour commencer, suivez les étapes ci-dessous :

1. Créez un dossier sur votre disque dur que vous nommez Essais Skins.
2. Placez dans ce dossier, une vidéo au format FLV intitulée lionel.flv.
3. Dans le dossier Flash 9/Configuration/SkinFLA de votre ordinateur, copiez un des fichiers présents pour le coller dans le dossier Essais Skins.
4. Renommez ce fichier que vous venez de coller sous le nom skinPerso.fla. Exécutez-le afin de générer un SWF (cliquez sur Ctrl+Entrée sur PC ou Cmd+Entrée sur Mac alors que vous venez d'ouvrir le FLA).
5. Créez un nouveau fichier Flash que vous enregistrez dans le dossier Essais Skins sous le nom essaiSkin.fla.
6. Dans ce fichier, placez un composant FLV Playback sur la scène et nommez l'occurrence obtenue, par exemple `ecranVideo`.
7. Saisissez le script suivant dans la fenêtre Actions en ayant préalablement sélectionné l'image clé qui contient l'occurrence du FLV Playback :

```
ecranVideo.contentPath = "lionel.flv";
ecranVideo.skin = "skinPerso.swf";
```

Vous devriez obtenir le contenu du dossier ci-après :

Figure 2-7

Vous allez éditer le fichier skinPerso.fla et le publier pour visualiser le résultat au travers du fichier essaiSkin.swf.

De façon très simple, vous pouvez commencer par changer uniquement la couleur du contrôleur. Dans l'exemple ci-dessous, nous en avons choisi une qui correspond à l'une des couleurs présentes dans la séquence.

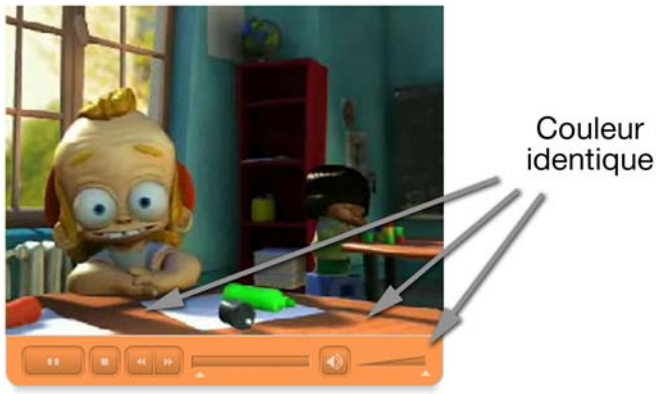


Figure 2-8

La couleur du contrôleur est la même que celle du bureau contenu dans la vidéo.

1. Ouvrez le fichier intitulé skinPerso.fla. Vous découvrez sur la scène, toutes les composantes qui constituent un contrôleur. Un bouton est composé de plusieurs apparences car il contient plusieurs états (normal, survolé, enfoncé et activé).

Figure 2-9

Toutes les composantes d'un contrôleur sont stockées dans un fichier SWF. Chaque symbole représentant un bouton ou toute autre partie du contrôleur est présent sur la scène sous forme d'occurrence.

2. Double-cliquez sur la barre qui se trouve en bas de la scène.
3. Cliquez sur le verrou de verrouillage/déverrouillage général (figure 2-10).

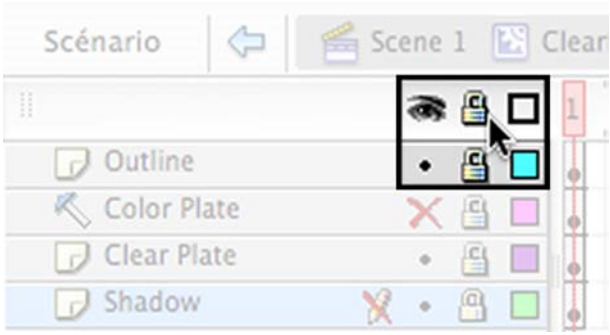


Figure 2-10
Libérez vos calques afin qu'ils puissent être édités.

4. Cliquez à nouveau sur la barre pour sélectionner la forme qu'elle contient.
5. Sélectionnez une couleur dans le nuancier ou composez-la via le mélangeur. Vous pouvez également saisir directement une valeur ou cliquer sur une partie de votre écran pour prélever une couleur dans une fenêtre qui contiendrait une image ou une vidéo (figure 2-11).

> **Remarque**
> Copiez le code couleur dans le Presse-papiers ou notez-le sur une feuille, afin de pouvoir le réutiliser ultérieurement.

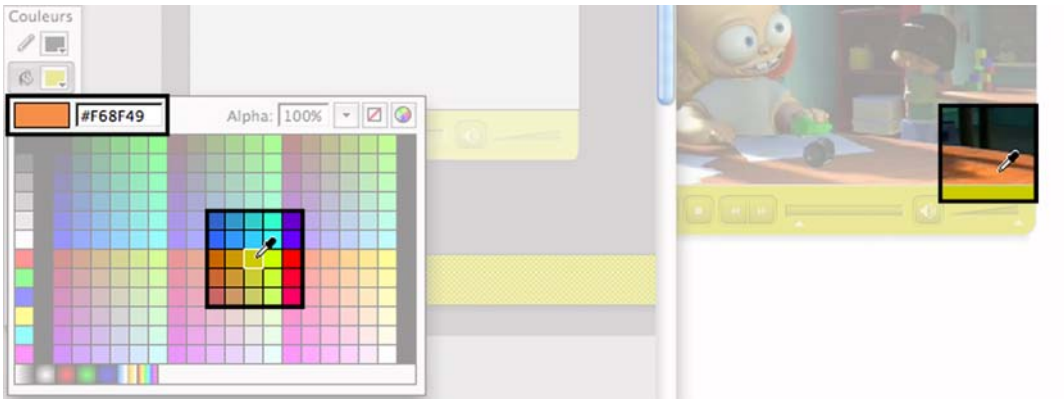


Figure 2-11
Pour sélectionner une couleur, vous disposez de plusieurs techniques dont une qui permet d'aller prélever une couleur sur une image ou une vidéo dans une fenêtre en arrière-plan.

6. Cliquez sur l'onglet Scène 1 ou Séquence 1 pour revenir sur la scène (figure 2-12).

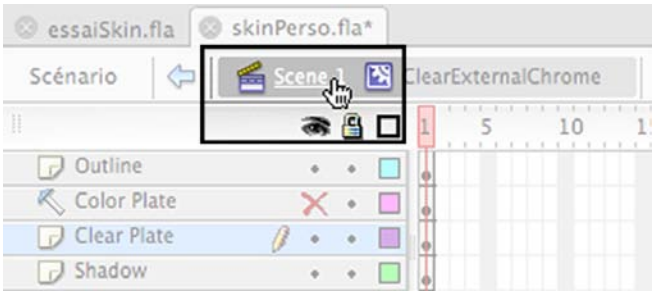


Figure 2-12

Pour revenir sur le scénario principal de l'animation, vous devez cliquer sur le bouton de retour à la scène, mais vous pouvez également utiliser le raccourci clavier Ctrl+E ou Cmd+E.

7. Vous n'avez plus qu'à effectuer le raccourci clavier Ctrl+Entrée sur PC ou Cmd+Entrée sur Mac afin de générer un SWF. Fermez cette fenêtre.
8. Revenez sur le fichier essaiSkin.fla puis effectuez à nouveau le raccourci clavier de l'étape précédente.

Comme vous pouvez le constater, la couleur de votre contrôleur a changé, mais en survolant les boutons, vous découvrez que les bordures sont encore vertes. Vous allez devoir éditer tous les symboles de type bouton et autres qui contiennent ces formes dans le fichier skinPerso.fla. Vous pouvez, soit double-cliquer sur les différentes occurrences présentes sur la scène, soit double-cliquer sur les différents symboles qui se trouvent dans les dossiers de la bibliothèque.

Si vous passez par la bibliothèque, modifiez les symboles qui se trouvent dans le dossier Wrappers. Attention, vous allez devoir effectuer de nombreux double-clics pour atteindre les différentes parties contenant les couleurs à changer. Le plus important est de penser à zoomer sur les différents contours à modifier pour faciliter leurs sélections.

Lorsque vous aurez réussi à modifier les couleurs, vous aurez compris le fonctionnement des boutons. Ils sont composés de plusieurs symboles (donc plusieurs occurrences) qui s'imbriquent parfois. Si vous souhaitez donc personnaliser l'apparence des boutons en changeant de formes, rien de plus simple : il vous suffit d'éditer les contours ou de redessiner vos propres boutons.

Utiliser les FLV Playback Custom UI

Si vous ne souhaitez pas utiliser un contrôleur complet, vous avez dans ce cas la possibilité de placer sur la scène des composants qui se trouvent dans le dossier FLV Playback Custom UI (*User Interface*).

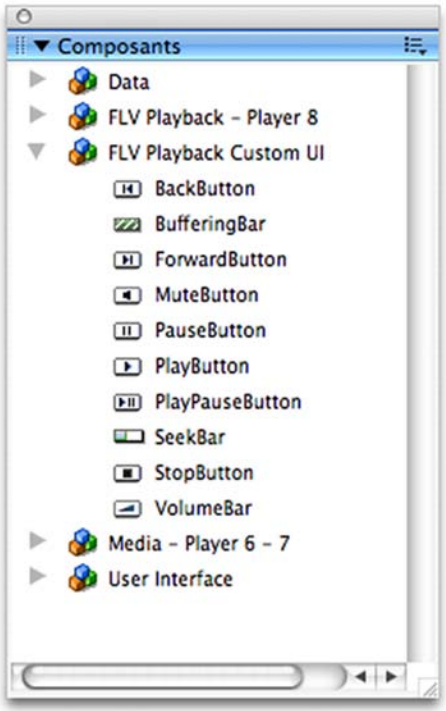


Figure 2-13

Le contrôleur n'est pas le seul composant à pouvoir contrôler une vidéo. Les symboles du dossier FLV Playback Custom UI peuvent être placés individuellement sur la scène.

Leur utilisation est très simple car il vous suffit de faire glisser sur la scène ceux que vous souhaitez utiliser, de nommer les occurrences obtenues, puis d'utiliser les propriétés appropriées pour définir des relations.

Dans l'exemple de la figure 2-14, nous avons utilisé quelques composants et nommé leurs occurrences `btDebut`, `btLecture`, `btPause`, `btMute` et `barreVolume`. Nous avons ensuite saisi le script ci-après sur la première image clé du scénario général, dans la fenêtre Actions.

```
ecranVideo.contentPath = "lionel.flv";
ecranVideo.backButton = btDebut;
ecranVideo.playButton = btLecture;
ecranVideo.pauseButton = btPause;
ecranVideo.muteButton = btMute;
ecranVideo.volumeBar = barreVolume;
```

Remarque

Attention de ne pas saisir les noms d'occurrences entre guillemets.

Figure 2-14

Vous n'êtes pas obligé d'utiliser un contrôleur, vous pouvez n'utiliser que certains boutons.

3

Synchroniser une vidéo

Les deux premiers chapitres de ce livre étaient consacrés aux techniques élémentaires de préparation et de contrôle d'une vidéo. Mais Flash ne se limite pas à ces possibilités de développement. Découvrons donc à présent des techniques plus avancées que seul le langage ActionScript peut nous permettre de déployer.

À quoi sert la synchronisation d'une vidéo ?

Depuis quelques années, les professionnels de l'Internet parlent du Rich Media qui se caractérise par le fait de pouvoir combiner dans une page Web tous les médias qui existent : texte, image, son et vidéo. Cette combinaison s'inscrit généralement dans un contexte de développement dynamique, où les données sont extraites à partir de bases de données (MySQL et XML).

Lorsque les premières vidéos ont fait leurs apparitions sur Internet, leurs utilisations se limitaient à une simple projection avec un contrôle élémentaire de la lecture. Aujourd'hui, grâce à l'avancement des différentes technologies (notamment celles de Flash et du XML), la consultation d'une vidéo peut constituer l'élément moteur d'un développement. Ce média est en effet très souvent à l'origine du déclenchement de nombreuses actions sur la scène d'une animation Flash.

Au cours de la lecture d'une vidéo dans une animation Flash, il est tout à fait possible d'afficher un texte sur la scène à un instant précis – c'est d'ailleurs un minimum à connaître – mais il est également possible de charger une image, mettre la vidéo en pause, en déclencher et/ou en charger une autre, déclencher un son, animer un clip, placer des éléments graphiques sur la vidéo, etc.

Pour répondre à la question que nous avons posée en titre à ce développement, nous pouvons dire que la synchronisation est au cœur de toutes les opérations évoquées dans

le paragraphe précédent, dans la mesure où la vidéo possède des repères (appelés aussi des *cuePoints*) et qu'elle va déclencher l'exécution de lignes d'instructions grâce à un gestionnaire d'événement.

Pour créer des points de repères dans une vidéo, vous disposez de quatre techniques qui présentent chacune des avantages et des inconvénients.

Gérer les repères dans une vidéo

La gestion des repères en Flash passe par deux étapes. Vous devez dans un premier temps ajouter des repères à une vidéo afin de la « marquer », puis vous devez programmer votre animation pour qu'elle réagisse au passage de la tête de lecture sur un repère.

> **Remarque**
> Nous appellerons toujours `ecranVideo` les occurrences de type FLV Playback.

Ajouter des repères dans une vidéo

Dès que vous placez un composant de type FLV Playback sur la scène, vous devez configurer l'occurrence obtenue en spécifiant le chemin qui relie la vidéo et l'animation, le mode de démarrage de la lecture (automatique ou manuel) et quelques réglages supplémentaires. Parmi ceux-ci, il en est un qui consiste à définir les cuePoints, c'est-à-dire les points de repères. Nous reviendrons sur cette technique plus loin dans ce chapitre. Commençons tout d'abord par vous présenter un tableau qui compare les différentes techniques de pose d'un repère.

Tableau 3-1 Avantages et inconvénients des différentes techniques d'ajout et de gestion des repères dans une vidéo

	Avantages	Inconvénients
À partir d'un logiciel d'édition vidéo ou d'encodage dédié (Sorenson Squeeze, On2Flix, Riva)	• Plus grande souplesse et meilleure précision dans la pose des repères. • Précision des repères au millième de seconde.	• Les repères sont intégrés à la vidéo et ne sont donc pas gérés dynamiquement. • Il faut penser à demander l'export des repères dans certains logiciels.
À partir du Flash Video Exporter	• L'étape de l'ajout de repères s'inscrit dans une procédure proposée par le logiciel au moment de l'encodage de la vidéo. • Grande simplicité de pose des repères. • Précision des repères au millième de seconde.	• Flash Video Exporter présente une interface qui ne facilite pas la pose des repères. • Les repères sont intégrés à la vidéo et ne sont donc pas gérés dynamiquement.
À partir de la palette Paramètres de Flash	• Édition des repères directement à partir de l'interface de Flash. • Grande simplicité et rapidité dans la pose et l'édition des repères.	• Saisie des timecodes sans contrôle visuel avec les images de la vidéo. • La gestion des repères n'est pas dynamique. • Précision des repères au dixième de seconde.
En faisant appel au langage ActionScript	• La gestion des repères est dynamique.	• Cela nécessite une première lecture de la vidéo en notant les différents timecodes des repères à marquer. • Précision des repères au dixième de seconde.

Nous faisons référence à l'aspect dynamique de la pose des repères sans en préciser le réel avantage : développons ce point.

Si vous possédez déjà un fichier vidéo au format FLV qui ne contient pas de points de repères, l'ActionScript restera la seule solution pour en ajouter dynamiquement. Par ailleurs, le choix des instants auxquels vous souhaitez ajouter des repères peut varier selon vos besoins, une pose dynamique étant une fois encore la seule solution. Lorsque des cuePoints sont posés directement dans le fichier FLV, il faut gérer leur état (actif ou désactivé) pour pouvoir les « retirer » temporairement.

Avant d'apprendre à déceler le passage de la tête de lecture sur un repère dans une vidéo, nous devons apprendre à placer des points de repères. Le tableau ci-avant indique les quatre méthodes disponibles.

Rappel

La traduction du terme *cuePoint* correspond à *repère* (de temps). Nous utiliserons donc les deux mots pour désigner la même chose.

Ce qu'il faut savoir avant de poser un repère

Pour placer un repère à un instant très précis d'une vidéo, vous allez certainement sélectionner une image bien définie. Lorsque vous programmerez ensuite votre animation, vous ferez généralement appel aux gestionnaires d'événements `cuePoint` et/ou `playheadUpdate`. Il est très important que vous preniez connaissance du développement ci-dessous car il vous permettra de comprendre certaines contraintes et limites de la pose de repères dans une vidéo en Flash.

Avant de démarrer votre travail d'insertion des cuePoints dans une vidéo, gardez toujours à l'esprit qu'il existe deux catégories de points de repères : les repères intégrés (`Navigation` et `Event`) et les repères externes (placés via la palette Paramètres de Flash ou définis en ActionScript). En fonction de la méthode que vous emploierez, vous ne bénéficierez pas de la même précision pour la détection de vos repères. Les repères intégrés permettent d'obtenir une précision de l'ordre des millisecondes alors que les repères externes se limitent au dixième de seconde. De ce fait, il est important d'utiliser la propriété `playheadUpdate-Interval` pour optimiser la détection des points de repères.

Remarque

Lorsque vous chercherez à connaître le type d'un repère, trois valeurs seront susceptibles de vous être renvoyées : `Event`, `Navigation` et `ActionScript`. La pose de points de repères via la palette Paramètres de Flash revient à exécuter un script en ActionScript.

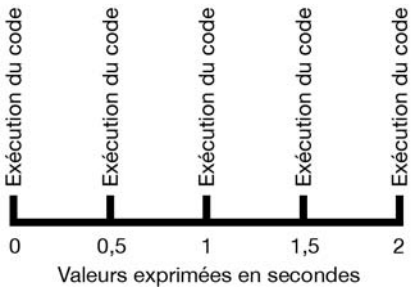
L'événement playheadUpdate

Il est utilisé à la place de la fonction `setInterval()` ou d'un `_root.onEnterframe` pour pouvoir exécuter des lignes d'instructions au cours de la lecture d'une vidéo. Comme vous le savez sûrement, la vitesse de répétition des lignes d'instructions contenues dans la fonction

associée à l'événement `onEnterFrame` est relative à la vitesse de l'animation, qui est exprimée en images/secondes. Pour la fonction `setInterval()`, la vitesse est définie entre les parenthèses en tant que paramètre. L'événement `playheadUpdate` est quant à lui invoqué selon un intervalle-temps défini par la propriété `playheadUpdateInterval`. Nous reviendrons sur cette dernière un peu plus loin.

Figure 3-1

Toutes les demi-secondes l'événement playheadUpdate est invoqué pour exécuter les lignes d'instructions contenues dans la fonction qui lui est rattachée.



Le script ci-dessous affiche dans la fenêtre Sortie de Flash, la position de la tête de lecture toutes les demi-secondes.

```
ecranVideo.playheadUpdateInterval = 500;
surveil = new Object();
surveil.playheadUpdate = function() {
  trace(ecranVideo.playheadTime);
};
ecranVideo.addEventListener("playheadUpdate", surveil);
```

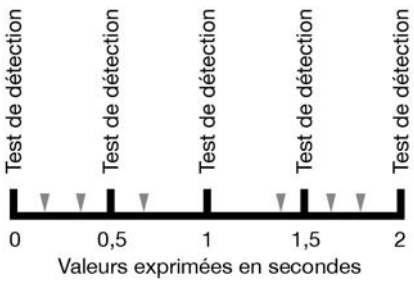
Pour plus d'explications sur ce script, consultez la section *Détecter les repères d'une vidéo* de ce chapitre.

L'événement cuePoint

Il permet de détecter le passage de la tête de lecture sur un repère, mais cette vérification est effectuée cycliquement. Pour être très précis dans notre propos, un test est effectué à intervalle régulier pour vérifier si la tête de lecture n'a pas rencontré un repère. Il y a donc un décalage entre la position du repère et la vérification. Pour diminuer cet écart, il faut donc réduire la valeur de l'intervalle-temps entre deux cycles.

Figure 3-2

La détection des repères a lieu toutes les demi-secondes, et consiste à déceler la présence des repères apparus depuis la dernière détection. Les triangles gris représentent les différents repères placés dans la vidéo.



Le script ci-après affiche dans la fenêtre Sortie de Flash le nom du ou des repères rencontrés depuis la dernière vérification.

```
ecranVideo.playheadUpdateInterval = 500;
surveil = new Object();
surveil.cuePoint = function(repere) {
  trace(repere.info.name);
};
ecranVideo.addEventListener("cuePoint", surveil);
```

Pour plus d'explications sur ce script, consultez la section *Détecter les repères d'une vidéo* de ce chapitre.

La propriété playheadUpdateInterval

Dans les deux exemples ci-avant, nous avons volontairement défini la valeur de la propriété `playheadUpdateInterval` à `500` afin de vous démontrer le fonctionnement des deux gestionnaires au travers des schémas.

L'événement cuePoint

Dans la pratique, il serait impensable de choisir une telle valeur (`500`), car une demi-seconde représente un intervalle-temps bien trop grand pour la détection de points de repères. L'idéal serait de pouvoir définir un intervalle irrégulier qui se calerait précisément sur les timecodes des repères, mais cela est impossible. Pour résoudre ce problème, la solution est de diminuer l'intervalle-temps en le ramenant à 100 ou 50/1000 de seconde. La valeur par défaut est 250/1000 (soit un quart de seconde), si vous ne modifiez pas la propriété `playheadUpdateInterval`.

L'événement playheadUpdate

Le réglage de la valeur de la propriété `playheadUpdateInterval` dépend de votre besoin. Si vous devez réaliser une jauge de progression qui représente le pourcentage de lecture de votre vidéo ou la position de la tête de lecture, vous avez intérêt à utiliser une valeur comprise entre 10 et 250. Vous devez en effet mettre à jour l'échelle de votre occurrence (la jauge) le plus rapidement possible, ainsi que le temps affiché dans le texte dynamique sur la scène.

Dans l'exemple ci-après, si nous ne changeons pas la valeur par défaut de la propriété `playheadUpdateInterval`, nous obtenons un décalage entre les timecodes de chaque repère et celui de la tête de lecture.

```
surveil = new Object();
surveil.cuePoint = function(repere) {
  trace(repere.info.time+" - "+ecranVideo.playheadTime);
  trace("_____");
};
ecranVideo.addEventListener("cuePoint", surveil);
```

Voici le contenu de la fenêtre Sortie.

```
4.44132285495191 - 4.321
--------------------------------
8.96266954512819 - 8.842
--------------------------------
16.0047670448718 - 15.882
--------------------------------
```

Pour plus d'explications sur ce script, consultez la section *Détecter les repères d'une vidéo* de ce chapitre.

En conclusion, si vous ne modifiez pas la valeur de la propriété `playheadUpdateInterval` alors que vous placez deux repères à moins d'une demi-seconde d'intervalle entre deux cycles, vous ne pourrez pas gérer correctement votre interactivité. Généralement, la valeur par défaut (250/1000) n'est pas modifiée pour ne pas trop solliciter le microprocesseur de l'ordinateur de l'utilisateur. Il est tout de même conseillé dans certains cas d'abaisser cette valeur entre 10 et 100 pour forcer la machine à réagir encore plus rapidement au passage de la tête de lecture sur un repère ou mettre à jour plus rapidement un changement sur la scène.

La méthode seek()

Vous utiliserez cette méthode pour déplacer la tête de lecture de votre vidéo à un instant précis. Malheureusement, le résultat que vous obtiendrez dépend du mode de diffusion de la vidéo.

En téléchargement progressif, vous ne pourrez atteindre qu'une image clé, et notamment celle qui se trouve après le temps spécifié en paramètre (valeur placée entre les parenthèses) de la méthode. Vous ne pouvez pas afficher une image de la vidéo à un temps précis.

En consultation d'une vidéo à partir d'un serveur de stream, vous pourrez afficher sans aucun problème une image correspondant au temps précisément indiqué.

Ajouter des cuePoints à partir de Sorenson Squeeze

1. Lancez l'application Sorenson Squeeze. Apparaît alors la fenêtre représentée à la figure 1-27 du chapitre 1.
2. Faites glisser une vidéo dans la fenêtre.

> **Remarque**
>
> Consultez, dans le chapitre 1, le paragraphe intitulé *Utiliser un autre logiciel dédié* pour un apprentissage rapide du fonctionnement de Sorenson Squeeze.

3. Faites glisser la tête de lecture de gauche à droite, comme vous l'indique la figure 3-3, pour rechercher une image sur laquelle vous souhaitez placer un repère (cuePoint).

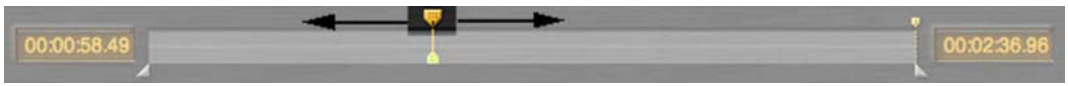


Figure 3-3

Placez la tête de lecture sur une image avant de placer un repère.

4. Sélectionnez dans le menu local déroulant situé à droite de la fenêtre principale, la commande Chapitre (*Chapter*) ou Image clé (*Key Frame*) (figure 3-4).

Figure 3-4

L'ajout d'un repère peut être de deux natures : un chapitre ou une image clé.

Vous pouvez également cliquer directement sur le bouton qui contient le signe + sans être obligé de sélectionner une commande du menu. Cela ajoutera un repère de type identique au dernier repère posé.

> **Remarque**
> Une fois que vous avez positionné la tête de lecture sur une image pour y placer un cuePoint, vous pouvez alors appuyer sur la touche M de votre clavier pour insérer un repère ou une image clé.

Dans les deux cas, l'événement `cuePoint` détectera les repères de type Chapitre et Image clé. Cela dit, les éléments suivants sont à prendre en considération :

- Les commandes `seekToNavCuePoint()`, `seekToNextNavCuePoint()` et `seekToNextPrevCuePoint()` ne fonctionnent qu'avec les repères de type chapitre.
- La commande `seek()` placera généralement correctement la tête de lecture au temps demandé lorsque vous avez créé un repère de type image clé.

Rappelons que Sorenson Squeeze est un logiciel capable d'encoder dans différents formats, y compris ceux qui sont dédiés à générer des fichiers destinés à être gravés sur DVD. On comprend ainsi pourquoi le terme Chapitre est utilisé. L'aide en ligne de Flash nous montre qu'il existe trois types de cuePoints : Event (événement), Navigation et ActionScript. Dans ce cas, à quoi correspondent les repères Chapitre et Image clé que propose Sorenson Squeeze ? La copie d'écran ci-après montre qu'ils sont similaires aux repères de type Navigation et Event.

Position	Type	Name
00:00:04.48	Chapter	Chapter
00:00:09.00	Chapter	Chapter
00:00:16.04	Key Frame	
00:00:24.92	Key Frame	

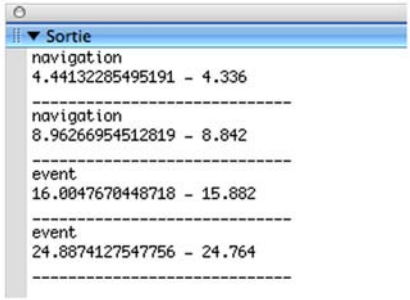


Figure 3-5

Les types de repères Chapitre et Image clé correspondent à Navigation et Event. Le tableau de gauche est une copie d'écran de la fenêtre Markers de Sorenson Squeeze. La fenêtre de droite correspond à ce que renvoie Flash lorsque nous cherchons à connaître les types des cuePoints avec plusieurs commandes trace().

Quelle différence doit-on faire entre les cuePoints de type Navigation et Event (événement) ? Pour ne rien vous cacher, dès que vous cherchez à obtenir des informations sur cette distinction, non seulement les explications ne sont pas toujours très claires, mais en plus elles ne se vérifient pas systématiquement. Voyons tout de même les définitions données par l'éditeur du logiciel.

Points de repère de navigation : vous intégrez des points de repères de navigation dans le flux FLV et le paquet de métadonnées FLV lorsque vous encodez le fichier FLV. Vous utilisez les points de repères de navigation pour permettre aux utilisateurs de rechercher une partie spécifiée d'un fichier.

Points de repère d'événement : vous intégrez des points de repères dans le flux FLV et le paquet de métadonnées FLV lorsque vous encodez le fichier FLV. Vous pouvez rédiger un code pour manipuler les événements qui sont déclenchés à certains points spécifiés pendant la lecture.

Dans la pratique, il faut reconnaître que pour utiliser les méthodes `seekToNavCuePoint()` et `seekToNextNavCuePoint()`, vous devez créer des points de repères de type Navigation. En revanche, le gestionnaire d'événement `cuePoint` fonctionne avec les deux types (Event et Navigation) contrairement à ce qui est stipulé dans la définition ci-avant. Par ailleurs, la méthode `seek()` qui permet de rechercher une partie spécifiée du fichier (comme le stipule une fois encore la définition ci-avant) fonctionne avec les points de repères de type Event.

À l'occasion de l'écriture de ce livre, de très nombreux essais ont été réalisés pour essayer d'éclaircir toutes les incompréhensions qui tournent autour de cette distinction. Dans la plupart des cas, nous avons pu constater que les points de repères de type Image clé, créaient généralement des images clés aux endroits spécifiés.

Ajouter des cuePoints à partir de Flash Video Encoder

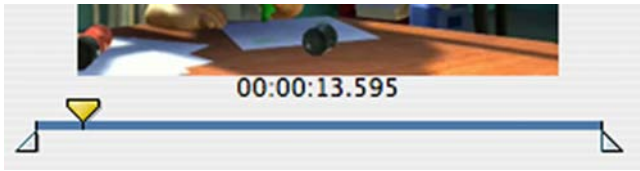
1. Lancez l'application Flash Video Encoder qui se trouve à la racine du dossier de l'application Flash.
2. Faites glisser un fichier dans la fenêtre principale comme le montre la figure 1-25 du chapitre 1.
3. Cliquez sur le bouton Paramètres…

> **Remarque**
>
> Si le bouton Paramètres... ne peut pas être sélectionné, cliquez sur la ligne qui contient le nom de votre fichier.

4. Cliquez sur le bouton Afficher les paramètres avancés puis sur l'onglet Points de repères.
5. Recherchez l'image sur laquelle vous souhaitez placer un repère, en faisant glisser de gauche à droite le triangle qui se trouve sous la vidéo. Vous pouvez également utiliser les flèches de votre clavier (gauche et droite) pour un déplacement image par image (figure 3-6).

Figure 3-6

Le triangle vous permet de sélectionner une image sur laquelle vous souhaitez placer un repère.



6. Cliquez sur le bouton + qui se trouve à gauche de la fenêtre afin d'ajouter un repère. Vous devez alors nommer le repère et choisir un type. Consultez les deux pages précédentes pour faire la distinction entre des repères de type Navigation et de type Event.
7. Pour ajouter des paramètres associés au repère cliquez sur le bouton + qui se trouve dans la partie droite de la fenêtre. Précisez le nom du paramètre ainsi que sa valeur. Un paramètre, appelé *attribut*, permet d'associer des informations à un repère, informations qu'il sera possible de lire en faisant appel à un script.
8. Répétez les étapes 5 à 7 autant de fois que nécessaire.

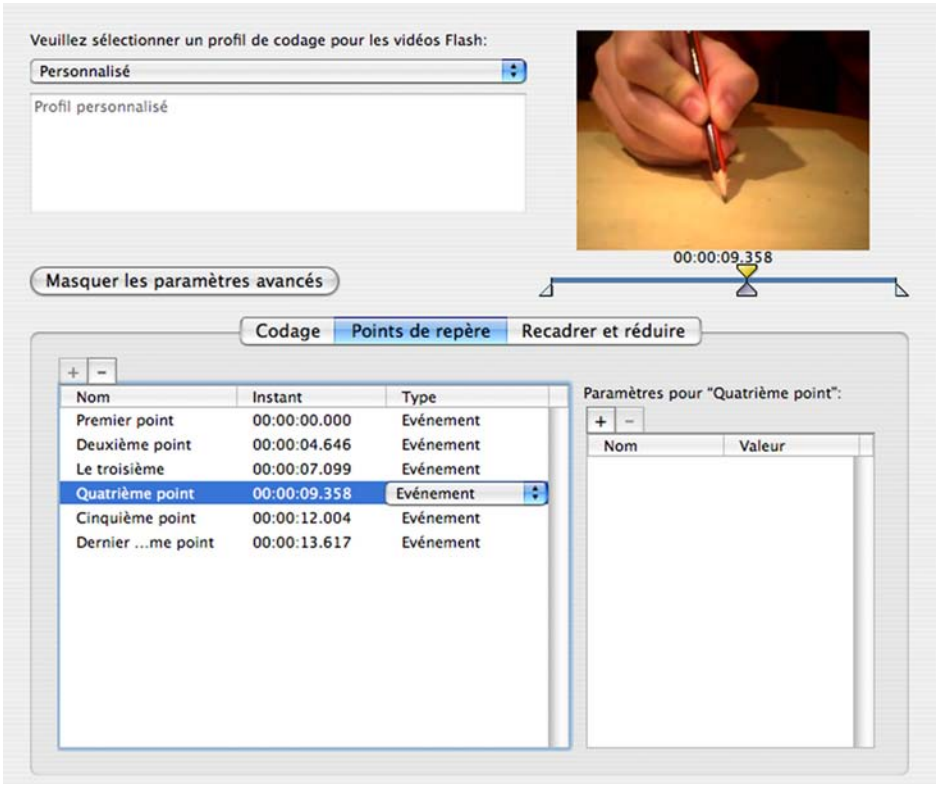


Figure 3-7

La gestion des points de repère se fait facilement, mais manque de précision à travers l'interface de Flash Video Encoder.

Ajouter des cuePoints à partir de la palette Paramètres de Flash

Avant de vous présenter la procédure ci-dessous, il est important de préciser que vous ne pouvez pas utiliser cette technique d'ajout de repères si vous définissez le chemin d'accès au fichier FLV en utilisant l'ActionScript. Vous devez impérativement passer par la palette Paramètres et cliquer sur la ligne contentPath pour localiser le fichier vidéo.

1. Sélectionnez l'occurrence de votre FLV Playback sur la scène.
2. Dans la palette Paramètres, cliquez une première fois sur la ligne cuePoints. Cliquez une deuxième fois sur le mot Aucune ou sur la loupe à droite de la ligne (figure 3-8).

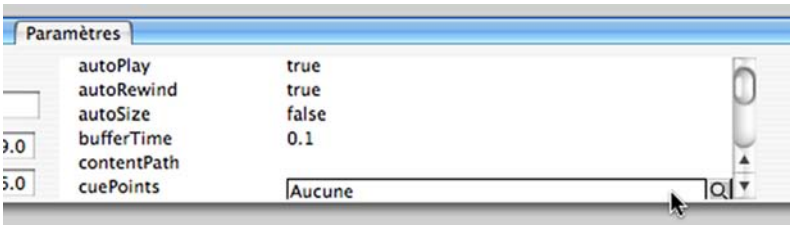


Figure 3-8

La palette Paramètres permet d'ajouter des repères très simplement.

3. Apparaît la fenêtre de la figure 3-9 dans laquelle vous allez ajouter les repères.

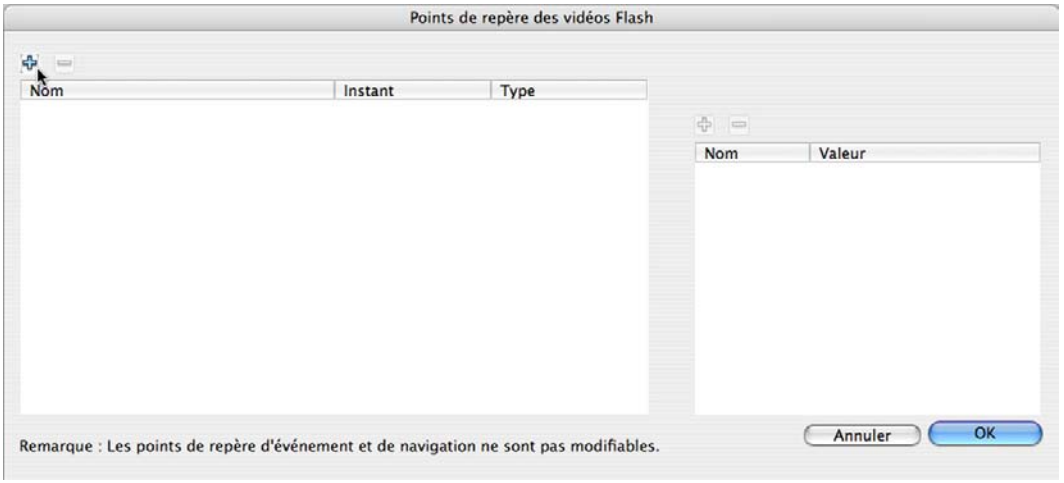


Figure 3-9

Cette fenêtre va vous permettre d'ajouter des points de repères en quelques clics seulement.

4. Cliquez sur le bouton qui contient le signe + bleu en haut à gauche de la fenêtre pour ajouter un cuePoint.

5. Définissez un nom de repère et un point temporel. Le type sera toujours ActionScript.

> **Rappel**
>
> Les cuePoints Event et Navigation sont inscrits dans la vidéo au travers des métadonnées au moment de l'encodage. Dans cette procédure en sept points, vous définissez des repères au travers d'une interface, mais vous n'allez pas pour autant encoder la vidéo.

6. Appuyez autant de fois que nécessaire sur le bouton d'ajout de repères.
7. Cliquez sur le bouton OK pour fermer la fenêtre.

Comme vous pouvez le constater, la procédure est simple et rapide. Il ne vous reste plus qu'à ajouter un script dans votre animation pour détecter le passage de la tête de lecture sur ces repères.

Ajouter des cuePoints en ActionScript

La technique qui a été présentée dans la section précédente (*Ajouter des cuePoints à partir de la palette Paramètres de Flash*) est proposée aux utilisateurs de Flash qui ne connaissent pas très bien l'ActionScript. Pour ceux qui préfèrent écrire quelques lignes d'instructions, voici une autre méthode plus souple.

> **Attention**
>
> Le composant FLV Playback génère l'événement `cuePoint` pour les points de repères de type ActionScript uniquement au moment de la mise à jour de la tête de lecture suite à l'événement `playheadUpdate`. Pour diminuer l'intervalle entre deux mises à jour, vous devez régler la propriété `playheadUpdateInterval` en utilisant une valeur très faible.

Pour ajouter des repères en ActionScript, vous disposez de la méthode `addASCuePoint()` qui se gère très facilement.

1. Saisissez le nom de l'occurrence du FLV Playback.
2. Ajoutez la méthode `addASCuePoint()`.
3. Précisez les deux paramètres attendus entre les parenthèses.

Le script ci-après a pour effet d'ajouter trois repères à une vidéo.

```
ecranVideo.addASCuePoint(2.15, "Entrez !");
ecranVideo.addASCuePoint(3.43, "Bonjour, prenez une chaîse");
ecranVideo.addASCuePoint(5.86, "Laquelle ?");
```

Sans même expliquer le rôle des paramètres évoqués ci-dessus, vous aurez compris que le premier fait référence au temps précis auquel vous souhaitez poser un repère, et que le deuxième correspond au nom du repère.

Au travers des nombreux exemples présentés dans ce livre, vous découvrirez qu'il est préférable d'utiliser des tableaux ou des fichiers XML pour insérer une quantité importante de cuePoints. Dans le cas où vous feriez appel à un document XML, il est alors judicieux

d'affecter un numéro en guise de nom, ce qui permet ensuite d'utiliser cette valeur pour parcourir l'arborescence.

Détecter les repères d'une vidéo

Attention

Si vous n'avez pas lu les pages qui précèdent, lisez attentivement la section *Ce qu'il faut savoir avant la pose d'un repère* de ce chapitre.

Quelle que soit la méthode que vous choisirez pour insérer des repères dans vos vidéos, vous pourrez utiliser les lignes d'instructions ci-après pour détecter le passage de la tête de lecture sur un cuePoint.

Remarque

Rappelons que si la première ligne vous semble complexe, vous pouvez utiliser la syntaxe suivante : `surveil = new Object()`.

```
var surveil:Object = new Object();
surveil.cuePoint = function(repere) {
  vRepere = repere.info.name;
};
ecranVideo.addEventListener("cuePoint", surveil);
```

Avant de commenter ce premier exemple et de vous présenter les différentes notions relatives à la détection des repères, commençons par préciser que nous aurions pu nous passer de la première ligne, le script devenant alors le suivant :

```
_root.cuePoint = function(repere) {
  vRepere = repere.info.name;
};
ecranVideo.addEventListener("cuePoint", _root);
```

Le strict minimum pour détecter le passage de la tête de lecture sur un repère peu consister réellement en un script de trois lignes. Dans la pratique, nous en utiliserons davantage car les instructions à exécuter lors d'une détection ne se limitent pas à l'affichage d'une information dans un texte dynamique (comme c'est le cas dans notre exemple : `vRepere = repere.info.name`).

Remarque

Afin qu'il n'y ait pas de confusion entre un objet et une occurrence, formulons la remarque suivante. Pour créer un objet, il est nécessaire de faire appel au constructeur `new` et de faire référence à une classe. Ainsi, nous pourrions écrire la ligne suivante : `monEcouteur = new Object()`. Nous venons de créer un objet, mais nous pourrions également dire que nous avons instancié la classe `Object()` pour obtenir une instance intitulée monEcouteur. De ce fait, nous obtenons une instance de la classe `Object()`. Rappelons également que le terme instance est la traduction anglaise du mot occurrence.

Les objets d'écoute

Pour détecter le passage de la tête de lecture sur un repère, il est nécessaire de programmer une surveillance permanente de la séquence vidéo. Pour ce faire, il existe une technique de développement dans certains langages qui consiste à créer un écouteur. Nous avons déjà évoqué cette notion dans le chapitre précédent, détaillons-la à présent.

Pour créer un écouteur, nous devons commencer par créer théoriquement un objet afin de lui associer un événement. Nous déclenchons ensuite la surveillance.

```
nomDunObjet = new Object()
nomDunObjet.evenement = function() {
  ligne d'instruction à exécuter si l'événement se produit
  ligne d'instruction à exécuter si l'événement se produit
  ligne d'instruction à exécuter si l'événement se produit
}
instanceAlaquelleEstAssocieeLaSurveillance.addEventListener("evenement",nomDunObjet)
```

Bien évidemment, l'exemple ci-avant ne peut fonctionner car il traduit le fonctionnement d'un écouteur.

Lorsque nous avons écrit, dans le paragraphe qui précède le script ci-dessus, que nous devions créer théoriquement un objet, pour être plus précis, nous devons en fait faire référence à une instance. Vous avez donc deux possibilités : soit vous créez un objet (une instance de la classe `Object()` ou n'importe quelle autre), soit vous faites référence à une occurrence qui existe déjà sur la scène. Ainsi, `_root` peut servir de nom d'objet auquel on associe un événement.

Lorsque vous possédez un objet, vous pouvez alors lui associer un événement. Dans le cas d'une détection de repère, vous devez utiliser l'événement `cuePoint`.

Vous devez ensuite définir une fonction qui contient toutes les lignes d'instructions à exécuter au moment de la détection.

Pour finir, vous devez enclencher l'écoute (la surveillance) de l'événement avec la méthode `addEventListener()` accompagnée de ses paramètres. Le premier fait référence à l'événement surveillé, le deuxième correspond au nom de l'objet auquel nous avons associé l'événement.

Le gestionnaire d'événement cuePoint

Après ce développement sur les écouteurs, nous devons maintenant voir plus en détail les différentes lignes d'instructions susceptibles d'être placées entre les accolades de la fonction associée à l'événement. L'exemple qui va suivre va nous permettre de découvrir trois lignes d'instructions différentes qui permettent de lire :

- le nom d'un repère ;
- le timecode (le temps) d'un repère ;
- la valeur d'un attribut associé à un repère.

Remarque

L'attribut d'un repère est aussi appelé `parameter`.

```
surveil = new Object()
surveil.cuePoint = function(repere) {
  vNomRepere = repere.info.name;
  vTempsRepere = repere.info.time;
  information1 = repere.info.parameters.quand;
  information2 = repere.info.parameters.nomPersonnage;
  information3 = repere.info.parameters.lieu;
};
ecranVideo.addEventListener("cuePoint", surveil);
```

Le paramètre `repere` (placé entre parenthèses) de la fonction nécessite une explication complémentaire. À la détection du passage de la tête de lecture sur le repère, les lignes d'instructions sont exécutées, mais celles-ci ont la particularité de lire des informations relatives au cuePoint. Le mot `repere` est en réalité un objet qui contient une propriété `info`. Cette dernière possède à son tour d'autres propriétés (`name`, `time` et `parameters`) qui vont nous permettre d'obtenir les informations que nous cherchons à lire.

Le tableau ci-dessous vous présente une synthèse des différentes informations que vous pouvez lire quand une vidéo possède des repères avec ou sans paramètres.

Remarque

Le nom de l'objet `repere` a été choisi librement ; vous pouvez utiliser n'importe quel autre nom, du moment qu'il ne contient pas d'espace et de caractères accentués.

Code à utiliser	Commentaire
nomdelobjet.info.time	Récupérer le timecode du repère
`nomdelobjet.info.name`	Récupérer le nom du repère
`nomdelobjet.info.parameters.nomparametre`	Récupérer la valeur d'un paramètre associé à un repère

Rappel

Le terme `nomdelobjet` du tableau ci-dessous est un nom d'instance, nom d'un objet que vous avez créé à partir de la classe `Object()`.

Pour une meilleure compréhension de cet exemple, nous vous présentons trois copies d'écran qui mettent en évidence les différents noms de repères, paramètres et variables utilisés dans cette animation.

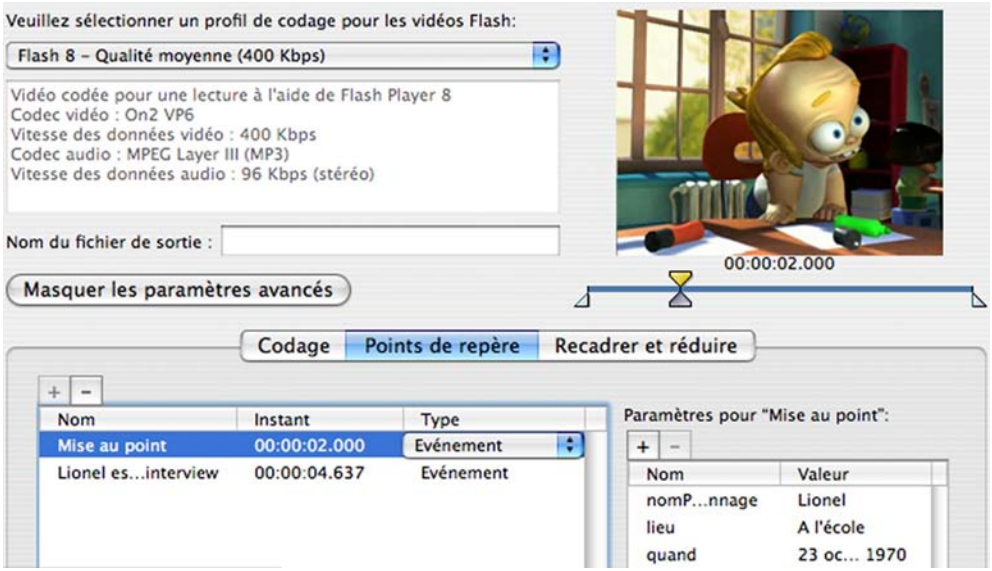


Figure 3-10

Au moment de l'encodage de la vidéo, lors de la pose des points de repères, vous pouvez définir des paramètres.

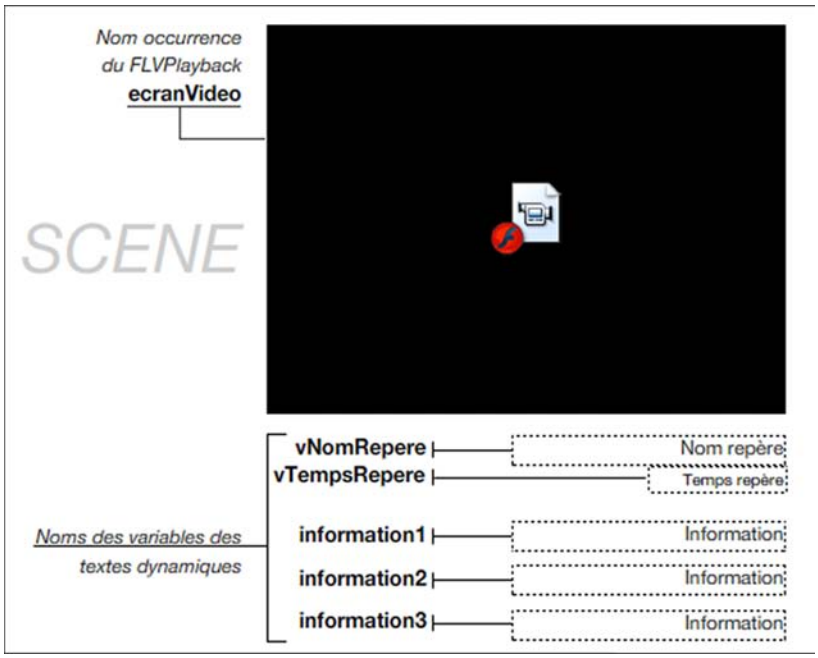


Figure 3-11

La scène de l'animation PremiersReperes.fla contenue dans le dossier KTechniques est constituée des éléments ci-dessus.

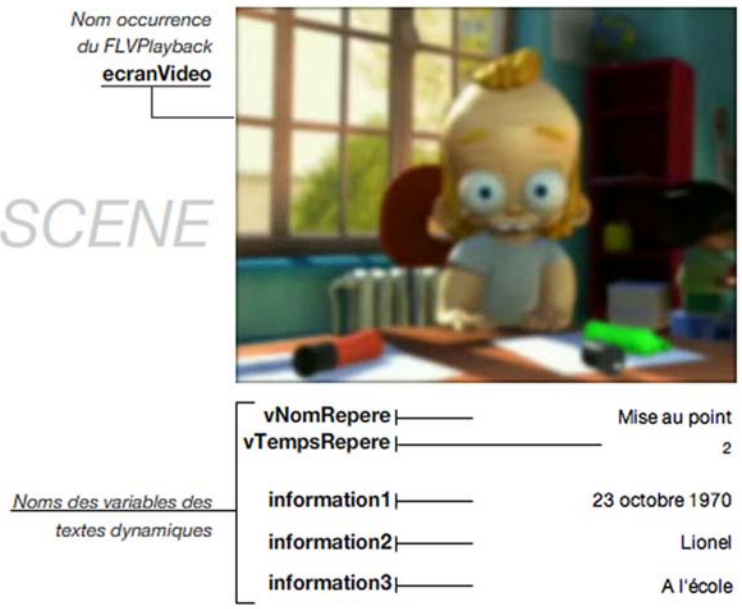


Figure 3-12

Lorsque l'événement cuePoint est invoqué, les lignes d'instructions contenues dans la fonction associée sont exécutées et remplissent les textes dynamiques présents sur la scène.

Connaître le nombre de cuePoints

Si vous souhaitez connaître ou vérifier le nombre de repères qui ont été intégrés dans une vidéo, vous pouvez utiliser le script ci-après.

```
surveil.metadataReceived = function(reperes) {
  trace(ecranVideo.metadata.cuePoints.length);
};
ecranVideo.addEventListener("metadataReceived", surveil);
```

Nous avons vu dans un chapitre précédent que cet événement permet de connaître les dimensions de la séquence. Il sert aussi à lire le nombre de repères contenus dans un FLV, si vous utilisez la propriété `cuePoints` accompagnée de `length`.

Sous-titrer une vidéo

> **Attention**
>
> Nous vous conseillons d'étudier toutes les animations les unes après les autres car elles sont de difficulté croissante.

Nous avons développé à l'occasion de la rédaction de ce livre, plusieurs animations qui font appel à des scripts différents pour gérer l'affichage de textes sur la scène en même

temps que la lecture d'une vidéo, à des instants précis. Ces animations possèdent toutefois la même construction de scène :

- une occurrence de type FLV Playback, intitulée `ecranVideo` ;
- un texte dynamique nommé `vCommentaires`.

Les fichiers d'origine se trouvent tous dans le dossier KVideosSousTitres.

> **Remarque**
> Nous utiliserons parfois le terme timecode qui fait référence à un instant précis d'une vidéo sous la forme heures:minutes:secondes:millisecondes.

Utiliser la fonction setInterval() avec un tableau et un test if()

Nous avons cherché à réaliser une première animation dont les contraintes techniques se limitent aux notions de programmation élémentaire. En dehors de `playheadTime`, il n'est pas nécessaire de connaître d'autres propriétés ou méthodes de la classe `FLVPlayback()`.

Animation : KVideosSousTitres /video_sstitrefla.fla

Vidéo : KVideosSousTitres/points.flv

```
ecranVideo.contentPath = "points.flv";
ecranVideo.autoRewind = false;
//
var repere:Number = 0;
var tempsReperes = [0, 4.646, 7.099, 9.358, 12.004, 13.617];
var textesCorrespondants = ["Premier point", "Le deuxième", "Un troisième",
➥"Quatrième", "Cinquième et avant dernier", "Le sixième et dernier"];
//
repereAtteint = function () {
  if (ecranVideo.playheadTime>tempsReperes[repere]) {
         vCommentaires = textesCorrespondants[repere];
         repere++;
         if (repere>=tempsReperes.length) {
                trace("fin");
                clearInterval(lancerSynchro);
         }
  }
};
lancerSynchro = setInterval(repereAtteint, 150);
```

Explications

Nous créons un tableau intitulé `tempsReperes` dans lequel nous stockons les différents timecodes des repères à surveiller. Dans un deuxième tableau, qui s'intitule `textesCorrespondants`, nous stockons les textes qui vont s'afficher sur la scène.

Nous créons une fonction intitulée `repereAtteint` qui est chargée de tester la position de la tête de lecture par rapport aux différentes valeurs du tableau `tempsReperes`. Cette fonction est appelée à un intervalle régulier de 150 millisecondes.

Cette technique peut convenir à des besoins limités en sous-titrage. Elle est inadaptée dans le cas où il faut gérer plusieurs dizaines de sous-titres.

Nous utilisons une variable intitulée `repere` qui va être incrémentée à chaque nouvelle détection d'un repère afin de rechercher le suivant. Sa valeur sert d'index dans la lecture des entrées des tableaux.

Utiliser la fonction setInterval() avec un fichier XML et un test if()

Nous montrons dans un deuxième et dernier exemple qu'il est possible de se passer des propriétés et méthodes de la classe `FLVPlayback()` pour gérer la synchronisation.

Animation : KVideosSousTitres/video_sstitrexmlsetInterval.fla

Vidéo : KVideosSousTitres/lionel.flv

XML : KVideosSousTitres/soustitreslionel.xml

```
ecranVideo.contentPath = "lionel.flv";
//
var replique:Number = 0;
//
var dialogues:XML = new XML();
dialogues.load("soustitreslionel.xml");
dialogues.ignoreWhite = true;
//
dialogues.onLoad = function() {
  nombreNoeuds = racine.childNodes.length;
};
//
sousTitrer = function () {
  if (ecranVideo.playheadTime>dialogues.firstChild.childNodes[replique].
  ➥attributes.temps) {
         vCommentaires = dialogues.firstChild.childNodes[replique];
         replique++;
  }
};
lancerVideo = setInterval(sousTitrer, 100);
```

Fichier XML :

```
<?xml version="1.0" encoding="UTF-8"?>
<SousTitres langue="français">
  <SousTitre temps="0" voix="Technicien">C'est parti !</SousTitre>
  <SousTitre temps="2.15" voix="Interviewer">Ca va, … ?</SousTitre>
  <SousTitre temps="3.12" voix="Lionel">Ouais.</SousTitre>
  <SousTitre temps="4.4" voix="Lionel">On peut y aller là ?</SousTitre>
  …
```

```
  <SousTitre temps="127.44" voix="Lionel">Les cheveux noirs, …</SousTitre>
  <SousTitre temps="128.93" voix="Lionel">une robe rouge.</SousTitre>
  <SousTitre temps="131.27" voix="Lionel">Y'en a qui dise …</SousTitre>
  <SousTitre temps="132.96" voix="Lionel">...</SousTitre>
</SousTitres>
```

Remarque

Le fichier XML d'origine est trop long pour être inutilement reproduit dans ce livre. Consultez le fichier d'origine pour lire l'ensemble des nœuds qu'il contient.

Explications

Nous chargeons un fichier XML qui contient tous les dialogues à afficher en tant que sous-titres. Lorsque le chargement est effectué, nous en profitons pour stocker dans une variable, le nombre de nœuds contenus dans l'arborescence.

Nous créons une fonction intitulée `sousTitrer` qui est chargée de tester la position de la tête de lecture par rapport aux différentes valeurs de l'attribut `temps` de chaque nœud. Cette fonction est appelée à un intervalle régulier de 100 millisecondes.

Contrairement à l'exemple précédent, cette technique peut convenir dans le cas où plusieurs dizaines de sous-titres doivent êtres gérés.

Dans le dossier KVideosSousTitres, un fichier intitulé video_sstitrexmlsetIntervalTempoAffich.fla contient un script comparable à celui que nous venons de voir. Il comprend quelques lignes d'instructions supplémentaires qui permettent de gérer la temporisation de l'affichage des sous-titres. Il est en effet désagréable de conserver à l'écran un sous-titre qui n'a plus lieu d'être.

```
sousTitrer = function () {
  if (ecranVideo.playheadTime>dialogues.firstChild.childNodes[replique].attributes.
  ➥temps) {
        vCommentaires = dialogues.firstChild.childNodes[replique];
        chaineCommentaire = new String(vCommentaires_inst.text);
        tempsAffichage = chaineCommentaire.length*90;
        clearInterval(temporiserAffichage);
        temporiserAffichage = setInterval(effacerCommentaires, tempsAffichage);
        replique++;
  }
};
lancerVideo = setInterval(sousTitrer, 100);
//
effacerCommentaires = function () {
  vCommentaires = "";
  clearInterval(temporiserAffichage);
};
```

Utiliser les repères intégrés à une vidéo et l'événement cuePoint

Voici le premier script optimisé ! Cette animation est extrêmement simple car elle possède un script relativement court, mais surtout efficace, car il fait appel à des instructions évoluées et concises. En revanche, il a fallu ajouter tous les repères, un à un, via Flash Video Encoder (nous aurions pu utiliser un autre logiciel dédié).

Animation : KVideosSousTitres/video_sstitreflv.fla

Vidéo : KVideosSousTitres/points.flv

```
ecranVideo.contentPath = "points.flv";
//
surveil = new Object();
surveil.cuePoint = function(recuperation) {
  vCommentaires = recuperation.info.name;
};
ecranVideo.addEventListener("cuePoint", surveil);
```

Explications

L'événement `cuePoint` accompagné d'un écouteur suffit à récupérer le nom du repère pour l'afficher sur la scène. Il n'existe pas de méthode plus simple. Rappelons que le terme `recuperation` est un nom d'objet, et que vous pouvez donc choisir n'importe quel autre nom qui ne contient ni d'espaces ni de caractères accentués ou spéciaux. Consultez le chapitre 2 pour obtenir plus de détails sur les objets d'écoute et leurs événements associés.

Utiliser des repères intégrés, un fichier XML et l'événement cuePoint

> **Rappel**
>
> Les repères intégrés sont ceux que vous avez insérés dans une vidéo en passant par Flash Video Encoder ou un logiciel dédié.

Cette technique est très intéressante car elle permet d'obtenir une grande souplesse dans le déploiement de sous-titrage. Nous avons utilisé des repères intégrés à une vidéo, mais ils ont la particularité de posséder des numéros en guise de noms. De ce fait, nous allons utiliser ces nombres en tant que numéros de nœuds pour parcourir l'arborescence d'un fichier XML.

Animation : KVideosSousTitres/video_sstitreflvNumerosXML.fla

Vidéo : KVideosSousTitres/formes.flv

XML : KVideosSousTitres/formes.xml

```
ecranVideo.contentPath = "formes.flv";
//
var chargeLegendes = new XML();
chargeLegendes.load("formes.xml");
chargeLegendes.ignoreWhite = true;
//
```

```
chargeLegendes.onLoad = function() {
  var surveil = new Object();
  surveil.cuePoint = function(recuperation) {
         vCommentaires = chargeLegendes.firstChild.childNodes[recuperation.
         ➥info.name].firstChild;
  };
  ecranVideo.addEventListener("cuePoint", surveil);
};
```

Fichier XML :

```
<?xml version="1.0" encoding="UTF-8"?>
<Legendes>
         <Legende>Construction en cours...</Legende>
         <Legende>Un triangle</Legende>
         <Legende>Construction en cours...</Legende>
         <Legende>Un carré</Legende>
         <Legende>Construction en cours...</Legende>
         <Legende>Une pyramide</Legende>
         <Legende>Fin de la construction</Legende>
  </Legendes>
```

Explications

Nous commençons par charger un ficher XML qui contient tous les sous-titres qui vont être placés sur la scène dans le texte dynamique dont le nom de variable est `vCommentaires`.

Dans le gestionnaire de vérification du chargement des données XML, nous déclarons un objet d'écoute avec l'événement `cuePoint` qui est approprié pour la technique du sous-titrage. La première ligne d'instruction, qui spécifie le chemin de la vidéo, aurait pu être saisie dans le gestionnaire `onLoad` si nous avions voulu interdire le chargement du fichier en cas d'échec du rapatriement des données XML.

Comme nous le spécifions en introduction à cette animation, chaque repère possède un numéro en guise de nom. Dès que l'événement de passage d'un cuePoint est émis, le nom du repère est alors récupéré puis utilisé pour accéder au nœud correspondant.

Ajout des repères avec la méthode addASCuePoint() et temporisation de l'affichage d'un sous-titre

Voici la première animation dont les points de repère de la vidéo ont été ajoutés en ActionScript à l'aide de la méthode `addASCuePoint()`. Cette technique ne nécessite pas de poser les cuePoints pendant l'encodage, mais a surtout pour avantage la possibilité d'utiliser un fichier XML pour référencer tous les noms et timecodes des différents repères. L'édition d'un fichier XML est toujours plus simple que la modification des cuePoints dans l'interface du Flash Video Encoder (ou un autre logiciel dédié).

Animation : KVideosSousTitres/video_sstitrexmladdCue.fla
Vidéo : KVideosSousTitres/lionel.flv
XML : KVideosSousTitres/soustitreslionel.xml

```
ecranVideo.contentPath = "lionel.flv";
//
var dialogues:XML = new XML();
dialogues.load("soustitreslionel.xml");
dialogues.ignoreWhite = true;
//
dialogues.onLoad = function() {
  racine = this.firstChild;
  nombreNoeuds = racine.childNodes.length;
  coursier = new Object();
  for (i=0; i<nombreNoeuds; i++) {
         coursier.time = Number(this.firstChild.childNodes[i].attributes.temps);
         coursier.name = this.firstChild.childNodes[i].firstChild.nodeValue;
         ecranVideo.addASCuePoint(coursier);
  }
};
//
var surveil = new Object();
surveil.cuePoint = function(passage) {
  vCommentaires = passage.info.name;
  chaineCommentaire = new String(vCommentaires);
  tempsAffichage = chaineCommentaire.length*90;
  clearInterval(temporiserAffichage);
  temporiserAffichage = setInterval(effacerCommentaires, tempsAffichage);
};
ecranVideo.addEventListener("cuePoint", surveil);
//
//
effacerCommentaires = function () {
  vCommentaires = "";
  clearInterval(temporiserAffichage);
};
```

Fichier XML :

```
<?xml version="1.0" encoding="UTF-8"?>
<SousTitres langue="français">
  <SousTitre temps="0" voix="Technicien">C'est parti !</SousTitre>
  <SousTitre temps="2.15" voix="Interviewer">Ca va, … ?</SousTitre>
  <SousTitre temps="3.12" voix="Lionel">Ouais.</SousTitre>
  <SousTitre temps="4.4" voix="Lionel">On peut y aller là ?</SousTitre>
  …
  <SousTitre temps="127.44" voix="Lionel">Les cheveux noirs, …</SousTitre>
  <SousTitre temps="128.93" voix="Lionel">une robe rouge.</SousTitre>
  <SousTitre temps="131.27" voix="Lionel">Y'en a qui dise …</SousTitre>
  <SousTitre temps="132.96" voix="Lionel">...</SousTitre>
</SousTitres>
```

Remarque

Le fichier XML d'origine est trop long pour être inutilement reproduit dans ce livre. Consultez le fichier d'origine pour lire l'ensemble des nœuds qu'il contient.

Explications

Nous chargeons les données du fichier XML que nous parsons (analysons) au travers d'une boucle `for()`. Nous créons un objet pour lequel nous définissons deux propriétés conventionnelles, `name` et `time`, puis nous utilisons la méthode `addASCuePoint()` pour intégrer les repères à la vidéo.

Nous utilisons ensuite un écouteur et l'événement `cuePoint` pour assurer la synchronisation.

Le point fort de cette animation est de pouvoir temporiser l'affichage du sous-titre. En effet, tant que la variable du texte dynamique qui se trouve sur la scène ne prend pas de nouvelle valeur, son contenu ne disparaît pas. La fonction `effacerCommentaires` est donc appelée au bout d'un laps de temps calculé en fonction du nombre de caractères contenus dans le sous-titre.

Utiliser un fichier XML et l'événement playheadUpdate

Si vous avez lu tous les scripts que nous venons d'aborder dans les pages précédentes, vous découvrirez que cette animation ressemble partiellement à plusieurs d'entre eux. Nous avons simplement voulu démontrer au travers de celle-ci que l'événement `playheadUpdate` permet de se passer de l'événement `cuePoint`. Vous pouvez ainsi centraliser toutes vos lignes d'instructions dans un même gestionnaire.

Animation : KVideosSousTitres/video_sstitrexmlplayheadupdate.fla

Vidéo : KVideosSousTitres/lionel.flv

XML : KVideosSousTitres/soustitreslionel.xml

```
ecranVideo.contentPath = "lionel.flv";
//
var replique:Number = 0;
//
var dialogues:XML = new XML();
dialogues.load("soustitreslionel.xml");
dialogues.ignoreWhite = true;
//
dialogues.onLoad = function() {
  nombreNoeuds = racine.childNodes.length;
};
//
var surveil = new Object();
surveil.playheadUpdate = function() {
  if (ecranVideo.playheadTime>dialogues.firstChild.childNodes[replique].
  ➥attributes.temps) {
```

```
            vCommentaires = dialogues.firstChild.childNodes[replique];
            replique++;
    }
};
ecranVideo.addEventListener("playheadUpdate", surveil);
```

Fichier XML :

Identique à l'animation précédente.

Cet exemple étant un regroupement des différentes techniques exposées ci-avant, vous trouverez les explications relatives à celui-ci au travers des exemples précédents.

Sous-titrage multilingue d'une vidéo

Avant d'aborder cette animation, nous vous conseillons vivement d'analyser les exemples des pages précédentes. Ils utilisent tous des techniques différentes de détection et d'insertion des repères que vous allez retrouver dans celui-ci. Gestion du XML, événement `cuePoint` et méthode `addASCuePoint()` résument l'alchimie retenue pour un sous-titrage multilingue.

Animation : KVideosSousTitres/video_sstitrexmladdCueMultilingue.fla

Vidéo : KVideosSousTitres/lionel.flv

XML : KVideosSousTitres/soustitreslionelmultilingue.xml

```
var noeudLangue:Number = 0;
var dialogues:XML = new XML();
dialogues.load("soustitreslionelmultilingue.xml");
dialogues.ignoreWhite = true;
//
dialogues.onLoad = function() {
  racine = this.firstChild;
  nombreNoeuds = racine.childNodes[0].childNodes.length;
  coursier = new Object();
  for (i=0; i<nombreNoeuds; i++) {
        coursier.time = Number(this.firstChild.childNodes[0].childNodes[i].attributes.
        ➥temps);
        coursier.name = i;
        ecranVideo.addASCuePoint(coursier);
  }
};
//
var surveil = new Object();
surveil.cuePoint = function(passage) {
  vCommentaires = dialogues.firstChild.childNodes[noeudLangue].childNodes
  ➥[Number(passage.info.name)].firstChild;
  chaineCommentaire = new String(vCommentaires);
  tempsAffichage = chaineCommentaire.length*90;
```

```
  clearInterval(temporiserAffichage);
  temporiserAffichage = setInterval(effacerCommentaires, tempsAffichage);
};
ecranVideo.addEventListener("cuePoint", surveil);
//
//
effacerCommentaires = function () {
  vCommentaires = "";
  clearInterval(temporiserAffichage);
};
//
drapeauAnglais.onPress = function() {
  noeudLangue = 1;
};
drapeauFrancais.onPress = function() {
  noeudLangue = 0;
};
drapeauAllemand.onPress = function() {
  noeudLangue = 2;
};
drapeauEspagnol.onPress = function() {
  noeudLangue = 3;
};
```

Fichier XML :

```
<SousTitres>
  <LaLangue nom="Français">
         <SousTitre temps="0" voix="Technicien">C'est parti.</SousTitre>
         …
         <SousTitre temps="128.93" voix="Lionel">une robe rouge.</SousTitre>
         <SousTitre temps="131.27" voix="Lionel">Y'en a qui…</SousTitre>
         <SousTitre temps="132.96" voix="Lionel">...</SousTitre>
  </LaLangue>

  <LaLangue nom="Anglais">
         <SousTitre temps="0" voix="Technicien">Here we go.</SousTitre>
         …
         <SousTitre temps="128.93" voix="Lionel">a red dress.</SousTitre>
         <SousTitre temps="131.27" voix="Lionel">Some say she's a witch</SousTitre>
         <SousTitre temps="132.96" voix="Lionel">...</SousTitre>
  </LaLangue>

  <LaLangue nom="Allemand">
         <SousTitre temps="0" voix="Technicien">Es geht los</SousTitre>
         …
         <SousTitre temps="128.93" voix="Lionel">ein rotes Kleid</SousTitre>
         <SousTitre temps="131.27" voix="Lionel">Einige sagen, dass …</SousTitre>
         <SousTitre temps="132.96" voix="Lionel">...</SousTitre>
  </LaLangue>
```

```
  <LaLangue nom="Espagnol">
        <SousTitre temps="0" voix="Technicien">En diamo</SousTitre>
        …
        <SousTitre temps="128.93" voix="Lionel">un vestido rojo.</SousTitre>
        <SousTitre temps="131.27" voix="Lionel">Algunos dicen que ella es una bruja
        ➥</SousTitre>
        <SousTitre temps="132.96" voix="Lionel">...</SousTitre>
  </LaLangue>
</SousTitres>
```

Remarque

Le fichier XML d'origine est trop long pour être inutilement reproduit dans ce livre. Consultez le fichier d'origine pour lire l'ensemble des nœuds qu'il contient.

Explications

Le fichier XML est constitué d'un nœud racine et de quatre nœuds enfant qui correspondent aux quatre langues disponibles à la traduction. Ils possèdent ensuite chacun des nœuds enfant qui correspondent aux sous-titres de la vidéo. Les attributs de ces balises définissent les temps des cuePoints à surveiller ainsi que le nom de la personne qui parle. Ce dernier attribut n'a pas été utilisé dans cet exemple.

Quatre gestionnaires permettent de changer l'aiguillage de la lecture du nœud enfant du nœud racine.

Comme nous l'énoncions en introduction à cette animation, les différentes parties qui constituent l'ensemble du script sont présentées dans les exemples des pages précédentes. Référez-vous à chacun d'eux pour une explication sur les différentes lignes d'instructions.

Synchroniser une vidéo avec l'affichage d'images

Une fois encore, la synchronisation peut se faire de différentes façons. Nous avons donc développé deux animations qui proposent les deux techniques suivantes.

Succession de chargements d'images dans une même occurrence

Il était très difficile de trouver un titre relativement court pour définir le rôle de cette animation. Nous allons donc analyser ce SWF qui propose un diaporama d'images qui se placent toutes au même endroit sur la scène, les unes après les autres. Chaque image remplace donc celle qui s'était précédemment chargée.

Animation : KVideoInterrompue/ video_synchro_images.fla

Vidéo : KVideoInterrompue /Tom.flv

```
numeroImage = 0;
//
surveilCues = new Object();
```

```
surveilCues.cuePoint = function() {
  numeroImage++;
  cadreChargement.loadMovie("img"+numeroImage+".jpg");
};
ecranVideo.addEventListener("cuePoint", surveilCues);
```

Explications

Le script de cette animation est très court et très simple car l'action à exécuter l'est tout autant.

Précisons avant de nous lancer dans les explications, qu'une série de quatre images intitulées img1.jpg à img4.jpg se trouvent dans le même dossier que le SWF.

Nous initialisons une variable qui va nous servir à la concaténation de la composition du nom des images (ligne 6). À chaque fois qu'un repère est décelé, la variable est incrémentée pour charger une image différente.

La méthode `loadMovie()` suffit à charger une image dans une occurrence intitulée `cadreChargement` qui se trouve sur la scène.

Chargements de plusieurs images sur la scène

Cette deuxième animation peut sembler plus complexe à la lecture du script, mais il n'en est rien. Nous avons simplement ajouté une ligne d'instruction complémentaire afin de placer un symbole sur la scène pour obtenir une occurrence à chaque détection de repère.

Animation : KVideoInterrompue/ video_synchro_images2.fla

Vidéo : KVideoInterrompue /Tom.flv

```
numeroImage = 0;
//
surveilCues = new Object();
surveilCues.cuePoint = function() {
  numeroImage++;
  _root.attachMovie("cadre", "cadre"+numeroImage, numeroImage,
  ➥{_x:620, _y:numeroImage*67});
  _root["cadre"+numeroImage].loadMovie("img"+numeroImage+".jpg");
};
ecranVideo.addEventListener("cuePoint", surveilCues);
```

Explications

La commande `attachMovie()` cible le symbole dont le nom de liaison est `cadre`. Lorsque l'événement `cuePoint` décèle un repère, la variable est incrémentée, comme dans l'animation précédente, afin de pouvoir faire référence à un nom d'image différent, mais sa valeur sert également à définir les paramètres de chaque nouvelle occurrence.

Si vous n'aviez pas l'habitude de la syntaxe `{_x:620,_y:numeroImage*67}`, sachez que vous pouvez utiliser celle-ci : `_root["cadre"+numeroImage]._x`. Cette ligne d'instruction permet

de placer sur la scène, des occurrences espacées de 3 pixels (une occurrence faisant 64 pixels de largeur et de hauteur).

Si vous deviez générer plusieurs colonnes d'images, il faudrait ajouter une variable dans le calcul de la position `_x` des occurrences.

Synchroniser une vidéo avec la méthode attachMovie()

Dans l'animation qui va suivre, nous cherchons à démontrer que la technique de synchronisation jusqu'alors utilisée pour du sous-titrage peut aussi servir à placer dynamiquement des symboles sur la scène.

Une fois encore, nous disposons de différentes techniques (que nous venons d'aborder) pour surveiller le passage de la tête de lecture à un instant précis de la vidéo. Nous avons opté pour des repères intégrés qui contiennent des paramètres et un objet d'écoute accompagné de l'événement `cuePoint`.

Animation : KVideosSousTitres/video_sstitreflvparametres.fla

Vidéo : KVideosSousTitres/3bouclesparametres.flv

```
var exemplairePuce:Number = 0;
//
ecranVideo.contentPath = "3bouclesparametres.flv";
//
var surveil = new Object();
surveil.cuePoint = function(recuperation) {
  coordonneeX = recuperation.info.parameters.coordX;
  coordonneeY = recuperation.info.parameters.coordY;
  _root.attachMovie("puce", "puce"+exemplairePuce, exemplairePuce,
  ➥{_x:coordonneeX, _y:coordonneeY});
  exemplairePuce++;
};
ecranVideo.addEventListener("cuePoint", surveil);
//
ecranVideo.autoRewind = false;
```

Explications

Nous créons une première variable dont la valeur va servir à spécifier le niveau de chaque occurrence placée sur la scène. Nous l'incrémenterons à chaque rencontre d'un nouveau repère.

Lorsque l'événement `cuePoint` décèle le passage d'un nouveau repère, la méthode `attachMovie()` place sur la scène le symbole dont le nom de liaison est `puce`.

> **Rappel**
>
> Pour définir le nom de liaison d'un symbole, effectuez un clic droit sur son icône dans la bibliothèque et sélectionnez la commande Liaison... Cochez la case Exporter pour ActionScript et définissez un nom d'identifiant. Laissez cochée l'option Exporter dans la première image.

Dans cet exemple, il est intéressant d'employer des paramètres, car ils sont obligatoirement définis et utilisés pour chaque repère. Nous accrochons tout le temps le même symbole sur la scène, mais nous aurions pu spécifier un nom de liaison de symbole dans un autre paramètre.

Temporiser des pauses en cours de lecture

Les fichiers des exemples qui vont suivre se trouvent dans le dossier KVideoInterrompue.

Pause avec redémarrage automatique

La particularité de cette animation est de proposer une lecture de vidéo qui se met en pause automatiquement lorsqu'un point de repère est atteint. Après cinq secondes d'attente, la lecture reprend sans l'intervention de l'utilisateur.

Dans cet exemple, précisons que le chemin de la vidéo a été spécifié à partir de la palette Paramètres, mais il aurait pu l'être à partir d'une ligne d'instruction.

Animation : KVideoInterrompuePlan/video_interrompue.fla

Vidéo : KVideoInterrompuePlan/Tom.flv

XML : KVideoInterrompuePlan/commentaires.xml

```
var commentaires:XML = new XML();
commentaires.load("commentaires.xml");
commentaires.ignoreWhite = true;
//
commentaires.onLoad = function() {
  nombreNoeuds = this.firstChild.childNodes.length;
  coursier = new Object();
  for (i=0; i<nombreNoeuds; i++) {
         coursier.time = Number(this.firstChild.childNodes[i].attributes.temps);
         coursier.name = i;
         ecranVideo.addASCuePoint(coursier);
  }
  ecranVideo.play();
};
//
surveilCues = new Object();
surveilCues.cuePoint = function(infoRenvoyee) {
  numeroRepere = Number(infoRenvoyee.info.name);
  vCommentaire = commentaires.firstChild.childNodes[numeroRepere].firstChild;
  ecranVideo.pause();
  relancerVideo = setInterval(pauseVideo, 5000);
};
ecranVideo.addEventListener("cuePoint", surveilCues);
//
pauseVideo = function () {
  vCommentaire = "";
  ecranVideo.play();
  clearInterval(relancerVideo);
};
```

Fichier XML :

```
<Commentaires>
  <Commentaire temps="1.35">L'enfant d'un an joue très bien avec des formes de
  ➥couleurs différentes.</Commentaire>
  <Commentaire temps="3.08">Il sait faire le ménage devant lui lorsque quelque chose
  ➥le gêne.</Commentaire>
  <Commentaire temps="7.60">Il est capable de tenir deux objets en même temps,
  ➥dans deux mains différentes</Commentaire>
  <Commentaire temps="9.96">L'enfant a entendu un bruit, il reconnaît la voix de sa
  ➥maman et lui sourit.</Commentaire>
  <Commentaire temps="12.32">Un autre son attire l'attention de l'enfant.
  ➥</Commentaire>
  <Commentaire temps="15">Il propose les objets qu'il tient à son père qu'il a
  ➥reconnu.</Commentaire>
  <Commentaire temps="17.78">L'enfant se lasse très vite d'un jeu.</Commentaire>
</Commentaires>
```

Explications

Nous commençons par charger un fichier XML qui contient les commentaires qui vont se succéder sur la scène ainsi que les temps des points de repères.

Lorsque le chargement est terminé, nous ajoutons les repères à la vidéo avec la méthode `addASCuePoint()` puis nous lançons la lecture de la vidéo.

Un écouteur associé à l'événement `cuePoint` met automatiquement la vidéo en pause lorsque la tête de lecture rencontre un repère. Il lance également la fonction `setInterval()` qui est chargée de temporiser la reprise de la vidéo au bout de cinq secondes (5 000 millisecondes).

Il est très important d'exécuter la commande `clearInterval()` dans la fonction appelée pour arrêter le cycle qui aurait lieu dans le cas contraire toutes les cinq secondes.

Cette animation est très intéressante car elle propose une lecture de vidéo sans que l'utilisateur n'ait paradoxalement besoin d'intervenir.

Pause avec reprise de lecture par un bouton

L'intérêt de cette animation est de démontrer qu'il est très simple de désactiver un bouton durant la lecture d'une vidéo. Ce dernier redevient actif dès que la vidéo entre en pause suite à la détection d'un repère.

Animation : KVideoInterrompuePlan/video_interrompue.fla

Vidéo : KVideoInterrompuePlan/Tom.flv

XML : KVideoInterrompuePlan/commentaires.xml

```
var commentaires:XML = new XML();
commentaires.load("commentaires.xml");
commentaires.ignoreWhite = true;
//
```

```
btSuite.enabled = false;
btSuite._alpha = 10;
//
commentaires.onLoad = function() {
  nombreNoeuds = this.firstChild.childNodes.length;
  coursier = new Object();
  for (i=0; i<nombreNoeuds; i++) {
         coursier.time = Number(this.firstChild.childNodes[i].attributes.temps);
         coursier.name = i;
         ecranVideo.addASCuePoint(coursier);
  }
  ecranVideo.play();
};
//
surveilCues = new Object();
surveilCues.cuePoint = function(infoRenvoyee) {
  numeroRepere = Number(infoRenvoyee.info.name);
  vCommentaire = commentaires.firstChild.childNodes[numeroRepere].firstChild;
  ecranVideo.pause();
  btSuite.enabled = true;
  btSuite._alpha = 100;
};
ecranVideo.addEventListener("cuePoint", surveilCues);
//
btSuite.onPress = function() {
  vCommentaire = "";
  ecranVideo.play();
  btSuite.enabled = false;
  btSuite._alpha = 10;
};
```

Fichier XML :

Identique à l'animation précédente.

Explications

Commençons par étudier ce qui change entre le script de cette animation et celui de l'animation précédente.

L'exemple de la vidéo qui se met en pause automatiquement et se relance après cinq secondes contient les deux lignes d'instructions suivantes (il s'agit du cœur du script) :

```
ecranVideo.pause();
relancerVideo = setInterval(pauseVideo, 5000);
```

Dans l'animation que nous sommes en train d'analyser, les deux lignes ci-avant ont été remplacées par celles-ci :

```
btSuite.enabled = false;
btSuite._alpha = 10;
```

…

```
    ecranVideo.pause();
    btSuite.enabled = true;
    btSuite._alpha = 100;
```

...

```
btSuite.onPress = function() {
    vCommentaire = "";
    ecranVideo.play();
    btSuite.enabled = false;
    btSuite._alpha = 10;
};
```

Au lancement de l'animation le bouton qui permet de relancer la vidéo après une pause est rendu inactif (non cliquable) afin qu'on ne puisse pas l'utiliser. Pour renforcer visuellement ce fait, sa transparence est réglée à 10.

Lorsqu'un point de repère est décelé, la vidéo se met en pause, mais le bouton est rendu cliquable et redevient opaque.

Le gestionnaire qui permet de relancer la vidéo vide non seulement la zone de commentaires, mais il rend à nouveau le bouton inactif.

Le cœur de ce script fait donc appel à des lignes d'instructions relativement simples, mais elles sont très efficaces.

4

Rendre une séquence interactive

Nous avons souhaité consacrer un chapitre à certaines techniques d'interactivité avec une séquence afin de démontrer qu'une vidéo ne se limite pas à une simple consultation. Nous allons donc découvrir des animations qui présentent les particularités suivantes :

- zones cliquables sur les images d'une vidéo ;
- mémorisation des instants de vidéo sous forme de bookmarks ;
- indexation d'un sommaire en cours de lecture ;
- accélération d'une vidéo ;
- utilisation du clavier pour le contrôle de la vidéo.

Il est vrai que les techniques employées dans ces animations peuvent paraître évidentes, mais elles ne le sont pas pour les utilisateurs débutants en Flash. Mettons-les en avant et commentons-les.

Séquence avec un sommaire

Nous allons vous présenter deux animations qui vous démontrerons qu'une vidéo peut annoncer d'elle-même une progression dans son déroulement (une présentation). Ces méthodes de projection de séquence sont intéressantes lorsque vous présentez des vidéos sous forme d'exposé.

Curseur indexant des titres

Dans cette première animation, nous avons placé un symbole sur la scène, dont le nom d'occurrence est `repereVisuel`. Ce dernier va venir se placer face à des lignes de texte sur la scène qui font référence à des chapitres, des instants clés, etc.

Figure 4-1

Le curseur se place face aux lignes de texte au cours de la lecture de la vidéo.

Animation : KVideoInterrompue/ video_plan_enumeration.fla

Vidéo : KVideoInterrompue /Tom.flv

```
var repere:Number = 0;
//
var commentaires:XML = new XML();
commentaires.load("commentairesListe.xml");
commentaires.ignoreWhite = true;
//
commentaires.onLoad = function() {
  nombreNoeuds = this.firstChild.childNodes.length;
  coursier = new Object();
  for (i=0; i<nombreNoeuds; i++) {
        coursier.time = Number(this.firstChild.childNodes[i].attributes.temps);
        coursier.name = i;
        ecranVideo.addASCuePoint(coursier);
        _root.createTextField("texte"+i, i, 425, 60+(i*20), 250, 50);
        _root["texte"+i].html = true;
        _root["texte"+i].htmlText = commentaires.firstChild.childNodes[i].firstChild;
        //
        style1 = new TextFormat();
```

```
        style1.font = "Arial";
        style1.size = 12;
        _root["texte"+i].setTextFormat(style1);
  }
  ecranVideo.play();
};
//
surveilCues = new Object();
surveilCues.cuePoint = function() {
  repereVisuel._y = _root["texte"+repere]._y;
  repere++;
};
ecranVideo.addEventListener("cuePoint", surveilCues);
```

Fichier XML :

```
<Commentaires>
  <Commentaire temps="1.35">Les formes de couleurs différentes.</Commentaire>
  <Commentaire temps="3.08">L'enfant autonome.</Commentaire>
  <Commentaire temps="7.60">L'enfant et ses deux mains.</Commentaire>
  <Commentaire temps="9.96">L'ouïe de l'enfant</Commentaire>
  <Commentaire temps="12.32">L'enfant sait reconnaître les sons.</Commentaire>
  <Commentaire temps="15">La notion de partage.</Commentaire>
  <Commentaire temps="17.78">L'enfant se lasse très vite d'un jeu.</Commentaire>
</Commentaires>
```

Explications

Dans un fichier XML, nous avons saisi le contenu de chaque ligne de texte qui va être placée sur la scène (figure 4-1) ; nous déclarons également un attribut qui va servir de timecode pour la pose du repère.

Puisque nous devons effectuer deux actions avec le fichier XML, à savoir lire les données de chaque nœud, puis exploiter les attributs associés, nous utilisons une boucle `for()`.

Nous parcourons donc l'arborescence de ce fichier au moyen de cette boucle pour utiliser la valeur de l'attribut de chaque nœud afin de créer des repères dynamiquement, à l'aide de la méthode `addASCuePoint()`.

> **Remarque**
> Nous n'avons pas besoin de spécifier de nom de repère particulier, c'est pourquoi nous utilisons la valeur de `i` en guise d'étiquette (`coursier.name=i`).

Vous noterez que le lancement de la vidéo ne se fait qu'à partir du moment où les lignes de texte et les repères dans la vidéo ont été créés.

L'écouteur que nous créons utilise l'événement `cuePoint` pour pouvoir placer l'occurrence `repereVisuel` à des nouvelles coordonnées `_y` à chaque repère rencontré.

> **Remarque**
>
> Pour augmenter ou diminuer l'espace entre les lignes, vous devez changer la valeur `20` contenue dans la ligne qui exécute la méthode `createTextField()`.

Suivre la progression de la lecture d'une vidéo

Dans certains cas, le curseur de progression qui indique le temps écoulé depuis le début de la lecture (figure 4-2) peut être inadapté. Nous avons conçu une animation dans laquelle la visualisation de la progression de la lecture se fait sous la forme d'éléments graphiques qui changent de taille. Lors d'un premier essai, nous avions utilisé la propriété `blendMode` ; autant dire que les deux lignes de la boucle `for()` situées à la fin du script, peuvent être remplacées par n'importe quelles autres instructions.

Figure 4-2

Le triangle blanc situé dans la barre du contrôleur indique non seulement la positon de la tête de lecture, mais également le temps écoulé proportionnellement à la durée totale de la vidéo.

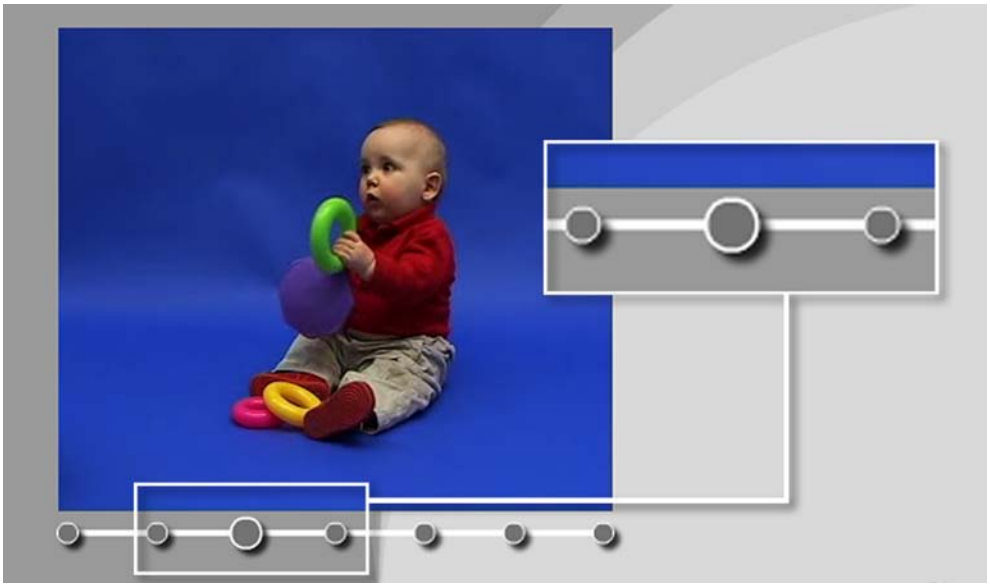


Figure 4-3

Chaque cercle grossit à tour de rôle au cours de la lecture de la vidéo.

Animation : KVideoInterrompue/ video_plan_metro.fla

Vidéo : KVideoInterrompue /Tom.flv

```
var repere:Number = 0;
//
var commentaires:XML = new XML();
commentaires.load("commentaires.xml");
commentaires.ignoreWhite = true;
//
commentaires.onLoad = function() {
  nombreNoeuds = this.firstChild.childNodes.length;
  coursier = new Object();
  for (i=0; i<nombreNoeuds; i++) {
        coursier.time = Number(this.firstChild.childNodes[i].attributes.temps);
        coursier.name = i;
        ecranVideo.addASCuePoint(coursier);
  }
  ecranVideo.play();
};
//
surveilCues = new Object();
surveilCues.cuePoint = function(infoRenvoyee) {
  repere++;
  changerTaille(repere);
};
ecranVideo.addEventListener("cuePoint", surveilCues);
//
//
function changerTaille(numeroOccurrences) {
  for (i=1; i<=7; i++) {
        _root["t"+i]._xscale = _root["t"+i]._yscale=100;
  }
  _root["t"+numeroOccurrences]._xscale = _root["t"+numeroOccurrences]._yscale=150;
}
```

Explications

Nous vous invitons à consulter les explications de l'animation précédente qui sont identiques à celles-ci, à une différence près : nous ne changeons pas la position d'une occurrence à chaque fois qu'un repère est rencontré par la tête de lecture, mais nous exécutons une fonction qui est chargée de changer la taille des occurrences en forme de cercle au bas de la scène. Par ailleurs, la première boucle `for()` se limite à l'ajout de points de repères dans la vidéo, mais ne crée pas de texte sur la scène.

Si vous souhaitiez rendre cliquables les textes de la première animation ou les cercles de ce dernier exemple, il vous suffirait d'utiliser la méthode `seek()` pour déplacer la tête de lecture.

Ajouter des zones cliquables

Animation : KVideoInteractives/ video_construire_zonesclic.fla

Vidéo : KVideoInteractives /lionel.flv

Dans la copie d'écran de la figure 4-4, un rectangle a été positionné dynamiquement sur l'image au cours de la lecture de la vidéo. L'utilisateur peut ainsi cliquer dessus pour déclencher une action. Cela pourrait paraître très simple de développer une telle animation, mais il est tout de même difficile de concevoir le fichier XML qui va nous aider à gérer les points suivants :

- la taille et la position de la zone cliquable ;
- l'instant à partir duquel la zone doit apparaître et disparaître sur l'image de la vidéo ;
- l'action à associer à une zone cliquable.

Figure 4-4

La zone cliquable (laissée visible pour les besoins de la copie d'écran) est positionnée sur la vidéo (sur la scène) grâce à un événement cuePoint et à des informations contenues dans un fichier XML.

> **Remarque**
>
> Avant de poursuivre la lecture des pages qui vont suivre, nous vous conseillons de consulter l'animation video zonesclic.swf (dossier KVideoInteractives) afin de mieux comprendre notre objectif.

Afin de créer le fichier XML dont nous avons besoin pour réaliser une animation avec des zones cliquables, nous avons développé une application qui présente l'interface de la figure 4-5.

Figure 4-5

Cette interface vous aide à créer la liste qui va servir à générer le fichier XML nécessaire à la réalisation de l'animation finale.

Les informations situées en bas à gauche de l'écran vous indiquent comment procéder pour générer une liste de plages temporelles durant lesquelles un rectangle va être placé à un endroit précis sur l'image, à une échelle bien définie. Lorsque vous aurez déterminé toutes les périodes dans la liste qui se trouve à droite de la vidéo, il ne vous restera plus qu'à les copier-coller dans un fichier XML. Le bouton Générer le XML permet de placer dans le Presse-papiers de votre ordinateur et dans la palette Sortie de Flash, le contenu du fichier XML que vous devez générer pour l'utiliser avec l'animation suivante. Avant de la découvrir, voici le script de cette première application.

```
var debutZoneClic:Number = 0;
var fintZoneClic:Number = 0;
var x_ZoneClic:Number = 0;
var y_ZoneClic:Number = 0;
var largeur_ZoneClic:Number = 0;
var hauteur_ZoneClic:Number = 0;
//
ecranVideo.contentPath = "lionel.flv";
ecranVideo.autoPlay = false;
ecranVideo.volume = 20;
//
cadreClics.onPress = function() {
  this.startDrag();
};
```

```
cadreClics.onRelease = cadreclics.onReleaseOutside=function () {
  stopDrag();
};
//
surveil = new Object();
surveil.onKeyDown = function() {
  laTouche = Key.getCode();
  switch (laTouche) {
  case 32 :
        activerDesactiverPause();
        break;
  case 37 :
        valeurX = -2;
        break;
  case 39 :
        valeurX = 2;
        break;
  case 38 :
        valeurY = -2;
        break;
  case 40 :
        valeurY = 2;
        break;
  }
  _root.onEnterFrame = function() {
        if (cadreClics._xscale>=3) {
                cadreClics._xscale += valeurX;
        } else {
                cadreClics._xscale = 3;
        }
        if (cadreClics._yscale>=3) {
                cadreClics._yscale += valeurY;
        } else {
                cadreClics._yscale = 3;
        }
  };
  surveil.onKeyUp = function() {
        valeurX = 0;
        valeurY = 0;
        delete _root.onEnterFrame;
  };
};
Key.addListener(surveil);
//
debut_cmp.onPress = function() {
  debutZoneClic = ecranVideo.playheadTime;
};
zonesClics = [];
fin_cmp.onPress = function() {
  finZoneClic = ecranVideo.playheadTime;
  temps_cmp.addItem(debutZoneClic+" - "+finZoneClic);
```

```
  decalageVertical = ecranVideo._y;
  decalageHorizontal = ecranVideo._x;
  zonesClics.push({coordX:cadreClics._x-decalageHorizontal, coordY:cadreClics.
  ➥_y-decalageVertical, largeur:cadreClics._width, hauteur:cadreClics._height,
  ➥debutTimer:debutZoneClic, finTimer:finZoneClic});
};
//
lectureEnCours = new Object();
lectureEnCours.playheadUpdate = function() {
  teteLectureVideo = ecranVideo.playheadTime;
};
ecranVideo.addEventListener("playheadUpdate", lectureEnCours);
ecranVideo.playheadUpdateInterval = 50;
//
btEffacerEntree.onPress = function() {
  temps_cmp.removeItemAt(temps_cmp.selectedIndex);
  zonesClics.splice(ligneCliquee, 1);
};
btEffacerListe.onPress = function() {
  temps_cmp.removeAll();
  zonesClics = [];
};
//
surveil.change = function(entree) {
  ligneCliquee = entree.target.selectedIndex;
  recuperation = zonesClics[ligneCliquee];
  vDebutSelection = recuperation.debutTimer;
  vFinSelection = recuperation.finTimer;
  ecranVideo.seek(vDebutSelection);
  cadreClics._x = zonesClics[ligneCliquee].coordX;
  cadreClics._y = zonesClics[ligneCliquee].coordY;
  cadreClics._width = zonesClics[ligneCliquee].largeur;
  cadreClics._height = zonesClics[ligneCliquee].hauteur;
};
temps_cmp.addEventListener("change", surveil);
//
debutSelection_inst.onSetFocus = debutSelection_inst.onSetFocus=function () {
  ecranVideo.seek(vDebutSelection);
};
btModifier.onPress = function() {
  zonesClics[ligneCliquee].debutTimer = vDebutSelection;
  zonesClics[ligneCliquee].finTimer = vFinSelection;
  zonesClics[ligneCliquee].largeur = cadreClics._width;
  zonesClics[ligneCliquee].hauteur = cadreClics._height;
  zonesClics[ligneCliquee].coordX = cadreClics._x;
  zonesClics[ligneCliquee].coordY = cadreClics._y;
  temps_cmp.replaceItemAt(ligneCliquee, {label:vDebutSelection+" - "+vFinSelection});
};
//
```

```
btGenererXML.onPress = function() {
  pressePapierVirtuel = new Object();
  pressePapierVirtuel.text = "<zonesClics>"+newline;
  for (i=0; i<zonesClics.length; i++) {
       pressePapierVirtuel.text += "<zone coordX='"+zonesClics[i].coordX+"' coordY='"
       ➥+zonesClics[i].coordY+"' largeur='"+zonesClics[i].largeur+"' hauteur
       ➥='"+zonesClics[i].hauteur+"' debut='"+zonesClics[i].debutTimer
       ➥+"' fin='"+zonesClics[i].finTimer+"' action='À définir'"+"/>"+newline;
  }
  pressePapierVirtuel.text += "</zonesClics>";
  trace(pressePapierVirtuel.text);
  System.setClipboard(pressePapierVirtuel.text);
};
```

Nous ne commenterons pas ce script qui ne vous apportera rien de plus par rapport à ce que nous cherchons à faire. Précisons simplement les points suivants :

- Six valeurs stockent les informations relatives à la taille (largeur et hauteur) et à la position (x et y) du rectangle, ainsi que les deux temps qui indiquent le début et la fin de la présence du rectangle sur la scène.
- Les nombreux gestionnaires de ce script permettent de gérer le déplacement de la zone cliquable, son agrandissement, les clics sur les différents boutons de l'interface et le remplissage de l'occurrence de type List.

Le script peut paraître assez long et légèrement difficile à comprendre, mais il a été conçu pour répondre à de nombreuses actions.

Passons à présent à l'animation qui nous intéresse, celle qui va nous permettre de définir des zones cliquables dans une vidéo (figure 4-4).

> **Rappel**
>
> La zone cliquable n'est pas obligatoirement visible. Dans notre exemple, nous l'avons rendue légèrement transparente afin que vous puissiez constater ce qui se passe réellement.

Animation : KVideoInteractives/ video_ zonesclic.fla

Vidéo : KVideoInteractives /lionel.flv

Dans cette animation, le concept est très intéressant car il permet de proposer à l'utilisateur une réelle interactivité. Nous avons l'habitude de cliquer sur des liens de type texte ou image, mais plus rarement de type vidéo.

Une simple occurrence a été placée sur la scène ; il s'agit d'un rectangle dont le point d'alignement est situé en haut à gauche. Nous allons pouvoir la rendre cliquable au cours de la lecture de la vidéo.

```
ecranVideo.contentPath = "lionel.flv";
//
var repere:Number = 0;
var racine:XML;
```

```
var decalageHorizontal:Number = ecranVideo._x;
var decalageVertical:Number = ecranVideo._y;
//
var commentaires:XML = new XML();
commentaires.load("zonesclics.xml");
commentaires.ignoreWhite = true;
//
commentaires.onLoad = function() {
  var nombreNoeuds:Number = this.firstChild.childNodes.length;
  racine = this.firstChild;
  coursier = new Object();
  for (i=0; i<nombreNoeuds; i++) {
        coursier.time = Number(racine.childNodes[i].attributes.debut);
        coursier.name = i;
        ecranVideo.addASCuePoint(coursier);
        coursier.time = Number(racine.childNodes[i].attributes.fin);
        coursier.name = "Retrait";
        ecranVideo.addASCuePoint(coursier);
  }
  ecranVideo.play();
};
//
surveilCues = new Object();
surveilCues.cuePoint = function(repere) {
  trace(repere.info.name);
  if (repere.info.name == "Retrait") {
        zoneClic._x = -10000;
  } else {
        numeroNoeud = repere.info.name;
        zoneClic._x = Number(racine.childNodes[numeroNoeud].attributes.coordX)
        ➥+decalageHorizontal;
        zoneClic._y = Number(racine.childNodes[numeroNoeud].attributes.coordY)
        ➥+decalageVertical;
        zoneClic._width = racine.childNodes[numeroNoeud].attributes.largeur;
        zoneClic._height = racine.childNodes[numeroNoeud].attributes.hauteur;
        zoneClic.onPress = function() {
                trace(racine.childNodes[numeroNoeud].attributes.action);
        };
  }
};
ecranVideo.addEventListener("cuePoint", surveilCues);
```

Fichier XML :

```
<zonesClics>
  <zone coordX='107.25' coordY='77' largeur='111.7' hauteur='114.2'
  ➥debut='0' fin='2.313' action='À définir'/>
  <zone coordX='100.25' coordY='19' largeur='158.6' hauteur='210.2'
  ➥debut='5.552' fin='8.66' action='À définir'/>
  <zone coordX='199.25' coordY='103' largeur='91.6' hauteur='129.05'
  ➥debut='8.66' fin='11.273' action='À définir'/>
```

```
    <zone coordX='26.25' coordY='57' largeur='160.85' hauteur='183.7'
    ➥debut='14.956' fin='18.717' action='À définir'/>
</zonesClics>
```

Explications

À la lecture du fichier XML, nous constatons que chaque nœud possède six attributs. Les six valeurs associées à ces derniers vont nous permettre de placer précisément l'occurrence du rectangle sur la vidéo, à une échelle précise et un instant précis. La sixième et dernière valeur va nous servir à effacer le rectangle lorsqu'il ne sert plus.

Nous commençons par charger le fichier XML pour utiliser les valeurs évoquées dans le paragraphe ci-dessus. Lorsque le chargement est effectif (`onLoad`), nous créons des repères dans la vidéo. Vous remarquerez que nous créons à chaque fois, deux cuePoints dont seul le nom d'étiquette change : soit nous utilisons un numéro, soit l'expression « Retrait ».

Remarque

Nous lançons la vidéo uniquement à partir du moment où les cuePoints ont été ajoutés.

Pour détecter à présent le passage de la tête de lecture sur un cuePoint, nous utilisons un écouteur accompagné de l'événement `cuePoint`. Si le nom d'étiquette du repère rencontré est Retrait, nous masquons la zone cliquable en la positionnant en dehors de la scène. En revanche, si le nom correspond à une valeur numérique, nous positionnons la zone cliquable sur la scène, à des dimensions précises.

La seule difficulté de cette animation est de définir l'action à réaliser lorsque l'utilisateur clique sur la zone. Dans notre exemple, nous faisons uniquement appel à la fonction `trace()`. En fonction de vos besoins, vous pourrez utiliser les méthodes et/ou propriétés suivantes :

- `loadMovie()`, `attachMovie()` ;
- `getURL()` ;
- `loadSound()`, `contentPath`.

Remarque

Pour pouvoir utiliser les propriétés et méthodes ci-dessus, il vous faudra remplacer la valeur `'À définir'` du dernier attribut de chaque nœud du document XML par des valeurs adéquates (un nom de fichier de vidéo, de son, de liaison ou une adresse Web).

Bien évidemment, ce ne sont que des suggestions. Nous avons simplement fait référence aux actions les plus logiques (chargement d'un son, d'une vidéo, d'une page Web, d'une image ou d'un symbole). Vous pourriez également effectuer un test pour évaluer la valeur de ce dernier attribut.

Voilà, vous pouvez à présent cliquer sur vos vidéos, à des endroit précis de l'image, et déclencher en conséquence les actions appropriées.

Gérer des bookmarks dans une vidéo

Pour vous apprendre à gérer des bookmarks dans une vidéo, nous avons réalisé deux animations. Nous vous présentons la plus complexe (NavigationReperesAjouts.fla) qui reprend exactement les mêmes lignes d'instructions que la plus simple (NavigationReperesAjouts2.fla). Celle que nous allons aborder contient des options supplémentaires.

Figure 4-6

L'utilisateur peut mettre la vidéo en pause et définir un repère sur l'image arrêtée afin de pouvoir y accéder plus facilement par la suite.

Animation : KVideoReperes/ NavigationReperesAjouts.fla

Vidéo : KVideoReperes /lionel.flv

```
ecranVideo.contentPath = "lionel.flv";
//
var nomsReperes:Array = [];
var tempsReperes:Array = [];
var nbrReperes:Number = 0;
//
//Pour faire disparaître la barre de l'ascenseur
//listeReperes_comp.vScrollPolicy = false;
//
zoneCommentaires.text = "";
//
```

```
btAjoutRepere.onPress = ajoutRepere=function () {
  if (zoneCommentaires.text == "") {
        nomsReperes.push("Repère "+(nbrReperes+1));
  } else {
        nomsReperes.push(zoneCommentaires.text);
  }
  tempsReperes.push(ecranVideo.playheadTime);
  nbrReperes++;
  zoneCommentaires.text = "";
  actualiserListeComposant();
};
//
btSuppressionRepere.onPress = function() {
  listeReperes_comp.removeItemAt(listeReperes_comp.selectedIndex);
};
//
var surveil = new Object();
surveil.change = function(transmission) {
  numeroElementClique = transmission.target.selectedIndex;
  ecranVideo.seek(tempsReperes[numeroElementClique]);
};
listeReperes_comp.addEventListener("change", surveil);
//
actualiserListeComposant = function () {
  listeReperes_comp.removeAll();
  var nbrReperesDansLaListe:Number = nomsReperes.length;
  for (i=0; i<nbrReperesDansLaListe; i++) {
        listeReperes_comp.addItem(nomsReperes[i]+" ("+tempsReperes[i]+")");
  }
};
//
_root.onKeyDown = function() {
  if (Key.isDown(Key.ENTER)) {
        ajoutRepere();
  }
};
Key.addListener(_root);
```

Explications

Nous commençons par créer deux tableaux vides dans lesquels nous enregistrerons les noms et temps de repères (*bookmarks*).

Nous initialisons ensuite le contenu du texte de saisie dont le nom d'instance est `zoneCommentaires`. Il s'agit de la zone dans laquelle l'utilisateur va indiquer un nom de bookmark.

Lorsqu'on clique sur le bouton qui permet justement d'ajouter un bookmark, il faut d'abord s'assurer que le texte de saisie n'est pas vide. Si ce texte est vide, un nom par

défaut sera utilisé. Il s'agit de la concaténation de la chaîne `Repère` et de la valeur de la variable `nbrReperes`. Nous obtiendrons ainsi les noms Repère1, Repère2, etc.

À chaque demande d'ajout d'un nouveau bookmark dans la liste, nous enregistrons une nouvelle entrée dans les listes `nomsReperes` et `tempsReperes`. Nous faisons également appel à la fonction `actualiserListeComposant` pour mettre à jour le contenu de l'occurrence de type List qui se trouve sur la scène.

Remarque
La ligne `zoneCommentaires.text="";` permet de vider la zone de saisie du nom de bookmark afin de faire comprendre à l'utilisateur que l'enregistrement est effectué et qu'il peut en saisir un nouveau.

Un gestionnaire permet également de supprimer une entrée sélectionnée de la liste.

Pour finir, la sélection d'une ligne de la liste présente sur la scène est surveillée par un écouteur associé à l'événement `change`. Ainsi, lorsque l'utilisateur clique sur l'occurrence du composant de type List, la tête de lecture est placée au temps mémorisé dans le tableau `tempsReperes`.

Le dernier gestionnaire sert simplement à valider l'enregistrement d'un nouveau bookmark lorsque l'utilisateur appuie sur la touche Entrée de son clavier. Il aurait été judicieux d'ajouter les quelques lignes supplémentaires ci-après pour éviter que la touche Entrée ajoute un bookmark si le focus n'est pas donné au texte de saisie (l'utilisateur n'est alors pas prêt à saisir un texte pour définir un nom).

```
zoneCommentaires.onSetFocus = function() {
  Key.addListener(_root);
};
zoneCommentaires.onKillFocus = function() {
  Key.removeListener(_root);
};
```

Utilisation du clavier

Avant d'aborder cette technique de contrôle de lecture d'une vidéo, nous attirons votre attention sur le point que nous évoquions en introduction à ce livre. Un certain nombre d'utilisateurs ont l'habitude de déclencher et de mettre une vidéo en pause à l'aide d'une pression sur la barre d'espace. Pour le réglage du volume, l'utilisation des touches fléchées Haut et Bas du clavier est également fréquente.

Bien sûr, le script de l'animation qui va suivre ne représente nullement une référence, et il est d'ailleurs très simple de changer les lignes de code chargées d'exécuter des actions lorsqu'une touche du clavier est enfoncée.

L'animation est très simple car son script se limite à l'exécution de cinq actions, chacune étant associée à une touche du clavier.

Animation : KVideoInteractives/interactiviteClavier.fla

Vidéo : KVideoInteractives /lionel.flv

```
ecranVideo.contentPath = "lionelAvecRepere.flv";
//
surveil = new Object();
surveil.onKeyDown = function() {
  laTouche = Key.getCode();
  //
  switch (laTouche) {
  case 32 :
        if (ecranVideo.paused) {
                ecranVideo.play();
        } else {
                ecranVideo.pause();
        }
        break;
  case 38 :
        ecranVideo.volume += 5;
        vVolume = ecranVideo.volume;
        break;
  case 40 :
        ecranVideo.volume -= 5;
        vVolume = ecranVideo.volume;
        break;
  case 37 :
        ecranVideo.seekToPrevNavCuePoint();
        break;
  case 39 :
        ecranVideo.seekToNextNavCuePoint();
        break;
  }
};
Key.addListener(surveil);
```

Explications

Nous définissons un gestionnaire `onKeyDown` qui va être chargé de surveiller les touches du clavier. Si l'une d'entre elles est enfoncée (`onKeyDown`), son code de touche est mémorisé dans la variable intitulée `laTouche`. Un traitement permet ensuite d'évaluer le code enregistré pour pouvoir exécuter une instruction (`play()`, `pause()`, `volume` et `seekTo…()`).

Accélérer une vidéo

Pour cette animation élémentaire, le développement a été très simple. Quatre gestionnaires ont suffi pour gérer l'action d'accélération arrière ou avant de la vidéo.

Animation : KVideoControleurVolumeAccel/reglageVitesse.fla

Vidéo : KVideoControleurVolumeAccel /lionel.flv

```
ecranVideo.contentPath = "lionel.flv";
//
avance.onPress = function() {
  function avancer() {
        if (ecranVideo.playheadTime<ecranVideo.totalTime-3) {
                ecranVideo.seek(ecranVideo.playheadTime+3);
        }
  }
  lancer = setInterval(avancer, 300);
};
avance.onRelease = avance.onReleaseOutside=function () {
  clearInterval(lancer);
  delete lancer;
};
//
arriere.onPress = function() {
  function reculer() {
        if (ecranVideo.playheadTime>=3) {
                ecranVideo.seek(ecranVideo.playheadTime-3);
        }
  }
  lancer = setInterval(reculer, 300);
};
arriere.onRelease = arriere.onReleaseOutside=function () {
  clearInterval(lancer);
  delete lancer;
};
```

Explications

Rappelons qu'il est presque impossible d'obtenir une accélération de la lecture d'une vidéo car il n'existe pas de méthode dédiée. Il est en effet impossible d'accéder à des images qui ne soient pas des images clés si vous n'utilisez pas Flash Media Server. Nous devons demander à Flash de déplacer la tête de lecture quelques secondes avant ou après la position actuelle. Pour les deux gestionnaires `onPress`, nous exécutons la fonction `setInterval()` pour répéter la demande de déplacement de la tête de lecture.

Lorsque l'utilisateur relâche le bouton de la souris, nous annulons la recherche accélérée avant ou arrière par la fonction `clearInterval()`, en détruisant ainsi le processus lancé avec la fonction `setInterval()`.

> **Remarque**
>
> Le choix des valeurs `3` et `300` est raisonnable. Si vous diminuez ces valeurs, vous risquez de rencontrer des problèmes d'exécution des lignes d'instructions. Tout va dépendre également du nombre d'images clés que possède la vidéo avec laquelle vous utilisez ce script.

5

Intégrer des effets dans une vidéo

Comment traiter graphiquement les pixels d'une vidéo ? Comment animer l'occurrence du FLV Playback que vous avez placée sur la scène ? C'est ce que vous découvrirez au cours de ce chapitre.

Afficher une vidéo au travers d'un masque

La copie d'écran de la figure 5-1 présente l'exemple d'une vidéo visible au travers d'un masque. Il s'agit dans le cas présent, d'une forme semblable à celle d'un écran.

Figure 5-1

L'incrustation d'une vidéo se fait par l'intermédiaire de la technique du masque.

Contrairement à ce que nous évoquions dans l'animation précédente, vous n'allez pas être obligé de placer l'occurrence de votre vidéo dans un clip car la méthode `setMask()` s'applique aux occurrences de type FLV Playback. La ligne d'instruction nécessaire pour la réalisation d'un tel masque est donc tout simplement :

```
ecranVideo.setMask(laformeDuMasque);
```

Précisons que `ecranVideo` est le nom de l'occurrence du FLV Playback alors que `laformeDuMasque` est le nom d'occurrence au travers de laquelle la vidéo va être visible.

Pour que vous découvriez la simplicité de ce code, une animation intitulée videoMasque vous est proposée dans le dossier KVideoDetoureeTraitGraphique.

Vidéo détourée (fond bleu ou vert)

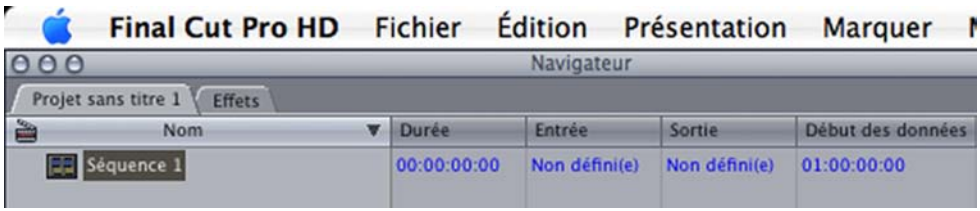
Avant de vous présenter la technique de détourage d'une vidéo tournée sur fond bleu, rappelons qu'il est difficile d'obtenir des contours parfaits, quelle que soit la qualité de votre fichier d'origine. L'éclairage de votre personnage ou de votre objet doit être parfait pour pouvoir obtenir une vidéo exploitable.

> **Conseil**
>
> Si vous devez tourner vous-même une séquence sur fond bleu, ne placez pas votre personnage trop près du fond et éclairez légèrement son dos. Cela facilitera le travail de détourage.

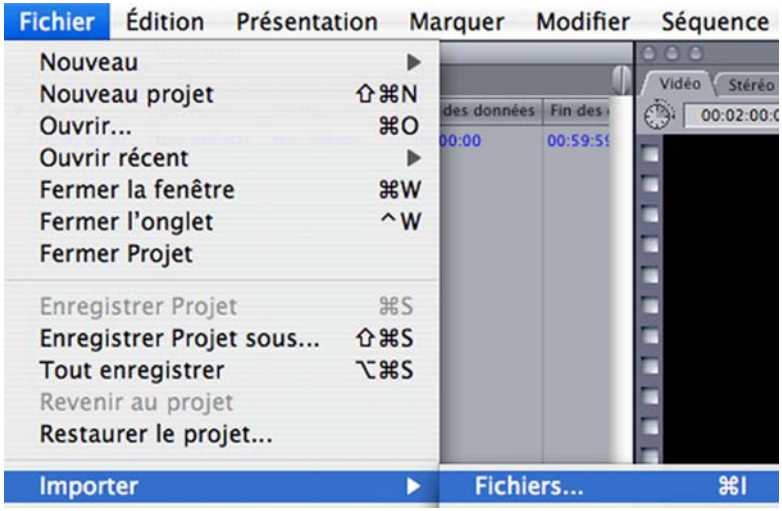
Au niveau de l'ActionScript, il n'y a pas de ligne d'instruction particulière, tout se joue dans les logiciels de montage vidéo. La procédure que nous vous présentons ci-dessous a été réalisée sur Final Cut.

> **Remarque**
>
> N'utilisez pas de HDV pour générer une séquence destinée à un détourage. Le résultat est à ce jour assez médiocre avec l'ensemble des caméscopes disponibles sur le marché.

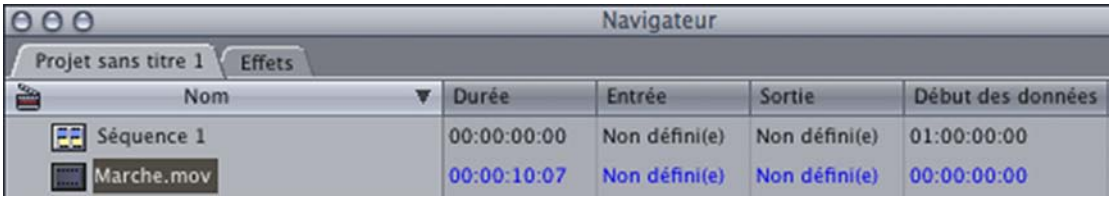
1. Dans Final Cut, commencez tout d'abord par enregistrer votre fichier de travail.



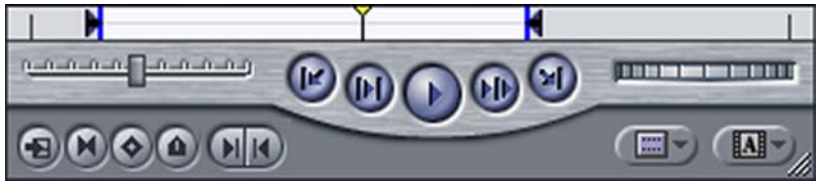
2. Importez un fichier correspondant à une vidéo tournée sur fond bleu ou vert.



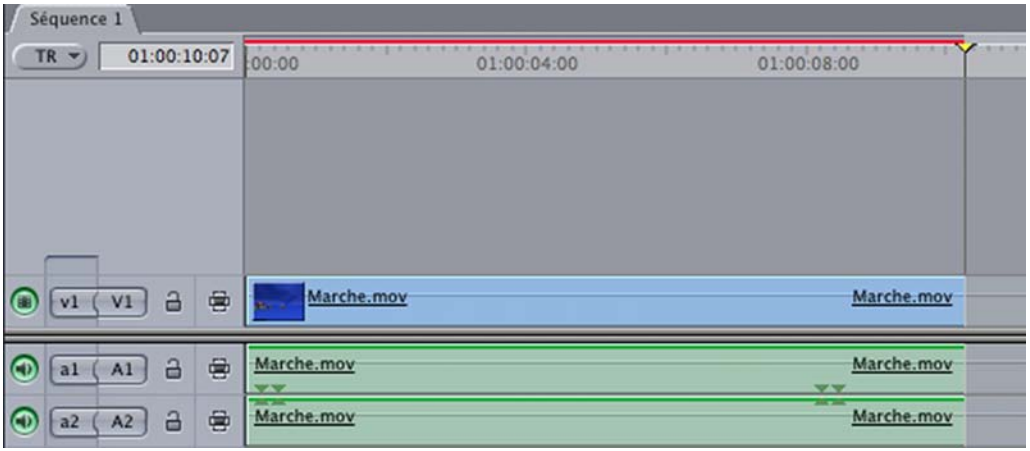
3. Double-cliquez sur l'icône du fichier que vous venez d'importer afin de visualiser les images qu'il contient dans la fenêtre Vidéo.



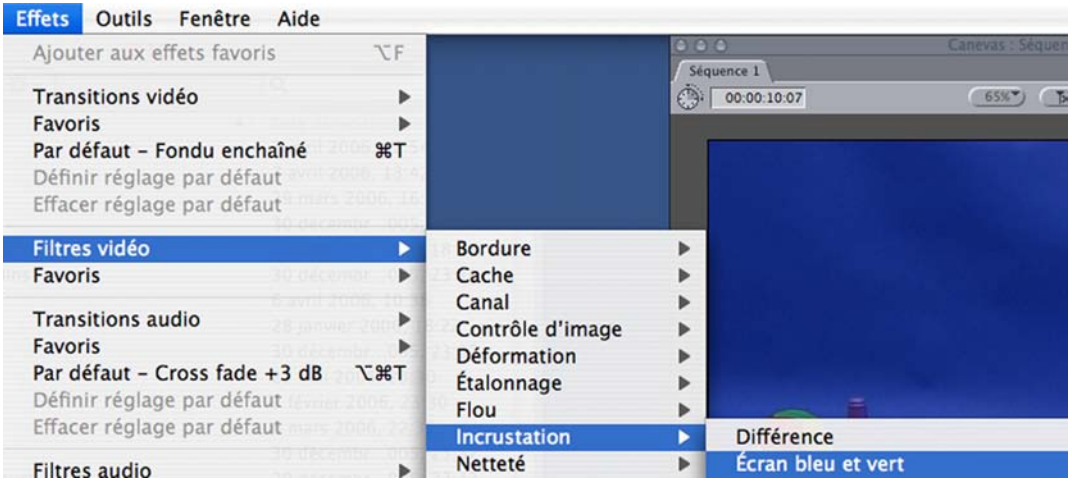
4. En bas de la fenêtre Vidéo, définissez les points d'entrée et de sortie de votre vidéo. En cours de lecture, vous pouvez utiliser les lettres I et O de votre clavier pour définir les points In et Out.



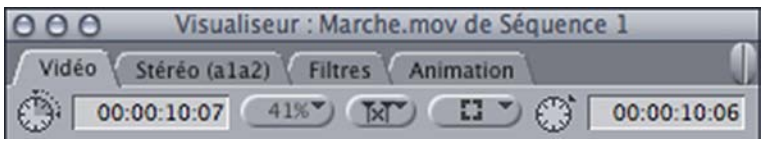
5. Faites glisser votre fichier (par exemple, Marche.mov) dans la fenêtre Séquence 1. Sélectionnez la piste V1 en cliquant dessus tout simplement.



6. Sélectionnez la commande Écran bleu et vert du menu Effets>Filtres vidéo>Incrustation.



7. Pour l'instant, vous ne voyez que la fenêtre Vidéo dans votre interface.

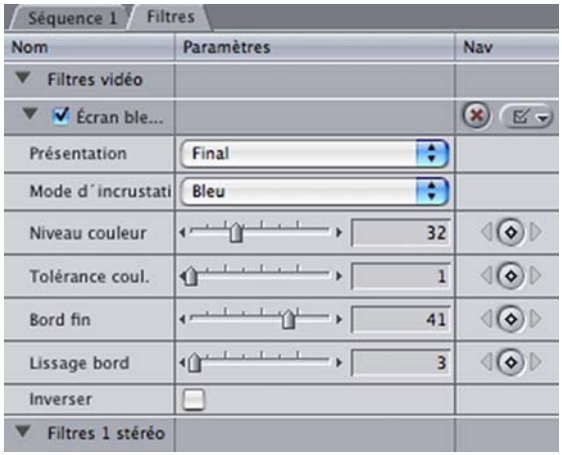


8. Cliquez sur l'onglet Filtres pour afficher une série de réglages.

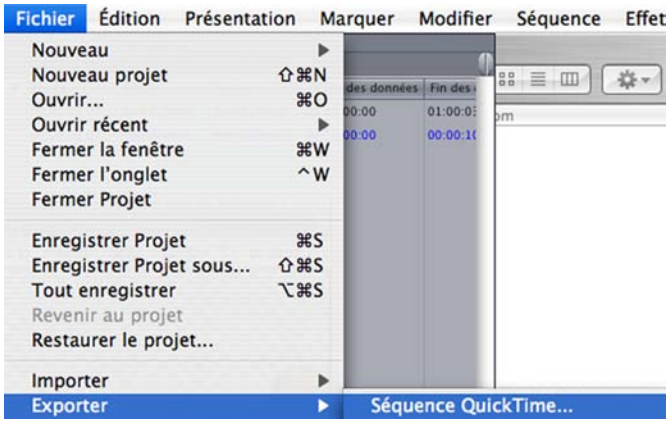
Remarque

Afin de disposer d'une meilleure interface de travail et pour une meilleure visibilité des différentes fenêtres dont vous avez besoin, séparez celles qui s'intitulent Vidéo et Filtres.

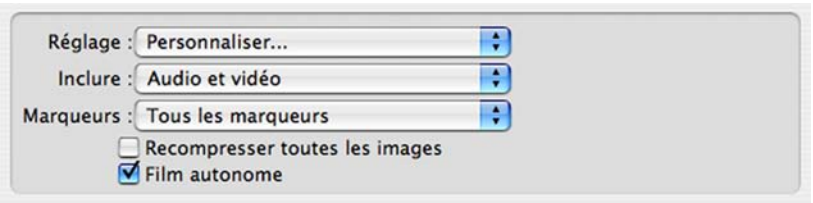
9. Faites varier les quatre curseurs du réglage de la transparence après avoir défini le mode d'incrustation (Bleu ou vert).



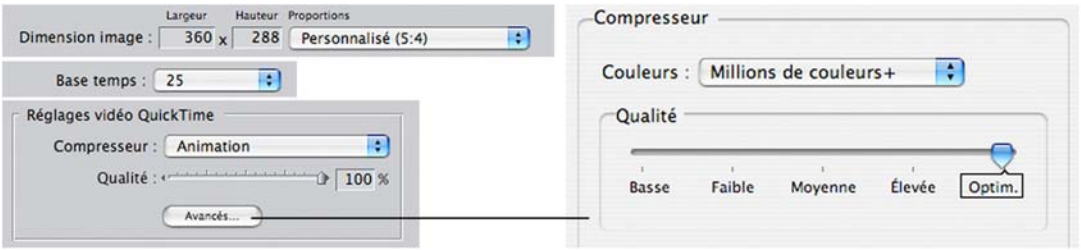
10. Lorsque les réglages sont terminés, exportez votre vidéo en une séquence QuickTime.



11. Dans le champ Réglage, sélectionnez Personnaliser… afin de déterminer votre format d'export.



12. Vous devez impérativement utiliser le compresseur Animation avec l'option Millions de couleurs+ pour pouvoir exploiter la transparence que vous avez définie. Profitez-en pour redimensionner votre vidéo si vous ne l'avez pas encore fait.



13. Votre vidéo peut à présent être encodée ; n'oubliez surtout pas de préciser que la couche alpha doit l'être également.

Comme nous l'évoquions au début de cette procédure, vous n'avez rien à faire de plus ; Flash va utiliser cette vidéo comme n'importe quelle autre. La transparence va être obtenue sans la moindre manipulation dans l'interface de Flash, ni même la moindre saisie d'une ou plusieurs lignes d'instructions.

Vidéo avec l'option Mélange

Dans la palette Propriétés de Flash, vous avez sûrement remarqué la présence d'un menu déroulant intitulé Mélange. Dans d'autres logiciels tels que Photoshop, c'est ce qu'on appelle le mode de fusion. Nous n'allons pas vous présenter une animation dans laquelle nous avons rédigé un script, car une seule ligne d'instruction suffit pour traiter graphiquement une vidéo :

```
rayon.blendMode = 10;
```

Le mot `rayon` est le nom d'une occurrence qui se trouve sur la scène et plus précisément sur la vidéo. Si vous changez la valeur de sa propriété `blendMode`, vous observerez un changement de couleur de votre vidéo (uniquement à l'emplacement de l'occurrence intitulée `rayon`) comme le montre la figure 5-2.

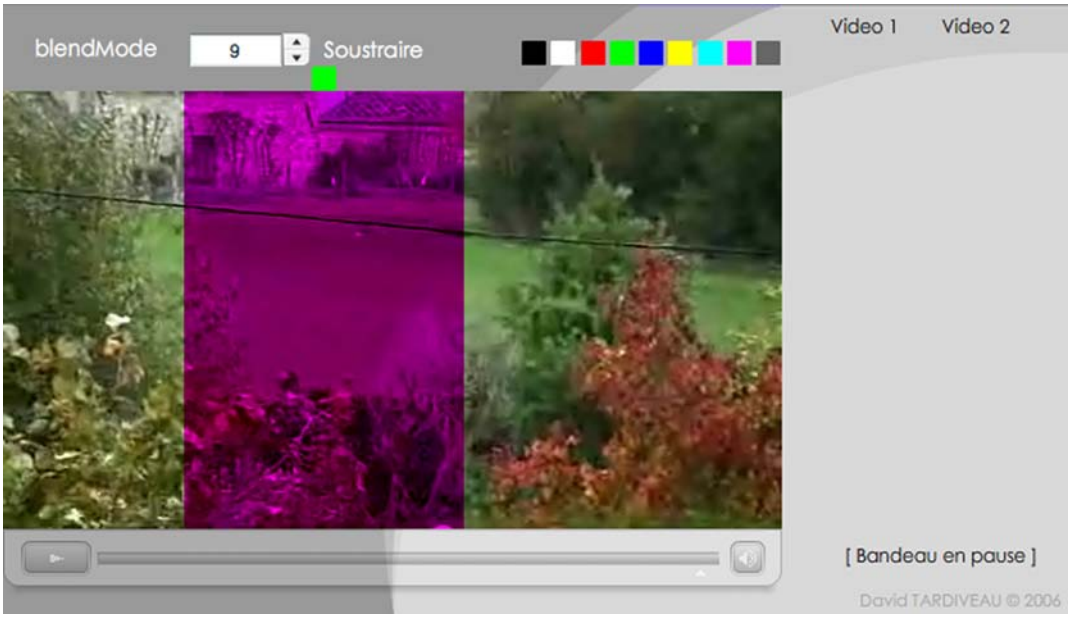


Figure 5-2

Cette animation vous permet de découvrir la réaction colorimétrique de votre vidéo en fonction du mode de fusion appliqué à une occurrence qui se trouve au-dessus d'elle.

À présent, si vous souhaitez appliquer un mode de fusion (un mélange) à une vidéo, vous devez d'abord encapsuler l'occurrence de votre FLV Playback dans un clip. Vous obtenez alors une occurrence dans laquelle se trouve votre vidéo.

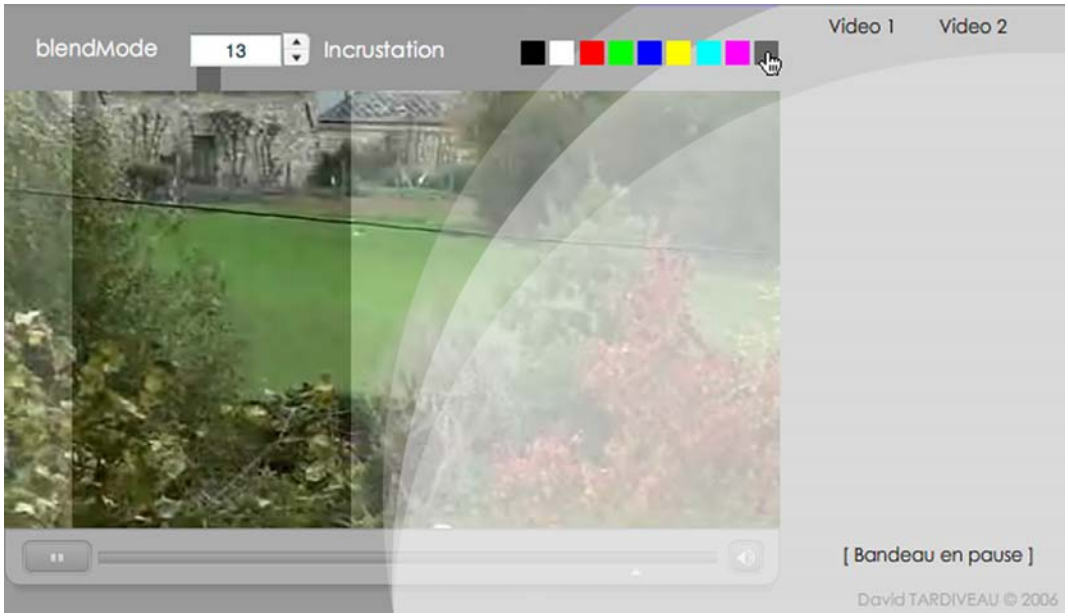


Figure 5-3

Cette animation vous permet de découvrir la réaction colorimétrique de votre vidéo en fonction du mode de fusion appliqué à l'occurrence qui la contient.

Dans l'exemple de l'animation videoEncre2.fla qui se trouve dans le dossier KVideo-DetoureeTraitGraphique, nous avons placé une vidéo dans un clip. Les lignes d'instructions doivent donc tenir compte du chemin de l'occurrence du FLV Playback.

Exemples

```
cadreVideo.ecranVideo.contentPath = "planLarge.flv";
```

ou encore

```
surveil = new Object();
surveil.complete = function() {
  cadreVideo.ecranVideo.play();
};
cadreVideo.ecranVideo.addEventListener("complete", surveil);
```

`cadreVideo` est le nom d'une occurrence de type Clip qui contient une autre occurrence intitulée `ecranVideo` de type FLV Playback.

Nous vous invitons à tester les deux animations qui s'intitulent videoEncre.swf et videoEncre2.swf. Vous verrez ainsi que vous pouvez vraiment appliquer de nombreux traitements graphiques à une vidéo. Vous constaterez également qu'il n'est pas nécessaire de passer par un logiciel de traitement vidéo pour obtenir certains effets, ni de programmer la moindre ligne de code.

Tableau 5-1 La propriété blendMode s'utilise avec des valeurs qui correspondent à des modes de fusion.

Numéro de valeur pour la propriété blendMode	Correspondance
1	Normal
2	Calque
3	Produit
4	Écran
5	Éclaircir
6	Obscurcir
7	Différence
8	Ajouter
9	Soustraire
10	Inverser
11	Alpha
12	Effacer
13	Incrustation
14	Lumière crue

Animation de l'occurrence FLVPlayback

Avant de lire les derniers paragraphes de ce chapitre, nous vous invitons à consulter les animations au format SWF du dossier KVideosAnimées. Vous découvrirez que le déroulement du scénario de chaque vidéo entraîne une action sur la scène de Flash. Il s'agit de simples synchronisations, mais nous agissons sur l'occurrence du FLV Playback, plutôt que de manipuler d'autres objets et/ou textes sur la scène. Pour certaines vidéos, nous prolongeons également l'action qui démarre dans la vidéo et se termine sur la scène.

Toutes les vidéos des animations qui vont suivre ont été scénarisées avant tournage, car l'improvisation est impossible. À chaque plan correspond un script précis.

Dans tous les scripts de ces six animations, un écouteur surveille, à l'aide de l'événement `cuePoint`, le passage de la tête de lecture sur un repère (cuePoint). Lorsque cela se produit, différentes actions sont déclenchées en fonction de la finalité de l'animation.

Le personnage qui fait glisser la vidéo sur la scène

Figure 5-4

Le personnage simule le fait qu'il se trouve sur un plancher qu'il essaye de faire glisser en donnant un à-coup au sol. L'occurrence de la vidéo se met alors à glisser de gauche à droite de la scène.

Animation : KVideoAnimées/video_glisse.fla

Vidéo : KVideoAnimées /SautPourVideoGlisse.flv

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
//
ecranVideo.load("SautPourVideoGlisse.flv");
ecranVideo.play();
//
surveilCues = new Object();
surveilCues.cuePoint = function() {
  maTransition = new Tween(ecranVideo, '_x', Regular.easeOut, ecranVideo._x,
  ➥ecranVideo._x+170, 2, true);
};
ecranVideo.addEventListener("cuePoint", surveilCues);
```

Explications

Lorsque l'unique point de repère contenu dans la vidéo est atteint par la tête de lecture, un `Tween` permet de faire glisser la vidéo 170 pixels plus loin que sa place d'origine. L'interpolation dure 2 secondes.

Le personnage qui déclenche une interpolation de mouvement sur la scène

Figure 5-5

Le personnage est arrivé avec les mains fermées au centre de la vidéo. Il a ensuite ouvert les mains pour libérer un papillon qui a pris son envol.

Animation : KVideoAnimées/ video_papillonLibere.fla

Vidéo : KVideoAnimées /PapillonLibere.flv

```
stop();
//
ecranVideo.load("PapillonLibere.flv");
ecranVideo.play();
//
surveilCues = new Object();
surveilCues.cuePoint = function() {
  _root.play();
};
ecranVideo.addEventListener("cuePoint", surveilCues);
```

Explications

Lorsque l'unique point de repère contenu dans la vidéo est atteint par la tête de lecture, celle du scénario de l'animation se met à défiler car une interpolation permet d'animer le déplacement de l'occurrence qui représente un papillon.

Le personnage qui perd son équilibre lorsque la vidéo se met en mouvement

Figure 5-6

Le personnage arrive tranquillement au centre de la vidéo, puis il perd l'équilibre lorsque celle-ci se met à glisser de la droite vers la gauche de la scène.

Animation : KVideoAnimées/viddeo_tapisRoulant.fla

Vidéo : KVideoAnimées /TapisRoulantDepart.flv

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
//
ecranVideo.load("TapisRoulantDepart.flv");
ecranVideo.play();
//
surveilCues = new Object();
surveilCues.cuePoint = function() {
   maTransition = new Tween(ecranVideo, '_x', Regular.easeOut, ecranVideo._x,
   ➥ecranVideo._x-340, 1.5, true);
};
ecranVideo.addEventListener("cuePoint", surveilCues);
```

Explications

Quelle différence faites-vous entre le script de cette animation et celui de la première (le personnage qui fait glisser la vidéo) ? Il n'y en a pas. Seul le scénario de la vidéo change, et nous obtenons alors un effet complètement différent.

Dans cette vidéo, nous avons l'impression que c'est le mouvement de la vidéo qui fait bouger le personnage alors que dans la première vidéo, l'impression est contraire : c'est le personnage qui réussi à faire bouger la vidéo.

Le personnage qui se fait bousculer par un texte de la scène

Figure 5-7

Le personnage se tient tranquillement debout au centre de la vidéo, quand soudain, un texte arrive de la gauche de la scène pour aller frapper les fesses du personnage qui perd alors l'équilibre.

Animation : KVideoAnimées/video_textePousse.fla

Vidéo : KVideoAnimées /TextePoussePerso.flv

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
//
ecranVideo.load("TextePoussePerso.flv");
ecranVideo.play();
//
```

```
surveilCues = new Object();
surveilCues.cuePoint = function() {
  maTransition = new Tween(texteQuiPousse, '_x', Regular.easeIn,
  ➥texteQuiPousse._x, 265, 1, true);
};
ecranVideo.addEventListener("cuePoint", surveilCues);
```

Explications

Cette animation est différente. Deux secondes avant que le personnage ne commence à tomber, nous avons placé un cuePoint. Lorsque la tête de lecture de la vidéo le rencontre, nous exécutons alors un `Tween` qui dure 2 secondes. L'impression est donnée : c'est le texte qui réussi à bousculer le personnage.

Le personnage qui fait glisser la vidéo sur la scène (verticalement)

Figure 5-8

Le personnage saute et fait glisser la vidéo vers le bas de la scène lorsqu'il retombe.

Animation : KVideoAnimées/ video_tombe.fla

Vidéo : KVideoAnimées / SautPourVideoTombe.flv

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
```

```
//
ecranVideo.load("SautPourVideoTombe.flv");
ecranVideo.play();
ecranVideo.autoRewind = false;
//
surveilCues = new Object();
surveilCues.cuePoint = function(reception) {
  if (reception.info.name == 1) {
        maTransition = new Tween(ecranVideo, '_y', Regular.easeOut, ecranVideo._y,
        ➥ecranVideo._y+100, 2, true);
  }
};
ecranVideo.addEventListener("cuePoint", surveilCues);
```

Explications

De façon analogue aux deux vidéos qui se mettent en mouvement (le personnage qui perd son équilibre et celui qui fait glisser la vidéo), un cuePoint déclenche un `Tween`.

Le personnage qui lance des occurrences en dehors de la vidéo

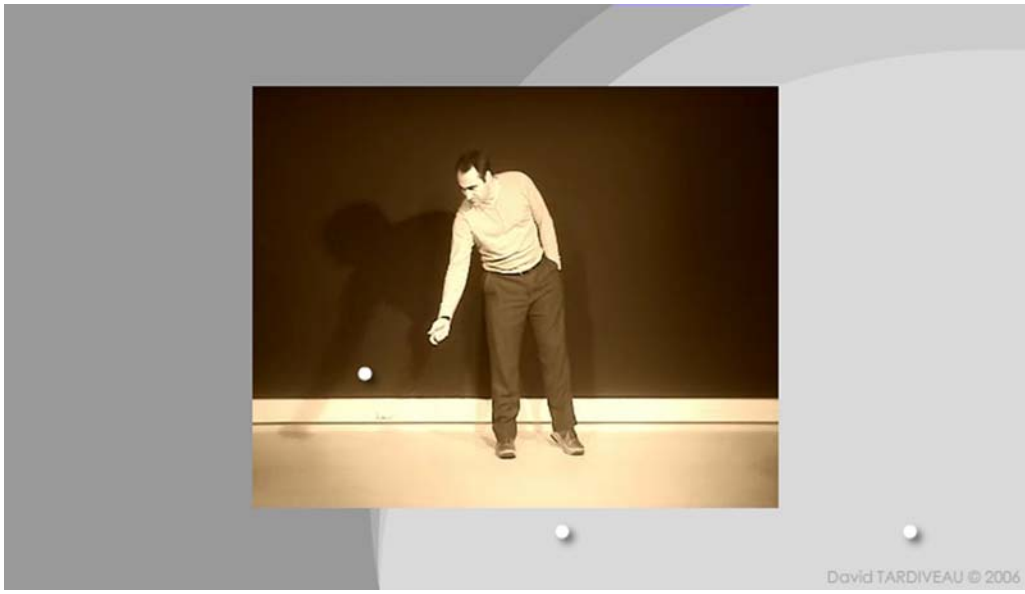


Figure 5-9

Le personnage lance des boules qui sortent de la vidéo.

Animation : KVideoAnimées/ video_lancer3boules.fla

Vidéo : KVideoAnimées / Lancer3boules.flv

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
ecranVideo.load("Lancer3boules.flv");
ecranVideo.play();
//
var placesXPuces:Array = [400, 375, 294];
var placesYPuces:Array = [215, 236, 230];
var destinationsXPuces:Array = [620, 382, 30];
var departX:Number = 0;
var departY:Number = 0;
var indexPuceAplacer:Number = 0;
//
//
surveilCues = new Object();
surveilCues.cuePoint = function() {
  departX = placesXPuces[indexPuceAplacer];
  departY = placesYPuces[indexPuceAplacer];
  _root.attachMovie("puce", "puce"+indexPuceAplacer, indexPuceAplacer,
  ➥{_x:departX, _y:departY});
  _root["maTransitionX"+indexPuceAplacer] = new Tween(_root["puce"+indexPuceAplacer],
  ➥'_x', Bounce.easeOut, departX, destinationsXPuces[indexPuceAplacer], 1, true);
  _root["maTransitionY"+indexPuceAplacer] = new Tween(_root["puce"+indexPuceAplacer],
  ➥'_y', Bounce.easeOut, departY, 360, 1, true);
  indexPuceAplacer++;
};
ecranVideo.addEventListener("cuePoint", surveilCues);
```

Explications

Cette animation présente un script un peu plus long car il exécute plusieurs actions. Essayez de mettre toutes les lignes d'instructions de cette animation en commentaires à l'exception des cinq premières et relancez l'animation. Vous constaterez que rien ne sort de la main du personnage.

À chaque fois que le personnage donne l'impression de jeter quelque chose sur la scène, des cuePoints, intégrés à la vidéo, déclenchent le placement d'un symbole sur la scène à l'aide de la méthode `attachMovie()`.

Un `Tween` se charge ensuite d'animer le déplacement de ces occurrences qui ressemblent à des balles. Nous utilisons une variable intitulée `indexPuceAplacer` et deux tableaux pour gérer le nombre d'occurrences à afficher sur la scène et définir les emplacements `x` et `y` des positions que doivent prendre les balles.

6

Diffuser une vidéo en streaming ou en direct

Avant d'aborder les différentes parties de ce chapitre, rappelons brièvement les deux modes de consultation d'une vidéo en ligne. Vous devez faire la distinction entre le streaming et le téléchargement progressif.

Les séquences que vous placez dans une animation en utilisant des scripts comme ceux présentés dans les cinq premiers chapitres de ce livre sont des vidéos dont la consultation se fait en téléchargement progressif. La technologie Flash Player permet en effet de lire les premières secondes de votre fichier alors que la fin du chargement n'est pas encore terminée. Vous pouvez ainsi lire une vidéo pendant que le fichier est progressivement téléchargé.

Dans ce chapitre, nous allons vous présenter une technique de diffusion qui fait appel à un serveur : c'est ce qu'on appelle le *streaming vidéo*. L'avantage de ce mode de diffusion est de pouvoir accéder à différents instants de la vidéo dès le début de sa consultation. De plus, le fichier vidéo ne se charge pas sur la machine comme c'est le cas pour le téléchargement progressif.

Diffuser une vidéo en streaming

Animation : KVideoDirectStream/video_streamSymboleVideo.fla

Pour diffuser une vidéo en streaming, vous devez disposer d'un serveur qui héberge et fait tourner l'application Flash Media Server.

1. Sur le serveur Flash Media, créez un dossier dans lequel vous allez placer votre fichier SWF (par exemple, david).

2. Dans ce dossier, créez un sous-dossier intitulé streams dans lequel vous en créerez un autre que vous nommerez studio. Vous pourriez ainsi créer autant de sous-dossiers que vous souhaitez au même niveau que le dossier studio, afin de ranger les différentes vidéos que vous désirez proposer pour une consultation en ligne.
3. Dans le sous-dossier studio, faites glisser une vidéo intitulée lionel.flv. Placez-en une également sur le disque dur de votre ordinateur le temps de programmer votre animation pour effectuer vos essais.
4. Créez une animation et enregistrez-la sur votre disque dur le temps du développement sur votre ordinateur. Une fois la programmation de votre animation terminée, placez-la sur le serveur dans le dossier racine intitulé david.
5. En haut à droite de la bibliothèque de l'animation, dans le menu local déroulant, sélectionnez la commande Nouvelle vidéo… Vous obtenez un symbole de type Vidéo.
6. Précisez que vous exploiterez cette vidéo en ActionScript.
7. Faites glisser ce symbole de type Vidéo sur la scène.
8. Nommez l'occurrence obtenue, par exemple `ecranVideo`
9. Saisissez le script ci-dessous :

```
connexionServeur = new NetConnection();
connexionServeur.connect("rtmp:/david/studio/");
flux = new NetStream(connexionServeur);
ecranVideo.attachVideo(flux);
flux.play("lionel");
```

Attention

Vous remarquerez que le nom du fichier auquel vous faites référence dans la méthode `play()` ne doit surtout pas contenir d'extension. Vous devez écrire lionel et non lionel.flv.

La première ligne nous permet d'établir une connexion entre le serveur et l'application cliente, c'est-à-dire le navigateur de l'utilisateur qui consulte une animation avec la description ci-dessus.

La deuxième ligne connecte l'utilisateur sur le serveur Flash Media en utilisant le protocole rtmp.

La troisième ligne permet de faire passer, au travers de la connexion préalablement établie, un flux dans lequel la diffusion de vidéo va être effectuée.

La quatrième ligne permet d'établir le lien entre l'occurrence vidéo de la scène et le flux.

Pour finir, la cinquième ligne choisit précisément un fichier à charger au travers de la connexion. Le nom du fichier ne doit pas comporter d'extension.

Diffuser une vidéo en direct (live)

Avant de diffuser un flux provenant d'une webcam, vous devez être capable de le récupérer. Pour être plus précis, vous allez non seulement devoir récupérer les images qui proviennent de l'appareil, mais également le son du microphone. Vous allez donc devoir commencer par faire appel aux deux classes `Camera` et `Microphone`.

Récupérer le flux de la webcam

Animation : KVideoDirectStream/video_Captation_WebCam.fla

Voici un premier script que vous pouvez essayer dans la procédure ci-dessous :

> **Remarque**
>
> La procédure ci-dessous est identique à celle que nous venons d'aborder dans l'exemple précédent. Seul le script change.

1. Créez une nouvelle animation et enregistrez-la.
2. Dans le menu local situé en haut à droite de la bibliothèque de l'animation, sélectionnez la commande Nouvelle vidéo… Précisez dans la fenêtre qui s'affiche que vous allez la contrôler par l'ActionScript.
3. Placez sur la scène le symbole de type Vidéo que vous venez de créer.
4. Réglez les dimensions de votre occurrence. Utilisez des valeurs ayant un ratio 4:3 (640 × 480 pixels ou 320 × 240 pixels).
5. Nommez l'occurrence obtenue, par exemple `ecranVideo`.
6. Cliquez sur une image clé de la timeline principale de l'animation (celle qui contient l'occurrence de votre symbole vidéo) puis dans la fenêtre Actions, saisissez le script ci-après :

```
var fluxWebcamEntrant:Camera = Camera.get();
fluxWebcamEntrant.setMode(320, 240, 15);
var fluxMicroEntrant:Microphone = Microphone.get();
ecranVideo.attachVideo(fluxWebcamEntrant);
```

Pour l'instant, nous avons récupéré le son, mais nous ne l'exploitons pas ; c'est en prévision des lignes qui vont suivre. Par ailleurs, si ces lignes vous semblent trop complexes, voici une simplification du code (nous ne typons pas les identifiants) :

```
fluxWebcamEntrant = Camera.get();
fluxWebcamEntrant.setMode(320, 240, 15);
fluxMicroEntrant = Microphone.get();
ecranVideo.attachVideo(fluxWebcamEntrant);
```

Sur la scène, nous avons une occurrence intitulée `ecranVideo`, sur laquelle, grâce à la méthode `attachVideo()`, nous réussissons à canaliser le flux de la webcam. La méthode `setMode()`

n'est là que pour préciser la taille et le débit de l'acquisition de votre vidéo. Essayez de changer la valeur `15` en `2`, et vous comprendrez pleinement le sens et l'importance de ce paramètre. Vous obtiendrez deux images par seconde.

> **Attention**
> Les valeurs que vous précisez dans la méthode `setMode()` ne doivent pas excéder les limites de votre matériel de capture. Certaines webcams présentent parfois des limites très faibles (taille et débit maximal faibles).

Essayons à présent d'aller un peu plus loin et de diffuser le flux en provenance de la webcam sur un serveur Flash Media.

Diffuser le flux d'une webcam sur un serveur Flash Media

Animation : KVideoDirectStream/video_Diffusion_FMS_WebCam.fla

Pour commencer, vous devez établir une connexion entre le client (l'animation qui se trouve dans la fenêtre du navigateur) et le serveur Flash Media, comme nous l'avons mentionné auparavant. Nous ne typerons pas nos instances pour vous présenter un code plus simple.

```
connexionServeur = new NetConnection();
connexionServeur.connect("rtmp:/david/enregistrements");
```

Comme vous pouvez le constater, nous avons précisé, entre les parenthèses de la méthode `connect()`, le chemin du répertoire dans lequel les vidéos vont être enregistrées.

> **Remarque**
> Vous ne devez utiliser qu'un seul slash (caractère /) après la déclaration de protocole rtmp. Cela donne rtmp:/david et non rtmp://david.

Maintenant que votre connexion est établie, nous devons faire passer un flux auquel nous rattachons les données provenant de la webcam.

```
fluxDeDiffusion = new NetStream(connexionServeur);
fluxDeDiffusion.attachVideo(fluxWebcamEntrant);
```

Nous devons également faire passer dans le flux, le son provenant du micro branché sur votre ordinateur (il peut également s'agir du micro intégré à la webcam).

```
fluxDeDiffusion.attachAudio(fluxMicroEntrant);
```

Pour l'instant, aucune image ne sort réellement de votre ordinateur. Pour cela, vous devez publiez votre captation sur le serveur.

```
fluxDeDiffusion.publish("videoDavidDirect", "live");
```

Voici le script complet :

```
var fluxWebcamEntrant:Camera = Camera.get();
fluxWebcamEntrant.setMode(320, 240, 15);
var fluxMicroEntrant:Microphone = Microphone.get();
ecranVideo.attachVideo(fluxWebcamEntrant);
//
connexionServeur = new NetConnection();
connexionServeur.connect("rtmp:/david/enregistrements");
//
fluxDeDiffusion = new NetStream(connexionServeur);
fluxDeDiffusion.attachVideo(fluxWebcamEntrant);
fluxDeDiffusion.attachAudio(fluxMicroEntrant);
fluxDeDiffusion.publish("videoDavidDirect", "record");
```

La méthode `publish()` comporte deux paramètres. Le premier est le nom du fichier qui est créé sur le serveur, temporairement (`live`), ou définitivement (`record`). C'est sur lui que nous allons nous connecter dans le prochain développement. Le deuxième paramètre permet de définir le mode d'enregistrement. Le mode `live` a pour effet de diffuser un flux en direct sans qu'aucun fichier ne soit définitivement enregistré sur le serveur. Lorsqu'une diffusion est terminée, il n'est plus possible de visualiser ce qui a été diffusé. En revanche, le mode `record` permet d'enregistrer un fichier sur le serveur. Lorsqu'une diffusion est terminée, il est alors possible de revoir le fichier enregistré sur le serveur.

Si vous placez à présent votre animation sur un serveur qui fait tourner l'application Flash Media Server, vous pouvez diffuser sur Internet un flux vidéo provenant de votre webcam. Il suffit de charger dans votre navigateur la page Web qui contient votre animation. Nous devons maintenant réaliser une animation qui va être capable de lire le flux en cours de diffusion.

Réception d'un flux provenant d'une webcam

Animation : KVideoDirectStream/video_Reception_FMS_WebCam.fla

Pour être plus précis, nous allons charger le flux audio et vidéo qui provient d'un serveur Flash Media, comme nous vous l'avons expliqué en introduction à ce chapitre. Pour commencer, nous devons établir une connexion au serveur Flash Media qui propose la diffusion d'un flux, comme nous l'avons fait pour la diffusion.

```
connexionServeur = new NetConnection();
connexionServeur.connect("rtmp:/david/enregistrements");
```

Vous remarquerez que nous utilisons le même chemin que celui utilisé lors de la diffusion. Le flux est également établi, comme pour la diffusion.

```
fluxReception = new NetStream(connexionServeur);
```

Nous devons ensuite rattacher le flux vidéo à une occurrence de symbole de type Vidéo. Le flux audio est quant à lui utilisé avec une instance de la classe `Sound()`.

```
ecranVideo.attachVideo(fluxReception);
ecoute = new Sound();
ecoute.attachAudio(fluxReception);
```

Pour finir, nous devons chercher à lire un flux. Rappelez-vous que nous avions utilisé dans notre exemple de diffusion, le nom `videoDavidDirect`. Nous devons donc le réutiliser.

```
fluxReception.play("videoDavidDirect");
```

Voici le script complet :

```
var fluxWebcamEntrant:Camera = Camera.get();
fluxWebcamEntrant.setMode(320, 240, 15);
var fluxMicroEntrant:Microphone = Microphone.get();
ecranVideo.attachVideo(fluxWebcamEntrant);
//
connexionServeur = new NetConnection();
connexionServeur.connect("rtmp:/david/enregistrements");
//
fluxReception = new NetStream(connexionServeur);
//
ecranVideo.attachVideo(fluxReception);
//
ecoute = new Sound();
ecoute.attachAudio(fluxReception);
//
fluxReception.play("videoDavidDirect");
```

Nous avons volontairement voulu dissocier deux exemples de fichiers qui proposent la diffusion et la réception. Il va de soi que vous pouvez tout à fait concevoir une animation dans laquelle il y aurait deux instances de symbole de type Vidéo sur la scène, avec des noms d'occurrences différents, et vous pourriez alors à la fois envoyer l'image provenant de votre webcam et recevoir celle de quelqu'un d'autre. Votre application devrait alors prévoir des textes de saisie sur la scène afin de pouvoir définir des noms de diffusion et de réception. Nous avons développé une telle animation dans un ouvrage précédent intitulé *120 scripts pour Flash 8*, éditions Eyrolles. Nous l'avons replacée dans le dossier KVideoDirectStream et intitulée MediasVisioConference.fla.

Informations sur une connexion Flash Media Server

Animation : KVideoDirectStream/video_Infos_Connect_FMS.fla

Vous aurez sûrement besoin d'obtenir des informations sur une connexion que vous tentez d'établir, surtout si cette opération rencontre des problèmes.

Pour cela, vous disposez du gestionnaire `onStatus`, que vous pouvez utiliser avec les instances des classes `NetStream` ou `NetConnection`. Vous pourrez donc obtenir, indépendamment, des informations sur les deux actions, à savoir sur la connexion et le passage du flux.

Faites un essai dans une animation en utilisant les lignes ci-après (adaptez éventuellement les noms d'instances si vous n'avez pas utilisé les mêmes).

```
connexionServeur.onStatus = function(etat) {
  trace(etat.code);
};
```

et

```
flux.onStatus = function(etat) {
  trace(etat.code);
};
```

Remarque

Les mots `connexionServeur` et `flux` sont les noms d'instances que nous avons créées préalablement avec le constructeur `new`.

Dans l'exemple qui va suivre, nous avons prévu deux voyants lumineux (occurrences intitulées `case1` et `case2`) qui vont changer de niveau de transparence en fonction de la connexion.

Un texte dynamique va également afficher les différents états de connexion.

```
case1._alpha = 50;
case2._alpha = 50;
//
var vInformations = "";
//
connexionServeur = new NetConnection();
connexionServeur.connect("rtmp:/david/studio/");
flux = new NetStream(connexionServeur);
ecranVideo.attachVideo(flux);
flux.play("coche");
//
//
btLecture.onPress = function() {
  flux.play("coche");
};
btPause.onPress = function() {
  flux.pause();
};
btStop.onPress = function() {
  flux.play("");
};
//
//
connexionServeur.onStatus = function(etat) {
  vInformations += etat.code+newline;
  if (etat.code == "NetConnection.Connect.Success") {
        case2._alpha = 100;
  }
  if (etat.code == "NetConnection.Connect.Failed") {
        case2._alpha = 50;
  }
};
```

```
flux.onStatus = function(etat) {
  vInformations += etat.code+newline;
  etatReception = etat.code;
  if (etat.code == "NetStream.Play.Start" || etat.code
  ➥== "NetStream.Play.PublishNotify") {
        case1._alpha = 100;
  }
  if (etat.code == "NetStream.Play.Stop" || etat.code
  ➥== "NetStream.Play.UnpublishNotify") {
        case1._alpha = 50;
  }
};
```

En conclusion, nous pouvons ajouter que l'utilisation de Flash Media Server est indispensable dans les cas suivants :

- utilisation d'une webcam pour chater (texte, vidéo et/ou audio) ;
- enregistrements de séquences en lignes.

Il n'est pas indispensable si vous souhaitez simplement diffuser de la vidéo en ligne. Le téléchargement progressif suffit dans la plupart des cas.

Pour information, Flash Media Server propose bien d'autres possibilités que nous ne pouvons pas développer dans cet ouvrage, entièrement dédié à la vidéo. Sachez simplement que pour la réalisation de jeux multi-joueurs en ligne, Flash Media Server est une solution envisageable.

7

Manipuler le XML et le composant List dans une animation

Le chapitre que nous allons à présent aborder n'est pas fondamentalement nécessaire pour gérer la vidéo dans Flash. Cependant, il y a de très fortes chances que vous soyez amené un jour à gérer la vidéo autrement qu'en l'affichant simplement avec son contrôleur sur la scène. Pour développer des applications de Rich Media, vous devrez nécessairement faire appel à des fichiers XML. Pour synchroniser le déroulé d'une vidéo avec l'affichage de textes et/ou d'images à des instants précis de la séquence, pour proposer une liste de vidéos à consulter, pour contrôler la lecture d'une vidéo au clavier, il sera conseillé d'utiliser respectivement un fichier XML, un composant `List` ou `ComboBox` et la classe `Key`.

Apprendre à utiliser le XML dans une animation

À quoi sert le XML ?

Il est à la fois très simple et très difficile de répondre à cette question dans la mesure où la fonction première d'un fichier XML est toujours la même, c'est-à-dire stocker des données sous forme de texte. L'utilisation qui est faite du XML est quant à elle très variable.

Pour illustrer notre dernier propos, voici de nombreux exemples de fichiers XML. Nous pensons que dans un premier temps, c'est à travers la présentation de plusieurs cas que vous allez commencer à comprendre le principe, l'intérêt et la structure d'un arbre XML. Dans un deuxième temps, nous ferons une analogie avec une notion qui vous est plus proche (un listing de personnes) avant de terminer dans une troisième partie par l'apprentissage concret de la construction d'un fichier XML et du code ActionScript nécessaire pour l'exploitation de ce type de données.

Remarque

Les trois premiers exemples ne vous proposent pas uniquement la présentation de structures XML, mais aussi les fichiers .fla qui les exploitent. Focalisez toute votre attention sur les arborescences XML et non sur le code ActionScript contenu dans les fichiers .fla. En effet, tant que vous n'avez pas lu tout ce chapitre dédié au XML, vous risquez de ne pas comprendre le sens des lignes d'instructions.

Fichier XML pour la réalisation d'un QCM

Pour réaliser ce type d'application, il est fortement conseillé de regrouper les questions et les réponses à choix multiple dans un fichier. Les commentaires du résultat du test figurent également dans ce fichier. Que ce soit dans Flash ou avec d'autres techniques/technologies, le fichier XML ci-dessous va être analysé (*to parse* en anglais), c'est-à-dire qu'il va être lu.

```
<QCM>
  <Question intitule="Lequel de ces hommes n'a pas été Président de la République ?">
        <Reponse point="0">Raymond POINCARÉ</Reponse>
        <Reponse point="1">Léon GAMBETTA</Reponse>
        <Reponse point="0">Alexandre MILLERAND</Reponse>
  </Question>
  <Question intitule="Que signifie l'acronyme PIB ?">
        <Reponse point="0">Produit International Brut</Reponse>
        <Reponse point="1">Produit Intérieur Brut</Reponse>
        <Reponse point="0">Produit Industriel Bénéfique</Reponse>
  </Question>
  <Question intitule="La Finlande est une...">
        <Reponse point="1">République</Reponse>
        <Reponse point="0">Monarchie</Reponse>
        <Reponse point="0">Monarchie parlementaire fédérale</Reponse>
```

Figure 7-1

La question de cette copie d'écran et les trois réponses qui l'accompagnent sont contenues dans un fichier XML.

```
  </Question>
  <Commentaires>
        <Commentaire>0 sur 5. Vous ne pourrez…</Commentaire>
        <Commentaire>1 sur 5. C'est un bon début…</Commentaire>
        <Commentaire>2 sur 5. Presque la moyenne.</Commentaire>
        <Commentaire>3 sur 5. Vous avez la moyenne.</Commentaire>
        <Commentaire>4 sur 5. L'erreur est humaine.</Commentaire>
        <Commentaire>5 sur 5. Parfait. No comment.</Commentaire>
  </Commentaires>
</QCM>
```

Source : retrouvez cet exemple dans le dossier Kxml au travers des fichiers xmlqcm.fla et qcm.xml.

Remarque

Cet exemple est extrait des exercices contenus dans *120 scripts pour Flash* aux éditions Eyrolles.

Fichier XML pour la réalisation d'un graphe dynamique

Dans cet exemple, nous n'utilisons plus le contenu d'un fichier XML dans le seul but d'afficher les valeurs qu'il contient, mais nous allons utiliser ces dernières pour définir la hauteur de certaines occurrences sur la scène.

```
<Ventes>
  <Legende>
        <TitreGraphique intitule="Ventes de l'année 2005" />
        <TitreAxeX intitule="Mois" />
        <TitreAxeY intitule="Milliers d'euros" />
  </Legende>
  <Donnees>
        <donnee intitule="Janvier">35</donnee>
        <donnee intitule="Février">71</donnee>
        <donnee intitule="Mars">45</donnee>
        <donnee intitule="Avril">62</donnee>
        <donnee intitule="Mai">38</donnee>
        <donnee intitule="Juin">12</donnee>
        <donnee intitule="Juillet">83</donnee>
        <donnee intitule="Août">22</donnee>
        <donnee intitule="Septembre">53</donnee>
        <donnee intitule="Octobre">25</donnee>
        <donnee intitule="Novembre">62</donnee>
        <donnee intitule="Décembre">39</donnee>
  </Donnees>
</Ventes>
```

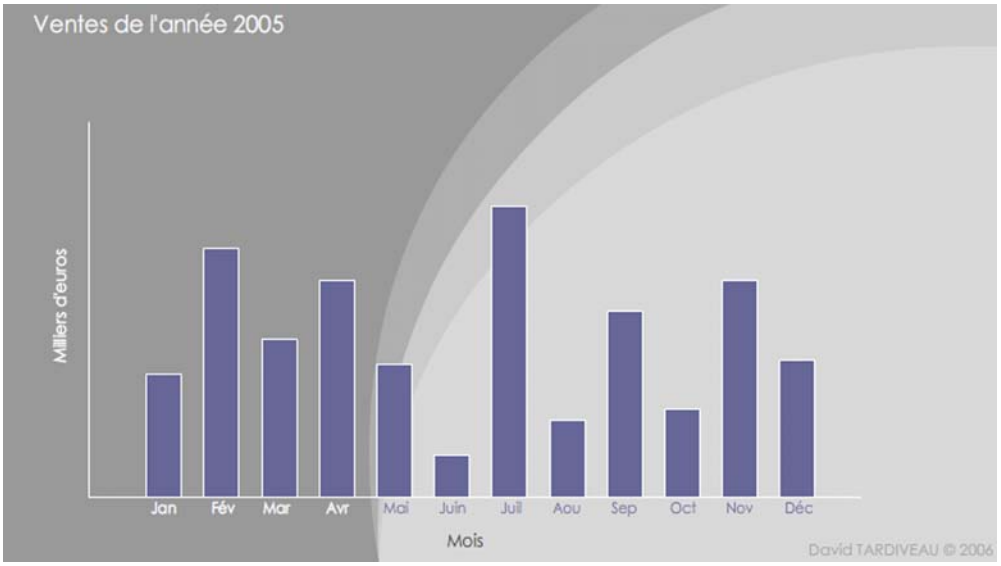


Figure 7-2

La hauteur de chaque barre est définie avec la propriété _xscale en utilisant des valeurs contenues dans le fichier XML ci-avant.

> **Source :** retrouvez cet exemple dans le dossier Kxml au travers des fichiers graphe.fla et graphe.xml.

Les valeurs des nœuds `donnee` sont utilisées pour changer la propriété `_yscale` des 12 occurrences présentes sur la scène. La construction d'un écran ou d'un document papier peut se faire schématiquement sous forme de blocs de textes et/ou d'images avec des dimensions précises (largeur et hauteur, positions des bords gauche et droit). Il serait donc très simple de concevoir un document XML qui contiendrait toutes ces informations pour la description d'une mise en page.

Fichier XML pour le tracé d'une forme

Dans cet autre exemple (où seules les premières lignes vous sont présentées car le fichier en contient 238), nous utilisons les valeurs des propriétés `xm`, `ym`, `x` et `y` pour dessiner la carte de France.

```
<Forme departX='541.95' departY='124'>
   <Coordonnees  xm='523.95' ym='132' x='523.95' y='132'/>
   <Coordonnees  xm='513.95' ym='130' x='513.95' y='130'/>
   <Coordonnees  xm='505.95' ym='141' x='505.95' y='141'/>
   <Coordonnees  xm='506.95' ym='151' x='506.95' y='151'/>
   <Coordonnees  xm='506.95' ym='169' x='506.95' y='169'/>
   <Coordonnees  xm='509.95' ym='176' x='509.95' y='176'/>
```

```
    <Coordonnees  xm='501.95' ym='177' x='501.95' y='183'/>
    <Coordonnees  xm='482.95' ym='187' x='485.95' y='191'/>
    <Coordonnees  xm='447.95' ym='198' x='448.95' y='214'/>
    <Coordonnees  xm='457.95' ym='216' x='457.95' y='216'/>
    <Coordonnees  xm='455.95' ym='223' x='455.95' y='223'/>
    <Coordonnees  xm='447.95' ym='220' x='447.95' y='220'/>
    <Coordonnees  xm='445.95' ym='222' x='445.95' y='222'/>
    <Coordonnees  xm='435.95' ym='227' x='435.95' y='227'/>
    <Coordonnees  xm='428.95' ym='217' x='413.95' y='222'/>
</Forme>
```

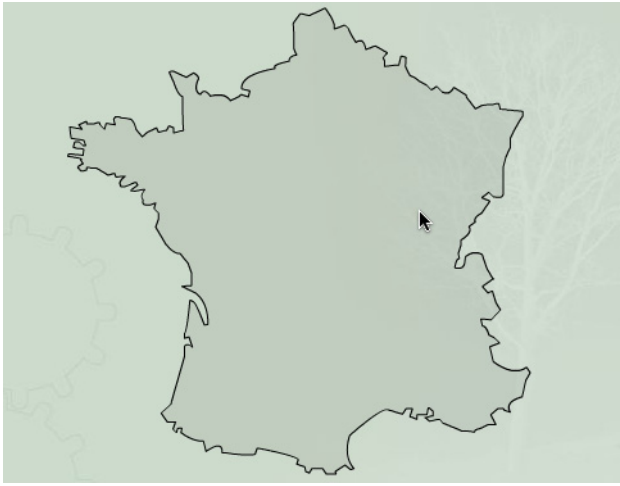


Figure 7-3

Cette carte de France n'existe pas, il ne s'agit pas d'un fichier ni d'un symbole, elle a été tracée en code grâce aux coordonnées ci-avant.

Source : retrouvez cet exemple dans le dossier Kxml au travers des fichiers carte.fla et cartefrance.xml.

Remarque

Cet exemple est extrait des exercices contenus dans *120 scripts pour Flash* aux éditions Eyrolles.

Après ces trois premiers exemples, essayons de donner davantage de sens à une exploitation potentielle d'un fichier XML avec la vidéo.

Fichier XML pour la réalisation d'un sous-titrage

En vidéo, la plus simple application que nous puissions faire du XML est de sous-titrer une séquence. En effet, l'ensemble de tous les textes qui s'affichent les uns après les

autres en bas d'un écran constitue un texte global que nous pourrions présenter sous la forme suivante :

(Ce dialogue est extrait de la vidéo intitulée *Lionel*. Il s'agit de la séquence que nous utilisons pour les techniques de sous-titrage).

– On peut y aller là ?
– Ouais attends, juste 2 minutes, j'arrange le micro.
– Moi je suis prêt.
– Ok d'accord, voilà c'est bon.
– Ça s'passe bien à l'école ?
– Ouais !
– Tu t'amuses bien ?
– Ouais !
– Qu'est-ce qui te plaît le plus dans l'école ?
– Ben c'est le midi.
– C'est le ?
– Le midi.
– Ah l'midi ! Pourquoi ?
– Parce que je peux mettre à manger dans mes...

Chaque tiret en début de phrase annonce l'expression d'une nouvelle réplique. Nous aurions pu représenter ce même texte sous la forme suivante :

Lionel : On peut y aller là ?
Interviewer : Ouais attends, juste 2 minutes, j'arrange le micro.
Lionel : Moi je suis prêt.
Interviewer : Ok d'accord, voilà c'est bon.
Interviewer : Ça s'passe bien à l'école ?
Lionel : Ouais !
Interviewer : Tu t'amuses bien ?
Lionel : Ouais !
Interviewer : Qu'est-ce qui te plaît le plus dans l'école ?
Lionel : Ben c'est le midi.
Interviewer : C'est le ?
Lionel : Le midi.
Interviewer : Ah l'midi ! Pourquoi ?
Lionel : Parce que je peux mettre à manger dans mes...

Si nous devions maintenant représenter le texte ci-dessus sous la forme d'un fichier XML, voilà à quoi il pourrait ressembler :

```
<SousTitre voix="Lionel">On peut y aller là ?</SousTitre>
<SousTitre voix="Interviewer">Ouais attends, juste 2 minutes, j'arrange le micro.
➥</SousTitre>
<SousTitre voix="Lionel">Moi je suis prêt.</SousTitre>
<SousTitre voix="Interviewer">Ok d'accord, voilà c'est bon.</SousTitre>
<SousTitre voix="Interviewer">Ça s'passe bien à l'école ?</SousTitre>
<SousTitre voix="Lionel">Ouais !</SousTitre>
```

```
<SousTitre voix="Interviewer">Tu t'amuses bien ?</SousTitre>
<SousTitre voix="Lionel">Ouais !</SousTitre>
<SousTitre voix="Interviewer">Qu'est-ce qui te plaît le plus dans l'école ?</SousTitre>
```

Dans la copie d'écran suivante, les différentes images extraites du film *Lionel* contiennent les sous-titres correspondants aux répliques ci-dessus. Flash a permis d'aller parcourir le document XML afin de le traiter pour l'afficher lors de la lecture de la vidéo.

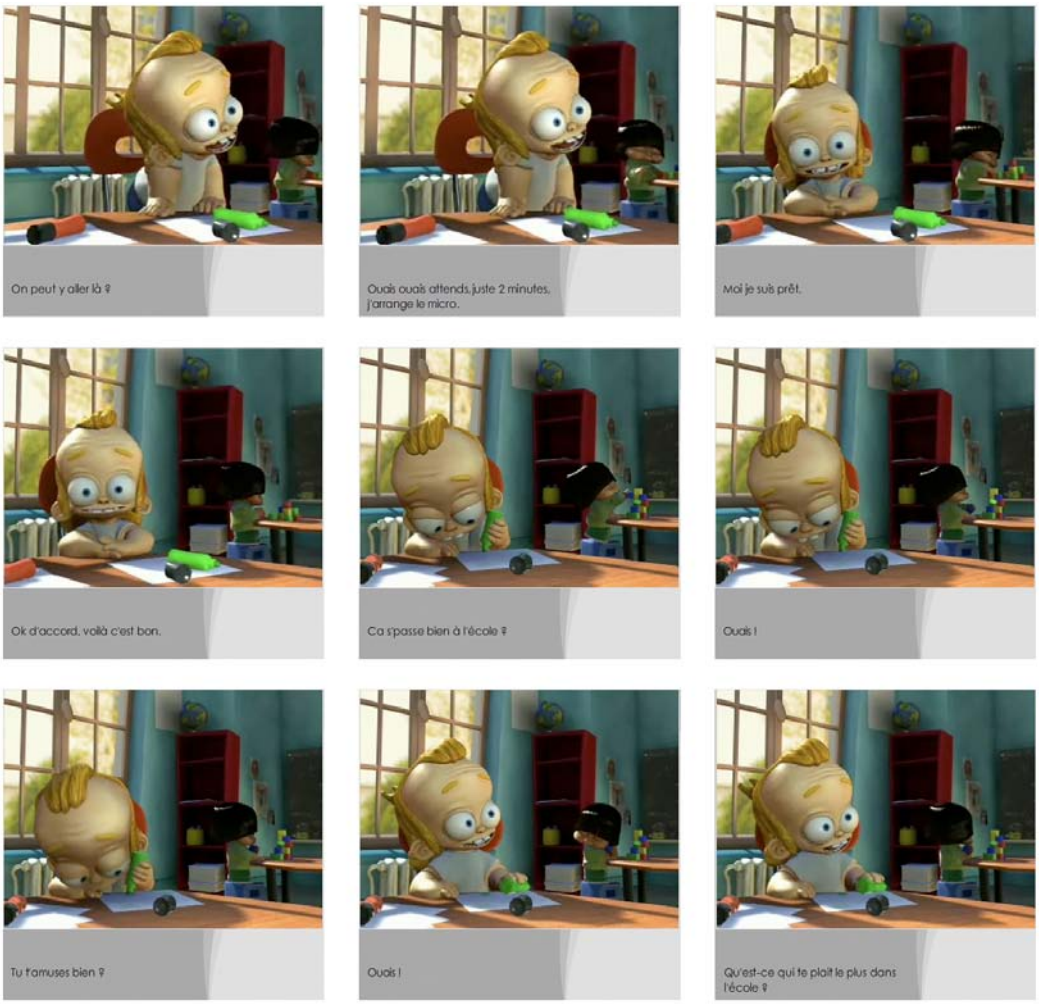


Figure 7-4

Ces neuf images sont extraites de la vidéo intitulée Lionel. *Les légendes sont extraites du fichier XML présentés ci-avant.*

Source : retrouvez cet exemple dans le dossier KVideosSousTitres au travers des fichiers video_sstitrexmladdCue.fla et soustitreslionel.xml.

Fichier XML pour la réalisation d'un diaporama

Si vous deviez réaliser un diaporama en Flash, vous auriez tout intérêt à opter pour un chargement dynamique des images avec la méthode `loadMovie()` ou la classe `MovieClipLoader()`. Dans ce cas, comment faire un listing des images que vous allez utiliser pour votre diaporama ? La création d'un fichier XML représente la meilleure solution, en voici un exemple :

```
<Visuels>
<Visuel>fruitpoire.jpg</ Visuel>
<Visuel>fruitorange.jpg</ Visuel>
<Visuel>fruitfraise.jpg</ Visuel>
<Visuel>legumeharicot.jpg</ Visuel>
<Visuel>fruitabricot.jpg</ Visuel>
<Visuel>fruitpeche.jpg</ Visuel>
<Visuel>legumeepinards.jpg</ Visuel>
</Visuels>
```

Pour cet exemple, nous n'avons pas développé d'animation (.fla) correspondante pour l'exploitation du fichier ; notre objectif était simplement de vous démontrer qu'une structure XML peut être extrêmement simple.

Analogie du XML

Pour aborder une analogie du XML, prenons l'exemple d'un professeur d'anglais (de collège ou de lycée) qui rencontre une de ses classes en début d'année pour la première fois. Souvenez-vous, au cours de la première séance de chaque matière, nous refaisions toujours la même chose ! Une fiche d'identité ! « Sortez une feuille de cahier, coupez-la en deux et inscrivez-y vos nom, prénoms, date de naissance et professions de vos parents.» Qui n'a jamais réalisé une telle fiche ? En voici un exemple.

Figure 7-5

Cette fiche correspond tout à fait à celle que nous avons tous remplie au moins une fois à la rencontre de chaque nouveau professeur en début d'année.

Toutes les fiches d'une même classe étaient ensuite sûrement regroupées dans une boîte ou un classeur afin de pouvoir constituer un ensemble de données communes dans le but de retrouver rapidement les informations relatives à un élève.

Partons de cet exemple pour rédiger un fichier XML contenant l'ensemble des informations relatives à chaque élève d'une classe.

```
<laclasse nom="6eA" annee="2006-2007">
  <eleve>
        <nom>TARDIVEAU</nom>
        <prenom>Marine</prenom>
        <datenaissance>15 septembre 1996</datenaissance>
        <telephone>02.30.40.50.60</telephone>
        <professionpere>Enseignant</professionpere>
        <professionmere>Mère au foyer</professionmere>
  </eleve>
  <eleve>
        <nom>MARTINI</nom>
        <prenom>Yvonne</prenom>
        <datenaissance>13 janvier 1997</datenaissance>
        <telephone>02.30.40.50.63</telephone>
        <professionpere>Cuisinier</professionpere>
        <professionmere>Mère au foyer</professionmere>
  </eleve>
  <eleve>
        <nom>PREYTAL</nom>
        <prenom>Laurence</prenom>
        <datenaissance>6 février 1997</datenaissance>
        <telephone>02.30.40.53.60</telephone>
        <professionpere>Pharmacien</professionpere>
        <professionmere>Enseignante</professionmere>
  </eleve>
  <eleve>
        <nom>SOURDOULAUD</nom>
        <prenom>Jean-Charles</prenom>
        <datenaissance>1er avril 1997</datenaissance>
        <telephone>02.30.43.50.60</telephone>
        <professionpere>Électricien</professionpere>
        <professionmere>Secrétaire</professionmere>
  </eleve>
  <eleve>
        <nom>LEMARCHAND</nom>
        <prenom>Jérôme</prenom>
        <datenaissance>1er octobre 1996</datenaissance>
        <telephone>02.33.40.50.60</telephone>
        <professionpere>Chef d'entreprise</professionpere>
        <professionmere>Galeriste-Comptable</professionmere>
  </eleve>
</laclasse>
```

Comprendre la structure d'un fichier XML

Le nœud

Comme vous pouvez le constater, un document contient des parties récursives appelées des *nœuds*.

Un nœud commence et se termine par une balise qui contient une barre oblique (*slash*). La balise de fermeture peut être placée sur la même ligne ou dessous.

```
<eleve>
</eleve>
```

ou encore

```
<eleve></eleve>
```

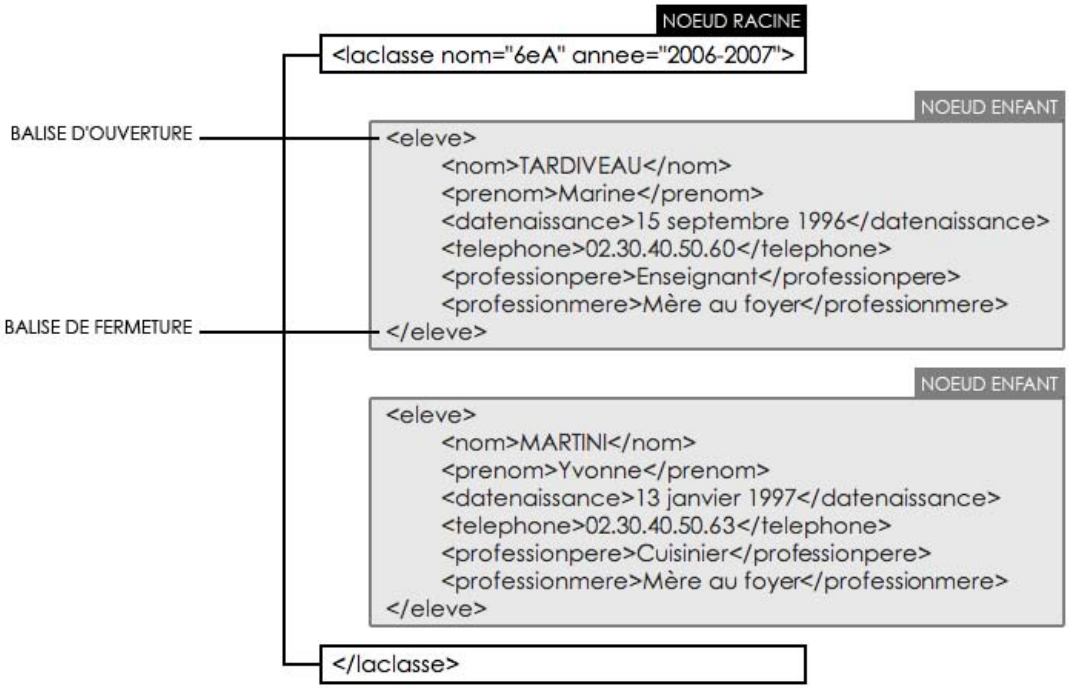


Figure 7-6

Ce document XML contient deux nœuds enfants qui en contiennent à leur tour six chacun.

Un document contient toujours un seul nœud racine (`<laclasse></laclasse>`) et ne peut donc pas en contenir plusieurs. Ainsi, dans l'exemple de ce fichier d'élèves, si la première balise ne s'y trouvait pas nous obtiendrions plusieurs nœuds racine dont les noms de balises seraient `<eleve>`. Ceci est impossible, nous aurions un fichier qualifié de *non valide*.

La première ligne d'un arbre XML

Commençons l'analyse de ce fichier par l'étude de la ligne suivante :

```
<laclasse nom="6eA" annee="2006-2007">
```

Attention

Gardez toujours à l'esprit que le nom d'une balise ne doit pas contenir d'espace, ni de caractères spéciaux ou accentués.

Cette ligne est la première à contenir un chevron d'ouverture de balise (signe inférieur). Dès qu'une même et nouvelle balise accompagnée du signe de la division apparaît dans l'arborescence, l'interpréteur du XML comprend que la balise est fermée et qu'il s'agit d'un nœud. Ainsi, dans la copie d'écran ci-avant, la première balise porte le nom `laclasse`, la prochaine à porter ce nom étant celle qui se situe à la fin de l'arbre. Notez qu'une barre oblique est insérée devant le nom : `</laclasse>`. La première balise de notre document est donc la balise racine.

Remarque

Vous observerez dans la plupart des exemples que la première ligne d'un fichier XML n'est pas le nœud racine. Il s'agit d'une ligne qui définit le codage de la page. Dans un fichier XML qui va être traité par Flash, cette ligne n'est pas obligatoire, c'est la raison pour laquelle nous n'y faisons pas référence systématiquement.

Un attribut

Une balise se caractérise par le fait qu'elle contient un nom, mais elle peut également posséder un ou plusieurs attributs. Il s'agit d'un mot auquel on associe avec le signe égal une valeur saisie entre guillemets (voir figure 7-7).

```
nom="6eA"
```

Cet attribut se trouve entre les chevrons (< et >). Pour en insérer plusieurs, séparez-les par une espace.

```
nom="6eA" annee="2006-2007"
```

Nœuds enfants et nœuds parents

Si vous avez besoin d'imbriquer plusieurs balises (comme `<nom>`, `<prenom>`, `<datenaissance>`, `<telephone>`, `<professionpere>` et `<professionmere>`) au sein d'une balise qualifiée de parent (par exemple, `<eleve>`), il vous suffit de les saisir les unes sous les autres comme nous le montre les figures 7-6 et 7-7. La ou les balises qui se trouvent à l'intérieur d'une autre sont qualifiées de *nœuds enfants*.

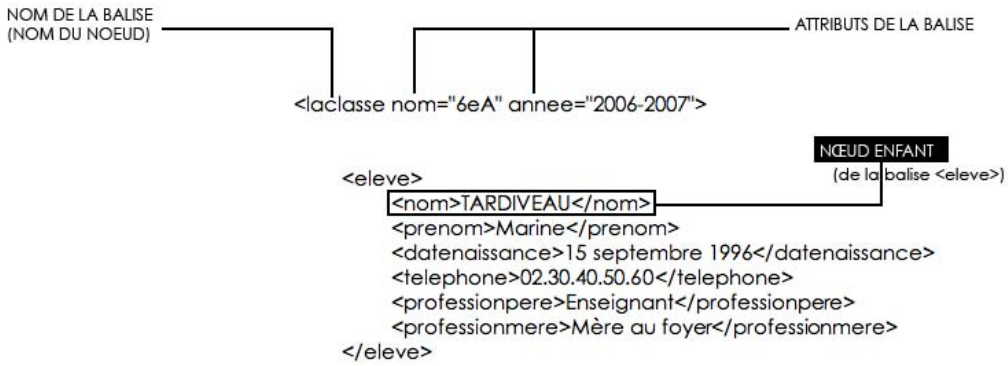


Figure 7-7

Une balise correspond toujours à un nœud.

Contenu d'un nœud

Entre deux balises, l'une à côté de l'autre ou l'une sous l'autre, vous pouvez insérer deux types de données :

- un autre nœud défini par deux balises (par exemple, `<prenom></prenom>`) ;
- un texte ou un chiffre.

Si vous placez du texte dans un nœud, vous n'êtes pas obligé de le saisir entre guillemets comme c'est le cas en programmation ; le XML n'est pas un langage.

En revanche, lorsque vous saisirez votre texte, gardez toujours à l'esprit que vous n'avez pas le droit d'insérer les signes < et > car l'interpréteur du XML va penser que vous ouvrez ou fermez une balise. L'usage de l'esperluette (&) pose également un problème et dans certains cas (pas en Flash), les apostrophes simples et doubles généreront aussi un bogue. Remplacez dans ce cas ces signes par leurs codes respectifs.

Tableau 7-1 Signes à ne pas utiliser en XML

Caractères	À remplacer par
<	&lt;
>	&gt;
&	&amp;
"	&quot;
'	&apos;

Remarque

En Flash, seuls les signes <, > et & sont à remplacer par les codes correspondants.

Dans certains cas, il sera difficile, tout du moins fastidieux, d'être obligé de remplacer tous ces signes par leurs codes correspondants. Utilisez dans ce cas l'échappement direct d'une chaîne de caractères avec la syntaxe suivante :

```
<![CDATA[Si a>b && b<c alors abc]]>
```

Grâce à la structure <![CDATA[ ]]>, le texte contenu entre les crochets n'est pas interprété.

Construire un fichier XML

Pour construire un fichier XML, commencez tout d'abord par lancer une application gérant le format UTF-8. L'encodage de vos fichiers XML va en effet devoir être fait en Unicode, ce qui permet de pouvoir saisir des caractères accentués et spéciaux dans les nœuds, pour définir la valeur de ces derniers.

> **Attention**
>
> Si vous n'avez pas lu les pages précédentes, consultez la partie ci-avant *Contenu d'un nœud* pour connaître les caractères à ne pas utiliser.

Suivez ensuite la procédure suivante :

1. Enregistrez votre document en spécifiant un nom de fichier qui ne contient pas d'espaces, ni aucun caractère accentué ou spécial.
2. Définissez une balise racine (saisissez la balise d'ouverture et celle de fermeture).
3. Rédigez votre arborescence XML sans jamais oublier de fermer une balise que vous avez ouverte.
4. Exécutez plusieurs fois le raccourci Ctrl+S (PC) ou Cmd+S (Mac) au cours de la rédaction de votre document pour le sauvegarder.
5. Terminez en vérifiant la validité de votre document en faisant glisser votre fichier dans la fenêtre d'un navigateur. Si ce dernier affiche correctement l'arborescence XML que vous venez de rédiger, votre document est valide ; dans le cas contraire, corrigez-le car il contient une erreur à la ligne spécifiée par le navigateur.

Comme vous l'aurez constaté, la méthode est simple, en revanche, le choix du nom et de la structure n'est pas évident lors des premières créations. Rassurez-vous, vous serez très rapidement capable de rédiger vos premières arborescences.

Il est un point très important qu'il ne faut pas négliger, l'évolution de l'exploitation du fichier XML. Vous devez toujours essayer d'anticiper en prévoyant la présence de balises vides que vous n'utiliserez pas obligatoirement au début.

Prenons l'exemple du fichier de la classe ci-dessus, où nous aurions pu ajouter les balises suivantes :

```
<eleve>
        <nom>TARDIVEAU</nom>
        <prenom>Marine</prenom>
        <datenaissance>15 septembre 1996</datenaissance>
        <telephone>02.30.40.50.60</telephone>
        <professionpere>Enseignant</professionpere>
        <professionmere>Mère au foyer</professionmere>
        <moyennetrim1></moyennetrim1>
        <moyennetrim2></moyennetrim2>
        <moyennetrim3></moyennetrim3>
  </eleve>
```

Ce sera en cours d'année que le professeur d'anglais de notre exemple va remplir progressivement son fichier XML pour le mettre à jour en ajoutant les notes de chaque élève.

Lorsque vous utiliserez un fichier XML en tant que base de données, vous découvrirez très rapidement que le copier-coller s'impose. Avant de faire cette manipulation, il faudra bien vous assurer que vous avez bien prévu toutes les balises, sinon, vous allez devoir jouer du rechercher-remplacer.

Remarque

Si vous devez gérer une base de données contenant de nombreuses informations, vous aurez tout intérêt à utiliser le binôme PHP et MySQL. Les pages XML seront alors obtenues dynamiquement au travers de requêtes.

À présent, essayons de construire ensemble un fichier en définissant un scénario.

Vous venez d'être embauché dans une société qui vend des produits et services dans le domaine des loisirs. Vous allez travailler avec la personne en charge de la billetterie. Vous observez lors de vos premières journées de travail, que vous consultez régulièrement un listing qui référence toutes les salles de spectacle de la région parisienne. Vous perdez beaucoup de temps à rechercher régulièrement différents types d'informations, vous décidez alors d'informatiser ce listing.

Créez un fichier texte que vous enregistrez en UTF-8 ou en Unicode sous le nom de sallesspectacles.xml. Commencez par saisir la balise racine.

```
<Salles>
</Salles>
```

À présent, vous ne devez plus ajouter de balises supplémentaires avant ou après ces deux premières. Nous devons créer un nœud enfant qui va contenir les informations suivantes :

- le nom de la salle ;
- le nombre de places maximal que supporte cette salle ;
- le type de salle (concert, théâtre, stade, polyvalente…) ;
- l'accès à la salle ;
- le nom de la personne à contacter ;
- le numéro de téléphone de l'administration de la salle ;

- l'adresse de la salle.

Le nom, le nombre de places et le type de salle vont être mis en paramètres de la balise de chaque nœud enfant. Ces informations sont effectivement assez courtes, elles peuvent donc être regroupées sur une seule ligne.

```
<salle nom="" nombreplacesmaxi="" type=""></salle>
```

Si nous n'avions que ces informations, nous pourrions d'ailleurs retenir la syntaxe suivante :

```
<salle nom="POPB" nombreplacesmaxi="17000" type="Polyvalente"/>
<salle nom="Zénith" nombreplacesmaxi="6400" type="Polyvalente"/>
<salle nom="Olympia" nombreplacesmaxi="3000" type="Concert"/>
<salle nom="Palais des sports" nombreplacesmaxi="6000" type="Concert"/>
```

Comme vous l'aurez peut-être remarqué, il n'y a plus qu'une seule balise, mais un slash a été ajouté à la fin de la ligne. Cette technique s'avère très efficace par son allègement des nœuds et sa lisibilité.

Revenons sur notre fichier et ajoutons-y les informations énumérées ci-dessus :

```
  <salle nom="" nombreplacesmaxi="" type="">
        <acces></acces>
        <personnecontact></personnecontact>
        <telephone></telephone>
        <adresse></adresse>
        <codepostal></codepostal>
        <ville></ville>
  </salle>
```

Pour l'instant, vous obtenez ce fichier :

```
<salles>
  <salle nom="" nombreplacesmaxi="" type="">
        <acces></acces>
        <personnecontact></personnecontact>
        <telephone></telephone>
        <adresse></adresse>
        <codepostal></codepostal>
        <ville></ville>
  </salle>
</salles>
```

Faites glisser le fichier sallesspectacles.xml dans la fenêtre d'un navigateur afin de le visualiser. Corrigez-le si vous obtenez un message d'erreur(s).

Copiez-collez à présent le nœud enfant (de la racine).

```
<salles>
  <salle nom="" nombreplacesmaxi="" type="">
        <acces></acces>
        <personnecontact></personnecontact>
        <telephone></telephone>
        <adresse></adresse>
        <codepostal></codepostal>
        <ville></ville>
  </salle>
```

```
  <salle nom="" nombreplacesmaxi="" type="">
        <acces></acces>
        <personnecontact></personnecontact>
        <telephone></telephone>
        <adresse></adresse>
        <codepostal></codepostal>
        <ville></ville>
  </salle>
  <salle nom="" nombreplacesmaxi="" type="">
        <acces></acces>
        <personnecontact></personnecontact>
        <telephone></telephone>
        <adresse></adresse>
        <codepostal></codepostal>
        <ville></ville>
  </salle>
</salles>
```

Maintenant, vous n'avez plus qu'à remplir les valeurs des balises.

Rédiger un script en ActionScript

Nous allons à présent passer à une nouvelle partie qui ne présente pas de difficulté particulière, et pourtant, nombreuses sont les personnes non initiées au XML dans Flash qui pensent que c'est trop difficile pour eux. Démontrons le contraire.

Pour nos explications, nous avons réalisé une animation (xmlapprentissage.fla) qui fait appel au fichier diaporama.xml. Ces deux fichiers se trouvent dans le dossier Kxml, mais nous vous conseillons cependant de reproduire cet exemple en suivant pas à pas les différentes étapes d'apprentissage ci-après.

Figure 7-8

Le nom de l'image et le texte de légende sont contenus dans un fichier XML.

Instancier la classe XML()

Vous venez de terminer la rédaction de votre document XML (en recopiant le contenu du fichier diaporama.xml) et vous souhaitez à présent lire les données qu'il contient à partir de votre animation Flash. Pour ce faire, vous devrez toujours commencer par écrire les deux lignes ci-dessous dans la fenêtre Actions, après avoir sélectionné la première image clé de la timeline principale de votre animation :

```
leslegendes = new XML();
leslegendes.load("diaporama.xml");
```

Dans notre exemple, rappelons que diaporama.xml est le nom du fichier que nous avons crée pour l'occasion. Voici ce qu'il contient au cas où vous ne l'auriez pas ouvert :

```
<Diaporama>
  <image nom="img1.jpg">Elle danse, elle danse en dormant.</image>
  <image nom="img1.jpg">Je chant'en dormant.</image>
  <image nom="img3.jpg">La penseuse de Rodin.</image>
  <image nom="img1.jpg">Allo j'écoute !</image>
</Diaporama>
```

Revenons sur la signification et le sens de ces deux premières lignes de code. Nous déclarons à Flash (nous lui expliquons) que le mot `leslegendes` est une instance de la classe XML. En clair, nous souhaitons pouvoir utiliser dans notre animation, un vocabulaire précis, celui du XML. À partir du moment où la première ligne a été exécutée, dès que nous allons écrire à nouveau le mot `leslegendes`, Flash va comprendre que le mot qui suit relève du XML.

> **Remarque**
>
> Si vous êtes novice en programmation, retenez bien que cette ligne aurait pu être écrite selon cette syntaxe : `var leslegendes:XML = new XML();` C'est ce qu'on appelle le *typage d'une variable* ou une *instance*.

Le mot-clé `new` est ce qu'on appelle un *constructeur* ; ce qui suit est toujours le nom d'une classe qui ne s'écrit pas toujours en capitales, mais démarre obligatoirement par une majuscule et se termine par des parenthèses.

Sur la deuxième ligne, le mot `load()` est ce qu'on appelle une *méthode de classe*. C'est cette dernière qui va effectuer le chargement du contenu du fichier XML dans la mémoire vive de l'ordinateur.

Maintenant, nous devons essayer de lire les valeurs et attributs des nœuds de notre arbre XML, mais nous devons d'abord expliquer à Flash qu'il doit ignorer d'éventuelles lignes vierges qui seraient contenues dans le fichier XML.

```
leslegendes.ignoreWhite = true;
```

Voici à quoi doit maintenant ressembler le script :

```
leslegendes = new XML();
leslegendes.load("diaporama.xml");
leslegendes.ignoreWhite = true;
```

Remarque

Dans certains exemples, l'instruction du chargement se trouve après le gestionnaire `onLoad`. C'est en effet plus logique, mais Flash ne fait pas la différence dans la mesure où l'exécution du code est bien plus rapide que le chargement d'un fichier XML.

Le gestionnaire onLoad

Avant de tenter de lire une des lignes de l'arbre XML, nous devons nous assurer que le chargement dans la mémoire vive de l'ordinateur est terminé. Attention, la logique de construction de votre algorithme vous amènerait sûrement à rédiger le script ci-dessous, malheureusement il n'est pas bon.

```
leslegendes = new XML();
leslegendes.load("diaporama.xml");
leslegendes.ignoreWhite = true;
vLegende = this.childNodes[0].childNodes[0].firstChild;
```

Vous essayez en effet de lire des informations en mémoire vive de l'ordinateur, mais elle ne sont sûrement pas chargées dans leur totalité. Vous devez attendre la fin du chargement, c'est le rôle du gestionnaire `onLoad`.

```
leslegendes = new XML();
leslegendes.load("diaporama.xml");
leslegendes.ignoreWhite = true;
leslegendes.onLoad = function() {
  vLegende = this.childNodes[0].childNodes[0].firstChild;
};
```

Pour l'instant, nous en convenons, vous ne savez pas encore décrypter la ligne qui commence par `vLegende`, nous y reviendrons plus loin dans ce développement.

C'est grâce à la structure ci-dessous que les lignes qu'elle contient vont pouvoir s'exécuter correctement.

```
leslegendes.onLoad = function() {
};
```

Concrètement, ce gestionnaire d'événement n'exécutera les lignes placées entre les accolades qu'à partir du moment où le chargement du fichier XML sera complètement terminé.

Vérifier le chargement du XML

Généralement, la première ligne que vous saisirez temporairement dans ce gestionnaire sera la suivante.

```
trace(this);
```

Cette ligne d'instruction permet de vérifier ce qui a réellement été chargé, car la commande `trace()`affiche dans la fenêtre Sortie, le contenu du chargement. Nous aurions pu remplacer le mot `this` par `lesLegendes` car il y fait référence.

Figure 7-9

Grâce à la fonction trace(), vous allez pouvoir vérifier le contenu du chargement.

Lire un nœud

Notre premier objectif est d'être capable de lire la valeur d'un nœud. En reprenant notre exemple, comment est-il possible d'afficher sur la scène, *Elle danse, elle danse en dormant*.

Nous devons commencer par placer un texte dynamique sur la scène que nous nommerons `vLegende`.

> **Rappel**
>
> Pour placer un texte dynamique sur la scène, commencer par placer un texte qui est de type statique, transformez-le en type dynamique via la palette Propriétés. N'oubliez pas également de définir le nom de variable comme le montre la figure 7-10.

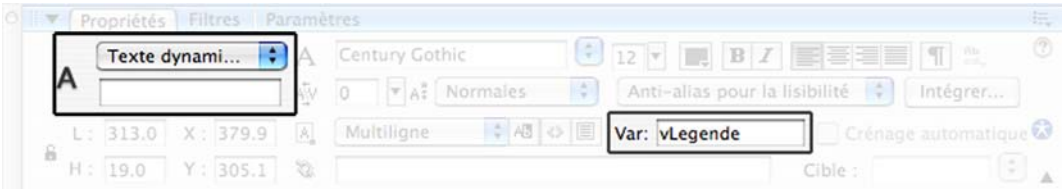


Figure 7-10

Le changement de type de texte en dynamique et l'attribution d'un nom de variable se font via la palette Propriétés.

Ensuite, dans la fenêtre Actions de l'animation, nous devons continuer notre script en cours :

```
leslegendes = new XML();
leslegendes.load("diaporama.xml");
leslegendes.ignoreWhite = true;
leslegendes.onLoad = function() {
  vLegende = this.childNodes[0].childNodes[0].firstChild;
};
```

Mais que signifie cette avant-dernière ligne que nous lisons pour la troisième fois ? Pour répondre à cette question, vous devez d'abord comprendre le principe de fonctionnement de navigation dans une arborescence XML.

Notre fichier XML est trop simple pour expliquer le système de lecture d'un nœud dans une arborescence. Pour cela, considérons les lignes suivantes :

```
<Ventes>
  <Legende>
        <TitreGraphique intitule="Ventes de l'année 2005" />
        <TitreAxeX intitule="Mois" />
        <TitreAxeY intitule="Milliers d'euros" />
  </Legende>
  <Donnees>
        <donnee intitule="Janvier">35</donnee>
        <donnee intitule="Février">71</donnee>
        <donnee intitule="Mars">45</donnee>
        <donnee intitule="Avril">62</donnee>
        <donnee intitule="Mai">38</donnee>
        <donnee intitule="Juin">12</donnee>
        <donnee intitule="Juillet">83</donnee>
        <donnee intitule="Août">22</donnee>
        <donnee intitule="Septembre">53</donnee>
        <donnee intitule="Octobre">25</donnee>
        <donnee intitule="Novembre">62</donnee>
        <donnee intitule="Décembre">39</donnee>
  </Donnees>
</Ventes>
```

Quelle que soit la nature de votre arborescence, vous devrez toujours commencer une ligne d'instruction en faisant référence au nom de l'instance XML, suivi du mot `firstChild` ou `childNodes[0]`.

```
vLegende = leslegendes.childNodes[0]
```

ou

```
vLegende = leslegendes.firstChild
```

Pour l'instant ces deux lignes d'instructions ont pour effet de placer la totalité du document XML dans la variable `vLegende`.

En ajoutant à nouveau `childNodes[]`, vous allez pouvoir désigner un nouveau nœud. La ligne suivante permet de lire les données contenues dans le nœud `Legende`.

```
vLegende = leslegendes.childNodes[0].childNodes[0]
```

Cette ligne permet quant à elle de lire les données contenues dans le nœud `Donnees`.

```
vLegende = leslegendes.childNodes[0].childNodes[1]
```

Attention : le comptage ou la numérotation des nœuds se fait toujours à partir de 0. Ainsi, dans notre dernier exemple, le nœud qui contient la valeur du mois de février est le

deuxième de la liste, mais il porte l'index numéro 1. Le mois de décembre porte l'index numéro 11.

En ajoutant un nouveau `childNodes[]`, vous allez pouvoir lire les nœuds enfants. Le tableau ci-après vous indique les correspondances entre le code et le résultat obtenu.

Tableau 7-2 Correspondance entre des lignes d'instructions et l'extraction obtenue

Ligne de code	Extraction obtenue

`leslegendes.childNodes[0].childNodes[0]`	`<Legende>`
	`<TitreGraphique intitule="Ventes de l'année 2005" />`
	`<TitreAxeX intitule="Mois" />`
	`<TitreAxeY intitule="Milliers d'euros" />`
	`</Legende>`

`leslegendes.childNodes[0].childNodes[1]`	`<Donnees>`
	`<donnee intitule="Janvier">35</donnee>`
	…
	`<donnee intitule="Décembre">39</donnee>`
	`</Donnees>`

`leslegendes.childNodes[0].childNodes[1].childNodes[0]`	`<donnee intitule="Janvier">35</donnee>`
`leslegendes.childNodes[0].childNodes[1].childNodes[0].childNodes[0]`	35
`leslegendes.childNodes[0].childNodes[1].childNodes[1]`	`<donnee intitule="Février">71</donnee>`
`leslegendes.childNodes[0].childNodes[1].childNodes[1].childNodes[0]`	71

En conclusion, plus la ligne est longue, plus elle permet l'accès en profondeur aux données des nœuds enfants. Vous noterez que toutes ces lignes contiennent toujours la même racine ; il serait donc intéressant de la raccourcir en changeant notre script ainsi :

```
racine = leslegendes.childNodes[0];
vLegende = racine.childNodes[1].childNodes[0];
```

Maintenant, notre script devient donc :

```
leslegendes = new XML();
leslegendes.load("diaporama.xml");
leslegendes.ignoreWhite = true;
leslegendes.onLoad = function() {
  racine = this.childNodes[0];
  vLegende =racine.childNodes[0].firstChild;
};
```

Remarque
Si vous retirez le dernier `firstChild` ou `childNodes[0]` d'une ligne, des balises encadreront votre texte. Vous devez alors cocher la case de la figure 7-11 afin de permettre une interprétation.

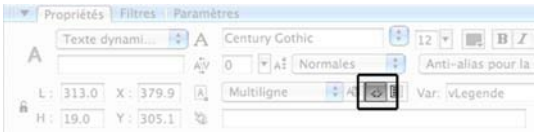


Figure 7-11

Si cette case est cochée, vous devez penser à retirer le firstChild (ou childNodes[0]) de fin.

Lire un attribut

Vous aurez sûrement noté que le nom de l'image n'est pas compris entre deux balises, mais en tant qu'attribut. Pour pouvoir le lire, la syntaxe est simple. Vous devez faire référence à la ligne et ajouter le mot `attributes` suivi du nom de la propriété. Dans notre exemple, cela donne :

```
leslegendes.onLoad = function() {
  racine = this.childNodes[0];
  vLegende = racine.childNodes[0];
  cadre.loadMovie("visuels/"+racine.childNodes[0].attributes.nom);
};
```

Précisons que le mot cadre est le nom de l'occurrence dans laquelle se charge l'image.

Nombre de nœuds

Dans certains cas, vous aurez besoin de connaître le nombre de nœuds contenus à l'intérieur d'un autre. La propriété `length` nous permet de les compter lorsqu'ils sont adjoints au chemin du nœud. Si nous exécutions la ligne d'instruction suivante, la valeur de la variable `nbrNoeuds` serait 4.

```
nbrNoeuds = racine.childNodes.length;
```

Vous possédez désormais les bases nécessaires pour manipuler un fichier XML à partir d'une animation Flash. Pour donner davantage de sens à notre exemple, nous avons développé une deuxième animation intitulée xmlapprentissage2.fla. Consultez-la pour découvrir que le diaporama affiche en continu et en boucle les images décrites dans le fichier XML. Nous vous présentons tout de même son script car nous souhaitons apporter quelques commentaires.

```
var numeroImage:Number = 0;
//
leslegendes = new XML();
leslegendes.load("diaporama.xml");
leslegendes.ignoreWhite = true;
//
//
leslegendes.onLoad = function() {
  racine = this.childNodes[0];
  nbrNoeuds = racine.childNodes.length;
  afficherImage();
};
```

```
//
function afficherImage() {
  numeroImage++;
  if (numeroImage>=nbrNoeuds) {
        numeroImage = 0;
  }
  vLegende = racine.childNodes[numeroImage];
  cadre.loadMovie("visuels/"+racine.childNodes[numeroImage].attributes.nom);
}
//
lancerDiaporama = setInterval(afficherImage, 2000);
//
ecouteur = new Object();
ecouteur.onKeyDown = function() {
  if (lancerDiaporama != undefined) {
        clearInterval(lancerDiaporama);
        delete lancerDiaporama;
  } else {
        lancerDiaporama = setInterval(afficherImage, 2000);
  }
};
Key.addListener(ecouteur);
```

Une fonction a été créée pour éviter de copier-coller plusieurs lignes de code, mais surtout, elle est indispensable si nous voulons temporiser le défilement des images avec la fonction `setInterval()`.

Le mot `ecouteur` a été choisi presque par hasard (nous aurions pu retenir n'importe quel autre mot) ; il sert d'entité pour pouvoir gérer la pause en fonction des pressions exercées sur les touches du clavier. Référez-vous à la dernière partie de ce chapitre pour apprendre à gérer l'interactivité d'une animation à partir du clavier.

Exercices de mise en pratique

Après ces quelques pages de théorie et d'exemples au cours desquelles vous avez sûrement testé le bon fonctionnement de ce que vous avez appris, passons à la pratique. En utilisant le fichier soustitres.xml qui se trouve dans le dossier Kxml, recréez les animations correspondant aux fichiers xml1.swf, xml2.swf, xml3.swf et xml4.swf.

Exercice 1 : le premier nœud est simplement affiché sur la scène.

Exercice 2 : deux boutons (`Suivante` et `Précédente`) permettent de visualiser les différentes répliques d'un dialogue.

Exercice 3 : si vous n'avez pas utilisé les fonctions (`function()`) dans l'exercice 2, recommencez l'animation en utilisant cette technique.

Exercice 4 : l'affichage des répliques se fait automatiquement toutes les 2 secondes.

Les corrections de ces quatre exercices se trouvent dans les fichiers xml1.fla, xml2.fla, xml3. fla et xml4. fla du dossier Kxml.

Apprendre à utiliser les composants List et ComboBox

Avant de développer nos explications sur le fonctionnement de ce composant, commençons par répondre à ces deux questions :

– À quoi sert ou qu'est-ce qu'un composant ?

– Pourquoi avoir choisi ce composant ?

À la première question, nous pouvons répondre qu'un composant est un symbole modèle préprogrammé. Pour être plus précis, certains symboles accessibles dans une bibliothèque précise de Flash peuvent êtres placés sur la scène sans que vous ayez à programmer leur fonctionnement. Il s'agit de la bibliothèque Composants qui contient de nombreux symboles servant à gérer les médias, des éléments de formulaires, des parties d'interface, etc. Il suffit de glisser-déposer un composant sur la scène pour avoir ensuite accès à la palette Paramètres dans laquelle vous réglez certaines options.

À la deuxième question, nous pouvons répondre que ce composant est celui que vous retrouverez le plus souvent dans une animation. Par ailleurs, en apprenant le fonctionnement de cet élément de formulaire, vous apprenez aussi celui du composant `comboBox`.

Figure 7-12

Le composant peut être rempli de plusieurs manières, l'ActionScript avec un fichier XML reste la meilleure solution.

Comme vous pouvez le constater, l'occurrence du composant `List` de la copie d'écran ci-avant contient déjà un contenu (des mois de l'année). Comment peut-on ajouter des données personnalisées ?

Ajouter des entrées dans un composant

Vous disposez de deux techniques pour remplir le contenu d'un composant. Soit vous renseignez quelques valeurs dans la palette Paramètres (voir figure 7-13) ou bien vous faites appel à l'ActionScript. La dernière solution reste la plus rapide, la plus souple, la plus dynamique et la plus optimisée.

Avant d'aller plus loin dans l'apprentissage du fonctionnement de ce type de symbole, rappelons que vous devez toujours nommer vos occurrences ; c'est d'autant plus vrai pour celles qui sont issues des composants.

L'essentiel

Nous pourrions résumer tout le développement de nos explications en une seule ligne.

```
annee_cmp.addItem("Janvier");
```

Dans cet exemple, `annee_cmp` est le nom d'occurrence d'un composant de `List`. `addItem()` est la méthode à utiliser pour ajouter un élément visible dans la liste comme le montre la figure 7-12.

Utiliser la palette Propriétés

Dans le cas où vous devriez remplir l'occurrence d'un composant sans passer par l'ActionScript, la figure 7-13 vous présente la marche à suivre. Précisons tout de même que vous n'aurez pas le choix, vous serez obligé de programmer pour gérer l'action à exécuter lorsqu'un élément de l'occurrence du composant est sélectionné.

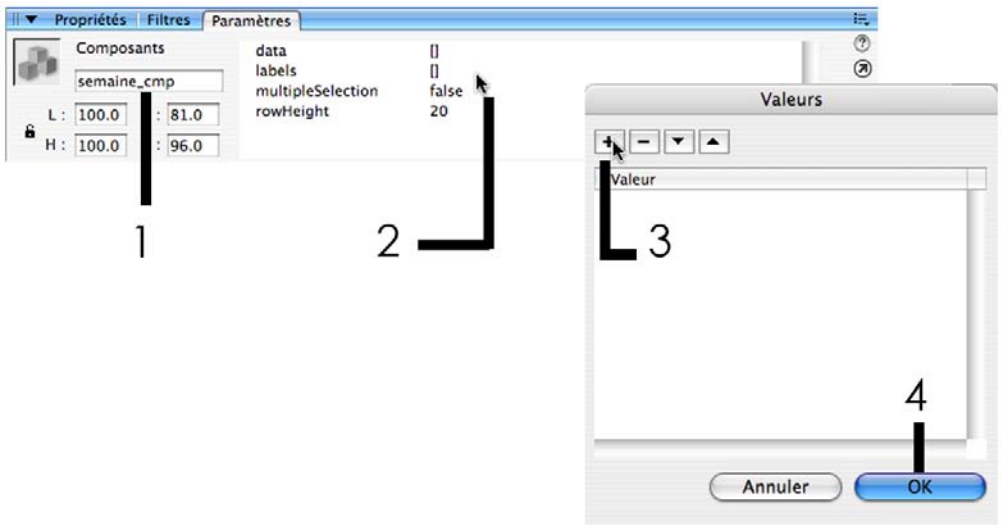


Figure 7-13

Nommez votre occurrence, double-cliquez sur la ligne labels dans la fenêtre Paramètres, puis ajoutez des valeurs à chaque pression sur le bouton +.

Attention

Le terme Valeur de la fenêtre du même nom est ambigu dans la mesure où vous ajoutez des étiquettes (labels) et non des valeurs. Un développement à ce sujet figure quelques paragraphes plus loin dans cet ouvrage.

Utiliser l'ActionScript

Nous évoquions le fait qu'il n'y a que deux solutions, mais pour être plus précis, en ActionScript, vous avez encore plusieurs possibilités.

1. Commencez tout d'abord par placer un composant de type `List` sur la scène.
2. Nommez l'occurrence obtenue via la palette Propriétés (par exemple, `semaine_cmp` de la figure 7-13).
3. Cliquez sur l'image clé 1 de la timeline principale, puis dans la fenêtre Actions (vous devez toujours voir l'occurrence du composant sur la scène après ces deux clics). Le curseur de saisie de texte clignote dans cette fenêtre pour vous indiquer que vous pouvez taper votre script.

```
semaine_cmp.addItem("Lundi");
semaine_cmp.addItem("Mardi");
semaine_cmp.addItem("Mercredi");
semaine_cmp.addItem("Jeudi");
semaine_cmp.addItem("Vendredi");
semaine_cmp.addItem("Samedi");
semaine_cmp.addItem("Dimanche");
```

Il faut reconnaître que cette solution est longue, car vous devez saisir autant de lignes de code qu'il y a de lignes dans l'occurrence du composant. Dans le cas où vous auriez à remplir une occurrence avec tous les pays du monde, la tâche risquerait d'être longue. Cette méthode a le mérite d'être simple, mais elle n'est pas optimisée.

Deux autres solutions consistent à utiliser des listes. Dans le premier script ci-dessous, nous utilisons un `dataProvider`, dans le deuxième nous créons une liste avant de faire appel à une boucle `for()` pour pouvoir remplir l'occurrence du composant. Ce qui donne pour le premier script :

```
mois_cmp.dataProvider = ["Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi",
➥"Dimanche"];
```

et pour le second :

```
var listeJours:Array = ["Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi",
➥"Dimanche"];
var nbrJours:Number = listeJours.length;
for (i=0; i<nbrJours; i++) {
  semaine_cmp.addItem(listeJours[i]);
}
```

Ces deux méthodes sont effectivement plus rapides que la première, mais elles manquent encore de souplesse. Il faudrait utiliser un fichier XML pour développer avec plus de dynamisme. Nous aborderons cette technique plus loin dans notre développement.

Abordons un troisième exemple qui présente la particularité de contenir un paramètre complexe dans la méthode `addItem()`. Nous n'ajoutons pas un texte saisi entre guillemets ou un chiffre, mais deux informations.

```
var listeMois:Array = ["Janvier à Mars", "Avril à Juin", "Juillet à Septembre",
➥"Octobre à Décembre"];
var listeTrimestres:Array = ["1er trimestre", "2e trimestre", "3e trimestre",
➥"4e trimestre"];
var nbrMois:Number = listeMois.length;
for (i=0; i<nbrMois; i++) {
   mois_cmp.addItem({data:listeTrimestres[i], label:listeMois[i]});
}
```

Commençons par simplifier la ligne qui nous intéresse (la dernière) en illustrant notre propos par un autre exemple :

```
{     data:"Alpes-de-Haute-Provence"     ,     label:"Provence-Alpes-Côte d'azur"     }
```

Ces accolades contiennent deux paramètres séparés par une virgule qui sont eux-mêmes composés de deux parties séparées par le caractère deux-points (:). Sur la scène, l'occurrence du composant affichera l'information associée au mot `label`. Lorsque l'utilisateur sélectionnera cette entrée de la liste, la valeur associée à cette étiquette sera Alpes-de-Haute-Provence. Cette syntaxe permet de définir des listes dont le but est de sélectionner une entrée pour obtenir une autre information. Ouvrez l'animation composantList2.fla du dossier KTechniques pour comprendre qu'en sélectionnant une période de l'année (janvier à mars), Flash nous affiche une autre information (1er trimestre).

Figure 7-14

En cliquant sur l'une des entrées de la liste, on obtient une valeur dans le texte dynamique placé sous l'occurrence.

Combiner l'ActionScript et le XML

La seule difficulté d'utilisation d'un fichier XML relève de la connaissance nécessaire de cette technique. Consultez les premières pages de ce chapitre si vous n'avez jamais travaillé avec du XML dans Flash.

Nous avons rédigé un fichier qui contient les lignes suivantes :

```
<vocabulaire>
        <mois nom="Janvier" traduction="January"/>
        <mois nom="Février" traduction="February"/>
        <mois nom="Mars" traduction="March"/>
        <mois nom="Avril" traduction="April"/>
        <mois nom="Mai" traduction="May"/>
        <mois nom="Juin" traduction="June"/>
        <mois nom="Juillet" traduction="July"/>
        <mois nom="Août" traduction="August"/>
        <mois nom="Septembre" traduction="September"/>
        <mois nom="Octobre" traduction="October"/>
        <mois nom="Novembre" traduction="November"/>
        <mois nom="Décembre" traduction="December"/>
</vocabulaire>
```

Il s'agit d'une arborescence qui contient un nœud racine avec 12 nœuds enfants qui ne contiennent pas de valeurs, mais seulement deux attributs.

Nous avons placé sur la scène d'une nouvelle animation, un composant `List`, et nous avons nommé l'occurrence obtenue `mois_cmp`. Le script ci-après a été placé dans la fenêtre Actions après avoir sélectionné préalablement une image clé de la timeline (celle qui contient l'occurrence du composant `List`).

```
var charge:XML = new XML();
charge.ignoreWhite = true;
charge.load("composantsmois.xml");
charge.onLoad = function() {
  var nbrMois:Number = charge.firstChild.childNodes.length;
  for (i=0; i<nbrMois; i++) {
        donnee = charge.firstChild.childNodes[i].attributes.traduction;
        etiquette = charge.firstChild.childNodes[i].attributes.nom;
        mois_cmp.addItem({data:donnee, label:etiquette});
  }
};
```

Nous ne détaillerons pas les quatre première lignes relatives au XML ; en revanche, expliquons les suivantes.

Nous stockons dans une variable intitulée `nbrMois`, le nombre de nœuds contenus dans notre fichier XML.

Nous exécutons ensuite une boucle `for()` afin de répéter la ligne d'instruction qui contient la méthode `addItem()`. À la fin de cette répétition, le composant est rempli, et la publication de l'animation nous permet d'obtenir le résultat de la figure 7-12.

> **Remarque**
>
> Afin d'éviter d'obtenir une trop longue ligne d'instruction (la dernière qui contient la méthode `addItem()`), nous utilisons deux variables. C'est la raison pour laquelle la boucle `for()` contient trois lignes au lieu d'une seule.

Nous savons à présent remplir l'occurrence d'un composant `List`, mais cela ne présente pas de réel intérêt si nous n'apprenons pas à exécuter une action au moment de la sélection d'une entrée de la liste. Comment peut-on afficher `1er trimestre` dans un texte dynamique lorsque l'utilisateur clique sur la première entrée de l'occurrence `List` de la figure 7-14 ?

Gérer l'interactivité relative au clic sur l'entrée d'une occurrence de type List

En ouvrant l'animation composantList3.fla qui se trouve dans le dossier KTechniques, vous découvrirez que la fenêtre Actions ne contient pas un très grand script. Nous avons par ailleurs abordé la première partie des lignes d'instructions. Attardons-nous donc sur celles que nous n'avons pas encore analysées.

L'essentiel

Le cœur du script qui permet d'obtenir une interactivité, ne se résume malheureusement pas en une ligne d'instruction. Un minimum de quatre lignes est nécessaire.

```
var declencheur:Object = new Object();
declencheur.change = function(info) {
  traductionSelection = info.target.value;
};
mois_cmp.addEventListener("change", declencheur);
```

Avant de commencer l'analyse de ces lignes, rappelons notre objectif. Nous devons être capables d'exécuter une ligne d'instruction lors de la sélection de l'une des entrées de l'occurrence du composant de type `List`.

La figure 7-14 démontre qu'en cliquant sur l'entrée `Janvier à Mars`, `1er trimestre` s'affiche en bas de l'écran. Cela se fait grâce au script ci-avant.

Nous devons tout d'abord commencer par créer un objet auquel nous associons l'événement `change` qui contient et définit l'action à exécuter. Le mot `info` est une variable dans laquelle sont stockées des informations lorsque l'événement se produit (événement `change` dans notre exemple). Ces informations sont accessibles avec les suffixes auxquels nous faisons référence dans le tableau 7-3.

Remarque

Le mot `info` a été saisi entre les parenthèses de la fonction, mais nous aurions pu choisir n'importe quelle autre expression dès lors qu'aucun caractère accentué ou spécial n'est utilisé.

Quant à la dernière ligne, qui permet d'enclencher la surveillance d'une éventuelle sélection d'entrée de la liste, commençons par rappeler que `mois_cmp` est le nom d'occurrence du composant de type `List` qui est sur la scène. Si cette ligne n'est pas exécutée, rien ne se produira au clic sur une entrée.

Remarque

Dans le cas où vous souhaiteriez désactiver temporairement ou définitivement la surveillance des clics sur une occurrence, vous pouvez exécuter la ligne d'instruction suivante : `mois_cmp.removeEventListener("change", declencheur);`

Revenons sur la ligne d'instruction qui s'exécute au moment de la sélection. `traductionSelection` est le nom de variable du texte dynamique qui se trouve sur la scène. Nous stockons donc visiblement dans cette zone de texte sur la scène, une valeur qui correspond à celle qui a été associée à la ligne (de la `List`) sélectionnée.

```
mois_cmp.addItem({data:donnee, label:etiquette});
```

La variable `donnee` de la ligne d'instruction ci-dessus possédait une valeur au moment de l'écriture des propriétés dans l'occurrence du composant.

Si l'explication de ce dernier paragraphe vous pose problème, continuez la lecture des paragraphes suivants ; un dernier exemple reprend cette notion.

Pour l'instant, nous avons appris à exécuter une instruction au moment de la sélection de l'une des entrées de l'occurrence, mais il existe trois autres événements :

Tableau 7-3 Événements disponibles pour la gestion du composant List

Événement	Rôle
scroll	Lorsque vous faites défiler la liste.
itemRollOver	Lorsque vous survolez l'occurrence.
itemRollOut	Lorsque vous déplacez le curseur de votre souris en dehors de l'occurrence.
change	Lorsque vous cliquez sur une entrée de la liste.

À la lecture du tableau ci-dessus, nous pourrions aussi ajouter les quelques lignes ci-après, à la suite de notre script, afin d'afficher un message sur la scène au survol de l'occurrence du composant `List`.

```
var declencheur:Object = new Object();
declencheur.itemRollOver = function(info) {
  traductionSelection = "Cliquez sur un mois de la liste";
};
```

```
mois_cmp.addEventListener("itemRollOver", declencheur);
//
var declencheur:Object = new Object();
declencheur.itemRollOut = function(info) {
  traductionSelection = "";
};
mois_cmp.addEventListener("itemRollOut", declencheur);
```

Vous noterez que ces deux gestionnaires fonctionnent sur le même principe que le premier, seuls les événements changent. Voici quelques petites explications… Lorsque l'utilisateur va survoler une entrée, le texte dynamique intitulé `traductionSelection`, qui se trouve sur la scène, sera affiché (il s'agit du texte Cliquez sur un mois de la liste).

Concluons cette partie en vous présentant un tableau des différents suffixes que vous pouvez associer à la variable `info` de nos exemples ci-après. Complétons d'abord le script initial avec les lignes d'instructions suivantes :

```
tableauEtiquettes = ["Luna", "Marine", "Marjorie", "David"];
tableauValeurs = ["Juillet 2006", "Septembre 2002", "Décembre 1977", "Février 1970"];
for (i=0; i<tableauEtiquettes.length; i++) {
  mois_cmp.addItem({label:tableauEtiquettes[i], data:tableauValeurs[i]});
}
//
var declencheur:Object = new Object();
declencheur.change = function(info) {
  traductionSelection = info.target.value;
};
mois_cmp.addEventListener("change", declencheur);
```

Figure 7-15

En cliquant sur l'une des entrées de cette liste, la variable info mémorise plusieurs informations accessibles avec les attributs du tableau ci-dessous.

La colonne Résultat du tableau ci-après vous présente ce qui s'affiche dans le texte dynamique (qui se trouve sur la scène et dont le nom de variable est `traductionSelection`.), lorsque la troisième entrée est sélectionnée (cliquée) comme le montre la figure 7-15.

Tableau 7-4 Le paramètre target fait toujours référence à l'entrée sélectionnée.

Ligne d'instruction	Résultat	Commentaire
`info.target.value`	Décembre 1977	
`info.target.selectedIndex`	2	La première entrée d'une liste porte l'index 0.
`info.target.selectedItem.label`	Marjorie	
`info.target.selectedItem.data`	Décembre 1977	Il est plus simple d'utiliser target.value.

Le mot `target` n'est pas le seul suffixe que vous pouvez employer : `type` permet de connaître l'événement utilisé dans le script.

Dans l'exemple que nous venons d'aborder, nous avons utilisé des noms d'entrées que nous aurions pu trier (David, Luna, Marine et Marjorie au lieu de Luna, Marine, Marjorie, David). Voyons comment procéder.

Trier les entrées d'une occurrence de type List

Gardez toujours à l'esprit que vous spécifiez deux informations pour définir l'entrée d'une liste : son étiquette et sa valeur. Nous allons donc pouvoir trier les étiquettes ou les valeurs d'une liste. Généralement, il est plus logique de trier les étiquettes car c'est ce que nous voyons dans l'occurrence du composant.

L'essentiel

Nous pourrions résumer tout le développement de nos explications en une seule ligne.

```
classe_cmp.sortItemsBy("label", "ASC");
```

Dans notre exemple, `classe_cmp` est le nom d'occurrence du composant dans lequel nous plaçons les noms des élèves d'une classe. Le paramètre `label` précise que le tri doit porter sur les étiquettes et non les valeurs (`data`). La chaîne de caractère `ASC`, permet de préciser l'ordre de tri croissant (`ASC`) ou décroissant (`DESC`), respectivement de A à Z ou Z à A.

> **Attention**
> Vous devez respecter la casse du premier paramètre, vous ne devez donc surtout pas écrire `LABEL` ou `Label`, mais bien `label`, tout en minuscules.

Dans cet exemple, nous démontrons que le tri d'une occurrence de composant de type `List` peut encore se faire après un premier affichage de son contenu. Pour être plus précis, si nous souhaitons trier différemment les entrées d'une occurrence, nous devons commencer par la vider, puis nous la remplissons à nouveau. C'est ensuite que nous pouvons effectuer notre tri.

Figure 7-16

Le tri d'une occurrence de composant de type List se fait avec une simple ligne de code.

Voici le script de l'animation composant4.fla qui se trouve dans le dossier KTechniques.

```
var charge:XML = new XML();
charge.ignoreWhite = true;
charge.load("eleves.xml");
charge.onLoad = function() {
  nbrEleves = charge.firstChild.childNodes.length;
  trierListeNom();
};
//
function trierListeNom() {
  classe_cmp.removeAll();
  for (i=0; i<nbrEleves; i++) {
         memoNom = charge.firstChild.childNodes[i].attributes.nom;
         memoPrenom = charge.firstChild.childNodes[i].attributes.prenom;
         classe_cmp.addItem(memoNom+" "+memoPrenom);
  }
  classe_cmp.sortItemsBy("label", "ASC");
}
function trierListePrenom() {
  classe_cmp.removeAll();
  for (i=0; i<nbrEleves; i++) {
         memoNom = charge.firstChild.childNodes[i].attributes.nom;
         memoPrenom = charge.firstChild.childNodes[i].attributes.prenom;
         classe_cmp.addItem(memoPrenom+" "+memoNom);
```

```
    }
    classe_cmp.sortItemsBy("label", "ASC");
}
//
btTriPrenoms.onPress = function() {
    trierListePrenom();
};
btTriNoms.onPress = function() {
    trierListeNom();
};
```

Une simple occurrence de composant de type `List` a été placée sur la scène, puis nommée `classe_cmp`. Le script ci-avant a ensuite été saisi dans la fenêtre Actions après avoir cliqué sur l'image clé 1 de la timeline principale de l'animation.

Si vous n'êtes pas familiarisé avec la gestion du XML dans Flash, nous vous invitons à consulter le début de ce chapitre. Pour ce qui suit, nous avons crée :

- Une fonction qui permet de trier la liste dans l'ordre alphabétique des noms.
- Une fonction qui permet de trier la liste dans l'ordre alphabétique des prénoms.
- Deux gestionnaires `onPress` permettant de lancer le tri et mettre à jour l'affichage du contenu de l'occurrence du composant de type `List`.

Revenons sur les lignes d'instructions contenues dans l'une des fonctions. Vous noterez que nous commençons toujours par vider l'occurrence du composant, puis nous parcourons le contenu de l'instance XML qui se trouve en mémoire vive de l'ordinateur. Nous stockons dans deux variables intitulées `memoNom` et `memoPrenom` les propriétés `prenom` et `nom` de chaque nœud de l'arborescence XML. Nous finissons par concaténer ces deux variables afin de les ajouter comme nom d'étiquette avec la méthode `addItem()`.

Remarque

La concaténation consiste à regrouper deux chaînes de caractères et/ou de valeurs pour n'en former qu'une seule.

Le tri d'une occurrence est donc une technique très simple, mais dans notre exemple, nous souhaitons effectuer deux tris différents, c'est pourquoi notre script est un peu plus long.

Les autres méthodes de la classe List

Jusqu'à présent, nous avons appris à construire (remplir) une occurrence de composant de type `List`, nous avons trié et supprimé toutes les entrées d'une occurrence, mais il existe d'autres méthodes pour cette classe. Voici un tableau qui vous les présente toutes.

Tableau 7-5 Méthodes de la classe List

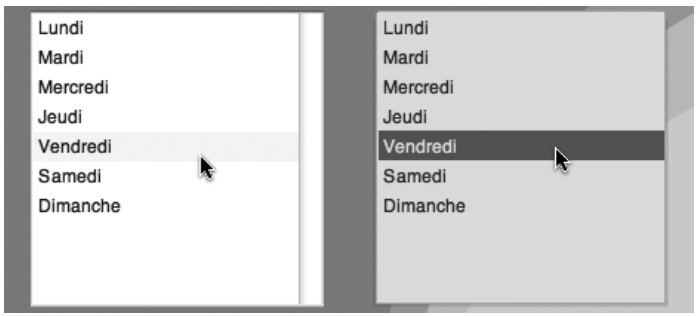
Méthodes	Rôle
`addItem()`	Ajoute une entrée à une liste (à la fin). Vous pouvez remplacer cette méthode avec la propriété `dataProvider`.
`addItemAt()`	Ajoute une entrée à une liste à un niveau (index) précis. Cela peut vous permettre d'éviter de trier si vous connaissez l'index (niveau) d'insertion.
`getItemAt()`	Permet d'obtenir l'entrée de l'index spécifié. Pour être plus précis, vous devez ajouter la propriété `label` ou `data` afin d'obtenir l'une des deux informations (par exemple, `semaine_cmp.getItemAt(2).label` pour obtenir mercredi dans l'exemple de la figure 7-17).
`removeAll()`	Vide ou supprime toutes les entrées d'une liste.
`removeItemAt()`	Supprime une entrée précise d'une occurrence à l'index spécifié.
`replaceItemAt()`	Remplace une entrée précise d'une occurrence par une autre à l'index spécifié.
`setPropertiesAt()`	Applique des propriétés à une entrée dont le numéro d'index est spécifié en paramètre de la méthode.
`sortItems()`	Trie les entrées de la liste à l'aide de la fonction de comparaison spécifiée.
`sortItemsBy()`	Trie les entrées d'une liste selon des critères specifiés en paramètres (`label` ou `data`, `ASC` ou `DESC`). Consultez l'aide en ligne de Flash en recherchant la chaîne de caractères `Array.UNIQUESORT` pour lire des informations complémentaires sur le tri des occurrences de composant de type `List`.

Mise en forme d'une occurrence de type List

Lorsque vous placez vos occurrences de composants sur la scène, leur apparence est toujours la même. Il s'agit généralement d'un cadre blanc avec du texte noir. L'ensemble est rarement en corrélation avec la charte graphique de votre interface et lorsqu'il y a un peu de couleur dans l'occurrence, elle est verte par défaut (par exemple, sélection et survol d'une entrée dans une `List`). Sur ce dernier point et avec peu de programmation vous pouvez opter pour d'autres couleurs. La copie d'écran de la figure 7-17 ne le met pas en évidence car elle est en noir et blanc, mais dans sa version originale le menu de droite est rose et marron (animation intitulée composantList.fla dans le dossier KTechniques). Dans certaines de vos animations, l'apparence standard des occurrences de vos composants ne posera pas de problèmes, dans d'autres cas, cela risque de dénoter. Imaginez en effet une interface très graphique où le moindre petit détail a été pensé et où vous placez un composant de type `List` avec son ascenseur ! Rassurez-vous, nous allons apprendre à régler l'apparence d'une occurrence de composant de type `List`.

Figure 7-17

La mise en couleur d'une occurrence de composant se fait avec quelques lignes de code relativement simples.



Important

Dans une occurrence de composant de type `List`, vous n'êtes pas obligé de garder l'ascenseur visible. En le masquant, vous proposez à l'utilisateur une zone de texte où il peut cliquer sur chaque ligne.

Vous allez découvrir dans les pages qui vont suivre qu'il est non seulement simple, mais aussi rapide de personnaliser l'apparence de ses occurrences de composant de type `List` afin qu'elles s'intègrent correctement dans la charte graphique de votre interface.

Souhaitez-vous modifier toutes les occurrences de composant de type `List` de votre animation ou bien une seule ? Nous allons aborder les deux techniques qui nous permettent de travailler sur l'ensemble des occurrences ou individuellement.

L'essentiel

Nous pourrions résumer tout le développement de nos explications en quelques lignes.

Approche globale (modifier une propriété commune à plusieurs occurrences).

```
import mx.styles.CSSStyleDeclaration;
_global.styles.List = new CSSStyleDeclaration();
_global.styles.List.setStyle("backgroundColor", "0xFFDDDD");
```

Approche ponctuelle (modifier une propriété d'une occurrence donnée).

```
mois_cmp.backgroundColor = "0xFFDDDD";
```

À la lecture de ces quelques lignes d'instructions nous constatons d'ores et déjà la simplicité avec laquelle nous pouvons mettre en forme une ou plusieurs occurrences d'un composant de type `List`. Pour être plus précis, ajoutons que vous avez la possibilité de changer un attribut particulier pour toutes les occurrences de type `List` présentes sur la scène (approche globale ci-dessus), mais vous pouvez aussi modifier l'attribut d'une seule occurrence (approche ponctuelle).

Remarque

Nous pouvons indifféremment parler de propriétés ou d'attributs pour qualifier ce qui caractérise l'apparence d'une occurrence.

Modifier une propriété commune à plusieurs occurrences

Avec la méthode globale, est-ce que la redéfinition d'une propriété s'applique à toutes les occurrences de toutes les images de l'animation ?

La réponse à cette question est oui. De ce fait, cette technique est celle que vous devez retenir pour garder une certaine cohérence avec la charte graphique de votre animation.

Pour procéder à un test, commencez tout d'abord par placer un composant de type `List` sur la scène, nommez l'occurrence obtenue (par exemple, `eleves_cmp`) et saisissez dans la fenêtre Actions, les deux lignes d'instructions suivantes :

Remarque

Remplissez tout d'abord votre occurrence avec l'une des méthodes abordées précédemment.

```
import mx.styles.CSSStyleDeclaration;
_global.styles.List = new CSSStyleDeclaration();
```

Commençons par préciser que la première des deux lignes ci-dessus permet d'étendre le vocabulaire de Flash. Le mot `import` a pour fonction d'aller rechercher des lignes de code qui se trouvent dans un fichier intitulé CSSStylesDeclaration afin de pourvoir interpréter les lignes qui suivent et font référence à la mise en forme d'un composant de type `List`.

Vous pouvez ensuite utiliser la méthode `setStyle()` pour faire référence à toutes les propriétés compatibles avec ce type d'occurrence pour effectuer le formatage correspondant à la charte graphique de votre interface. Ajoutez les trois lignes d'instructions ci-dessous pour compléter notre premier test.

```
_global.styles.List.setStyle("backgroundColor", "0xFFDDDD");
_global.styles.List.setStyle("textRollOverColor", "0xEECCCC");
_global.styles.List.setStyle("rollOverColor", "0x666666");
```

Dans le tableau ci-après, le style (qui est comparable à une propriété) doit être saisi entre guillemets comme premier paramètre de la méthode `setStyle()`. Le deuxième paramètre est une valeur généralement spécifiée dans la colonne de droite du tableau.

Attention

Respectez la casse de chaque caractère des styles sous peine d'un dysfonctionnement inéluctable (par exemple, `color` et non `Color`, `rollOverColor` et non `rolloverColor`, etc.).

Tableau 7-6 Principales propriétés/styles de la classe List

Style/Propriété	Description
`themeColor`	Couleur générique d'une occurrence. Les différentes valeurs sont : `haloGreen` (par défaut), `haloBlue` et `haloOrange`.
`alternatingRowColors`	Permet de mettre en couleurs les fonds de lignes de façon alternée. Vous devez spécifier une liste de couleurs comme dans l'exemple suivant : `_global.styles.List.setStyle("alternatingRowColors", ["0xFF6600","0xFF8D47","0xFFA97F"]);`
`backgroundColor`	Couleur d'arrière-plan de l'occurrence, par exemple `0XFF6A6A`.
`backgroundDisabledColor`	Couleur d'arrière-plan de l'occurrence lorsque sa propriété `enabled` est définie sur `false`. Par défaut un gris moyen est visible (`0xDDDDDD`).
`borderStyle`	Nous vous invitons à consulter l'aide en ligne de Flash pour découvrir les nombreuses possibilités de mise en forme du contour d'une occurrence. Cette option déborde du cadre de ce livre.
`color`	Couleur du texte.
`disabledColor`	Couleur du texte de l'occurrence lorsque sa propriété enabled est définie sur `false`. La couleur par défaut est `0x848384`. Il s'agit d'un gris foncé.
`embedFonts`	Si la propriété `fontFamily` ci-dessous fait référence à une police intégrée, vous devez régler la valeur de la propriété `embedFonts` à `true`. Cela permet d'encapsuler la police.

Tableau 7-6 Principales propriétés/styles de la classe List *(suite)*

Style/Propriété	Description
`fontFamily`	Nom d'une police.
`fontSize`	Taille du texte, en points. La valeur par défaut est 10.
`fontStyle`	Utilisez les valeurs `normal` (valeur par défaut) ou `italic`.
`fontWeight`	Utilisez les valeurs `none` ou `bold` pour définir une graisse.
`textAlign`	Utilisez les valeurs `left`, `right` ou `center` pour définir l'alignement du texte.
`textDecoration`	Utilisez les valeurs `none` ou `underline` pour gérer le surlignage d'une entrée.
`textIndent`	Propriété permettant de définir un retrait du texte par rapport au bord gauche de l'occurrence.
`rollOverColor`	Couleur d'arrière-plan d'une entrée (ligne) survolée.
`selectionColor`	Couleur d'arrière-plan d'une (entrée) ligne sélectionnée.
`textRollOverColor`	Couleur du texte lorsque le curseur de la souris passe sur une entrée. Comme le spécifie si bien la documentation de Flash, la propriété `rollOverColor` doit être bien choisie afin d'obtenir un contraste suffisament important pour lire le texte.
`textSelectedColor`	Couleur du texte de l'entrée (la ligne). Même observation que la ligne précédente avec la propriété `selectionColor`.
`useRollOver`	Régler la valeur de cette propriété sur `false` si vous ne souhaitez plus voir le changement de couleur au survol d'une entrée.

Le script ci-dessous vous propose un exemple de mise en forme d'une occurrence.

```
import mx.styles.CSSStyleDeclaration;
_global.styles.List = new CSSStyleDeclaration();
_global.styles.List.setStyle("alternatingRowColors",["0xFFDDDD", "0xFFEEEE"]);
_global.styles.List.setStyle("textRollOverColor", "0xEECCCC");
_global.styles.List.setStyle("rollOverColor", "0x666666");
_global.styles.List.setStyle("Color", "0X00FF00");
_global.styles.List.setStyle("textSelectedColor", "0XFFFFFF");
_global.styles.List.setStyle("selectionColor", "0xE89999");
```

Modifier une propriété d'une occurrence donnée

Dans la partie que nous venons d'aborder, nous avons agi sur l'ensemble des occurrences présentes dans une animation. Parfois, vous aurez besoin d'agir localement, sur une seule occurrence. Vous ne pouvez donc plus utiliser la même technique. Vous devez écrire de nouvelles lignes d'instructions qui font simplement référence à des propriétés que vous associez au nom de l'occurrence du composant de type `List`.

> **Remarque**
>
> Si vous possédez une seule occurrence de composant de type `List`, vous pouvez opter au choix pour l'une des deux méthodes (globale ou ponctuelle).

La technique qui va suivre est donc beaucoup plus simple que la première, mais elle présente tout de même l'inconvénient suivant : vous serez obligé de mettre en forme chaque occurrence séparément. Ce sont donc autant de lignes de code qu'il faudra copier-coller.

Après la méthode globale, pour procéder à un deuxième test, placez à nouveau un composant de type `List` sur la scène, nommez l'occurrence obtenue (par exemple, `semaine2_cmp`) et saisissez dans la fenêtre Actions, l'instruction suivante :

Remarque

Vous pouvez mettre en commentaires les lignes de code qui se trouvent actuellement dans votre fenêtre Actions et utiliser celle qui suit :

```
semaine2_cmp.rollOverColor = "0x996666";
```

Comme vous pouvez le constater, la technique est extrêmement simple car il vous a suffi de faire référence au nom de l'occurrence et d'y associer un nom de propriété pour changer un premier attribut. Il ne vous reste plus à présent qu'à multiplier vos lignes d'instructions pour changer d'autres propriétés. L'exemple de script ci-après vous démontre qu'une mise en forme d'occurrence de composant de type `List` s'écrit rapidement et simplement.

Remarque

La mise en forme d'une occurrence prime sur la mise en forme globale, quel que soit l'ordre d'exécution des lignes d'instructions.

```
//
// Remplissage du composant
//
semaine2_cmp.dataProvider = ["Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi",
➥"Samedi", "Dimanche"];
//
// Masquer la barre de défilement
//
semaine2_cmp.vScrollPolicy = "off";
//
// Définition des couleurs de fond
//
semaine2_cmp.backgroundColor = "0xFFDDDD";
semaine2_cmp.color = "0x663333";
//
semaine2_cmp.rollOverColor = "0x996666";
semaine2_cmp.textRollOverColor = "0xFFDDDD";
//
semaine2_cmp.selectionColor = "0x663333";
semaine2_cmp.textSelectedColor = "0xFFFFFF";
//
// Définition du style de contour
//
semaine2_cmp.setStyle("borderStyle", "default");
//none, inset, ouset, solid et
semaine2_cmp.setStyle("borderColor", "0xAA6666");
```

Vous noterez que les propriétés auxquelles il est fait référence sont celles que se trouve dans le tableau 7-6.

Gestion des lignes d'une occurrence de composant de type List

Jusqu'à présent, nous nous sommes concentrés sur la mise en forme des entrées contenues dans une occurrence de composant de type `List`, mais nous n'avons pas encore appris à manipuler les lignes qu'elle contient et gérer son enveloppe (largeur et hauteur de l'occurrence, hauteur et nombre de lignes…).

Remarque

Vous ne devez pas utiliser les propriétés `_xscale`, `_yscale`, `_width` et `_height` pour régler la largeur et la hauteur d'une occurrence de composant. Vous risqueriez de modifier l'apparence de l'ascenseur en l'anamorphosant. La méthode `setSize()` permet de procéder à la mise à l'échelle d'une occurrence tout en gardant les proportions. Les valeurs à passer en paramètres sont exprimées en pixels. Un exemple de cette méthode est :

```
eleves.setSize(200,300);
```

Si vous souhaitez effectuer l'une des opérations ci-dessous, référez-vous au tableau 7-7 pour procéder aux changements des valeurs de ces propriétés. Celui-ci n'est pas un simple copier-coller de l'aide officielle, mais une sélection des propriétés les plus importantes et/ou les plus utilisées. La description y est également redéfinie. Celle-ci est parfois difficile car abstraite, nous avons donc développé une animation intitulée composantListExo.fla dans le dossier KTechniques afin de vous aider à mieux comprendre.

Gardez à l'esprit que ces propriétés vous permettent :

- d'afficher/masquer les ascenseurs (horizontal et vertical) ;
- de définir la largeur de l'occurrence ;
- d'autoriser ou d'interdire la sélection multiple d'entrées ;
- de définir le nombre de lignes affichées ;
- de remonter ou abaisser les entrées d'une occurrence (en se passant de l'ascenseur) ;
- de connaître le nombre d'entrées contenues dans une occurrence ;
- et quelques autres commandes.

Avant de vous présenter le tableau, rappelons que les étiquettes sont les mots que vous voyez dans une occurrence. Dans la copie d'écran de la figure 7-18, les étiquettes sont : Cinéma, Informatique, Gym, Natation, Vélo et celles que nous ne voyons pas. L'entrée (que nous pourrions aussi désigner comme une ligne) qui contient le mot Musique a pour nom d'étiquette Musique et pour index la valeur 2. La première entrée de la liste porte le numéro d'index 0.

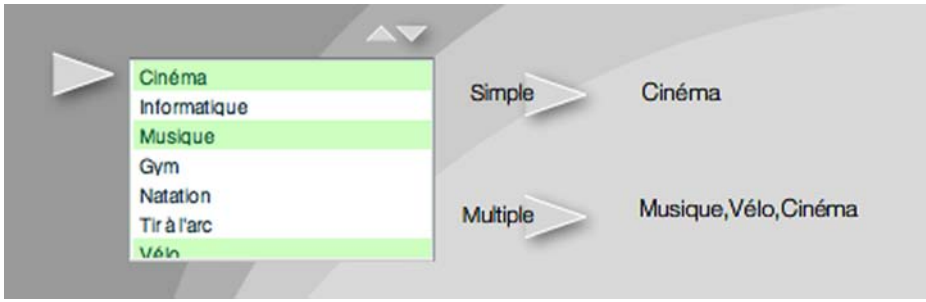


Figure 7-18

L'occurrence a été redimensionnée et l'affichage des lignes redéfini (hauteur et nombre).

Tableau 7-7 Principales propriétés de la classe List

Propriétés	Description
`List.vPosition`	Il vous arrivera parfois d'avoir plus de lignes (entrées) que ne peut en afficher l'occurrence. Vous avez donc la possibilité de spécifier, grâce à cette propriété, le numéro de ligne qui doit se retrouver en haut de l'occurrence. Cette propriété permet également de remonter ou abaisser les lignes de l'occurrence sans être obligé d'utiliser l'ascenseur. Consultez l'exemple présenté après ce tableau pour plus de details.
`List.vScrollPolicy`	Permet d'afficher ou masquer l'ascenseur vertical. Utilisez les valeurs `on` et `off` au lieu de `true` et `false`. Utilisez la propriété `vPosition` pour faire défiler les lignes de votre occurrence, comme `eleves.vScrollPolicy++`. Consultez l'exemple présenté après ce tableau pour plus de details.
`List.length`	Permet de connaître le nombre d'entrées (lignes) contenues dans l'occurrence. Attention, vous pouvez lire la valeur renvoyée par la propriété, mais vous ne pouvez pas la modifier. Utilisez la propriété `_height` pour connaître la hauteur d'une occurrence (exprimée en pixels).
`List.rowCount`	À l'inverse de la propriété précédente, vous pouvez définir le nombre de lignes visibles dans l'occurrence. Attention, si vous redéfinissez la propriété `rowHeight`, le nombre de lignes visibles ne sera pas celui que vous avez demandé. Consultez l'exemple présenté après ce tableau pour plus de details.
`List.rowHeight`	Hauteur de chaque ligne de l'occurrence (valeur exprimée en pixels). Si vous utilisez cette propriété, définissez peut-être une nouvelle taille de caractères. Consultez l'exemple présenté après ce tableau pour plus de details.
`List.selectedIndex`	Permet de connaître l'index de la ligne cliquée (et donc sélectionnée).
`List.selectedItem`	Permet d'obtenir le nom d'étiquette de l'entrée sélectionnée. Consultez l'exemple présenté après ce tableau pour plus de details.
`List.selectable`	Permet d'autoriser ou d'interdire la sélection d'une entrée dans l'occurrence.
`List.multipleSelection`	Réglée sur `true`, vous autorisez la sélection multiple dans une occurrence de composant de type `List`. Consultez l'exemple présenté après ce tableau pour plus de details.
`List.selectedIndices`	Lorsque vous ferez une sélection multiple dans une occurrence de composant de type `List`, vous pourrez obtenir les numéros de lignes sélectionnées en utilisant cette propriété. Rappelons que la première entrée d'une occurrence porte toujours le numéro 0. Consultez l'exemple présenté après ce tableau pour plus de details.
`List.selectedItems`	Contrairement à la propriété précédente, vous n'obtenez pas une liste (un tableau) des numéros d'index sélectionnés mais directement les noms d'étiquettes. Consultez l'exemple présenté après ce tableau pour plus de details.

Le script est celui de l'animation intitulée ComposantListExo qui se trouve dans le dossier KTechniques.

```
activites.dataProvider = ["Cinéma", "Informatique", "Musique", "Gym", "Natation",
➥"Tir à l'arc", "Vélo", "Marche à pied", "Randonnée", "Visite laiterie", "Poterie"];
activites.vScrollPolicy = "off";
activites.rowCount = 5;
activites.rowHeight = 15;
activites.fontSize = 10;
activites.multipleSelection = true;
//
btDepart.onPress = function() {
  activites.vPosition = 0;
};
//
btRemonter.onPress = function() {
  activites.vPosition--;
};
btAbaisser.onPress = function() {
  activites.vPosition++;
};
btAffichage.onPress = function() {
  vAffichage = activites.selectedItem;
};
btAffichageMultiple.onPress = function() {
  vAffichageMultiple = activites.selectedItems;
};
```

Et le composant ComboBox ?

Le titre de cette partie consacrée aux composants était *Apprendre à utiliser les composants List et ComboBox*. Pourtant, nous n'avons pas encore abordé ces derniers. Pourquoi a-t-on abordé uniquement des développements sur des occurrences de composant de type `List` ? Pour répondre à cette question, reprenez l'ensemble des exemples que nous avons abordés, supprimez l'occurrence du composant de type `List` que vous avez sur la scène, remplacez-la par une autre d'un composant de type `ComboBox` et renommez-la de la même façon. Vous constaterez que 90 % des lignes d'instructions fonctionnent de la même façon. Seules quelques spécificités propres à chaque occurrence diffèrent. Tournez-vous dans ce cas vers l'aide en ligne de Flash pour découvrir les quelques petites différences. Nous vous présentons déjà un tableau sur les événements de ce type d'occurrence.

Tableau 7-8 Événements disponibles pour la gestion du composant ComboBox

Événement	Rôle
`scroll`	Lorsque vous faites défiler la liste.
`itemRollOver`	Lorsque vous survolez l'occurrence.
`itemRollOut`	Lorsque vous déplacez le curseur de votre souris en dehors de l'occurrence.

Tableau 7-8 Événements disponibles pour la gestion du composant ComboBox *(suite)*

Événement	Rôle
`change`	Lorsque vous cliquez sur une entrée de la liste.
`close`	Lorsque le menu se referme.
`enter`	Lorsque l'utilisateur appuie sur la touche Entrée ou Retour de son clavier alors qu'il a saisi lui-même le nom d'une entrée dans la case supérieure de l'occurrence de type `ComboBox`.
`open`	Lorsque le menu s'ouvre.

Apprendre à gérer l'interactivité via le clavier

Si vous avez lu toutes les pages de ce chapitre depuis le début, vous avez donc ouvert l'animation xmlapprentissage2.fla qui contient un script semblable à celui que nous allons voir. Développons tout de même notre propos.

Avant tout, pourquoi une telle partie dédiée à la gestion de l'interactivité via le clavier a été insérée dans ce livre ? Pour celles et ceux qui travaillent déjà avec des logiciels de vidéo, l'usage du clavier est fréquent. Dans QuickTime une pression sur la barre espace permet par exemple de lancer et mettre en pause la lecture d'une séquence. Les touches fléchées Haut et Bas permettent d'ajuster le son, celles de Gauche et Droite de déplacer image par image la tête de lecture. Certains utilisateurs des animations Flash que vous allez réalisées vont avoir le même réflexe, à savoir contrôler le déroulement d'une vidéo au clavier : vous devez donc être capable de gérer cette interactivité.

Avant de démarrer notre apprentissage, sachez que la technique de gestion du clavier en ActionScript n'est pas simple. Plus précisément, le code que nous allons apprendre n'est pas très explicite à la première lecture. Avec des explications, vous allez très vite comprendre le sens de chaque ligne d'instruction. Il aurait été difficile de trouver la solution par vous-même si vous n'aviez pas déjà développé en objet.

Pour commencer, saisissez le script ci-après dans la fenêtre Actions d'une nouvelle animation que vous venez d'enregistrer :

```
surveil = new Object();
surveil.onKeyDown = function() {
};
Key.addListener(surveil);
```

Il s'agit du script le plus simple que vous pouvez utiliser. À ce stade, il ne propose aucune interactivité. Ajoutons-lui une ligne supplémentaire et analysons l'ensemble.

```
surveil = new Object();
surveil.onKeyDown = function() {
  gotoAndStop(10);
};
Key.addListener(surveil);
```

La première ligne de ce script a pour fonction de créer une instance de la classe `Object()` afin de pouvoir lui associer un événement `onKeyDown` à la ligne suivante. Nous aurions pu ne pas créer d'instance et remplacer le mot `surveil` de la deuxième ligne par `_root`.

La dernière ligne est celle qui enclenche réellement l'interactivité entre le clavier et l'animation. Tant qu'elle n'a pas été exécutée, toutes les lignes qui précèdent ne servent à rien. Si un jour vous aviez besoin de désactiver temporairement ou définitivement la réaction du clavier à la pression d'une touche, utilisez la ligne ci-dessous :

```
Key.removeListener();
```

Vous l'aurez compris, il serait absurde d'écrire le script ci-après, sauf dans le cas où vous souhaiteriez qu'une seule pression sur une touche du clavier soit possible.

```
surveil.onKeyUp = function() {
  Key.removeListener(surveil);
};
```

À présent, voici le script complet de l'animation clavier.fla qui se trouve dans le dossier KTechniques. Vous allez découvrir un cas intéressant, celui d'une occurrence dont le déplacement sur la scène est géré par les touches fléchées du clavier. Cette logique de développement a été retenue (utilisation du gestionnaire `onEnterFrame`), car cela permet d'obtenir un meilleur résultat. En effet, user de la répétition des touches pour reproduire l'exécution d'une ligne d'instruction n'est pas une solution à retenir dans la mesure où le réglage de ce paramètre du système d'exploitation peut varier d'une machine à l'autre.

```
var pasHorizontal:Number = 0;
var pasVertical:Number = 0;
surveil = new Object();
surveil.onKeyDown = function() {
  codeTouche = Key.getCode();
  switch (codeTouche) {
  case 37 :
         pasHorizontal = -3;
         pasVertical = 0;
         break;
  case 38 :
         pasHorizontal = 0;
         pasVertical = -3;
         break;
  case 39 :
         pasHorizontal = 3;
         pasVertical = 0;
         break;
  case 40 :
         pasHorizontal = 0;
         pasVertical = 3;
         break;
  }
```

```
  viseur.onEnterFrame = function() {
         this._x += pasHorizontal;
         this._y += pasVertical;
  };
};
surveil.onKeyUp = function() {
  delete viseur.onEnterFrame;
};
Key.addListener(surveil);
```

Dès que l'utilisateur appuie sur l'une des quatre touches fléchées du clavier, une occurrence en forme de viseur se déplace sur la scène.

Index