Partie 8

Diffuser de la vidéo à l'aide de flash

Préparer une vidéo pour l'incorporation.

Codecs et formats sont deux choses différentes.

Une vidéo d'un format donné peut être encodée et lue par des codecs différents. Codec est l'abréviation de « **co**deur **déc**odeur ».
Le format de fichier est donc une enveloppe dans laquelle on insère les données du film codées selon un cryptage donné. Cependant toutes les enveloppes ne supportent pas tous les codecs. Par exemple un fichier vidéo encodée avec sorenson spark(.mov) ne pourra pas être lue si elle est dans une enveloppe avi.
L'encodage d'une vidéo doit donc donner lieu à deux questions :
Quel est le format de sortie du contenu (quel format de fichier). Ici nous devrons le lire au travers d'un swf…

Quels sont les codecs accepté par le format qui contiendra ma vidéo ?

La vidéo dans flash : le player 9 accepte de lire les formats suivants :

- .mp4 : sera en terme de compression le plus léger. D'autres médias peuvent également lire ce format. Proche de l'avi, il peut être encodé à l'aide des codecs « divx, xvid, h264 ». Pour être lue dans flash il faudra choisir par exemple le « h264 » lors de l'encodage.
- .mov : Format vidéo utilisé par quicktime (sorenson spark).

Toutefois, préférez travailler dans les formats natifs du player flash :

- .flv : format vidéo par défaut de flash. Comporte entre autre la possibilité de diffuser des évènement durant la diffusion. Nous choisirons ce format pour le tutorial. Par défaut ces vidéo seront codées en On2 VP6 dans « adobe media encoder ».
Le son des vidéos pourra être diffusé dans le format mp3, flash supportant nativement ce format.
- F4v : format construit autour du codec (libre) h264. C'est le format qui doit à terme prendre la relève. Toutefois le système de diffusion des points de repère de ce type de vidéo à été modifié. Les évènements ne sont plus transmis de la même manière.

Pourquoi l'un plus que l'autre ? La question se pose en terme de taille du fichier et des capacités multimédias que voudrions employer (action particulière à un moment donné). Attention aussi aux mauvaises surprises sur le codec employé. Une bonne expérience est conseillée pour editer d'autres formats que le simple flv en dehors du logociels fourni dans la suite CS3 : « adobe CS3 media encoder »

Flash permet l'utilisation des codecs suivants :

Sorenson Spark : il s'agit du premier codec vidéo à être apparu dans le lecteur Flash. La qualité d'affichage n'est pas optimale, ce codec est voué à disparaître.

• Screen Video : Il s'agit du codec utilisé pour la capture d'écran par Connect (anciennement Breeze).
• On2 VP6 : ce codec fut introduit au sein du lecteur Flash 8. Il introduit une qualité d'image supérieure ainsi que la gestion du canal alpha.
• H.264 : ce codec fut introduit au sein du lecteur Flash 9.0.115. Il améliore à nouveau la qualité de l'image tout en garantissant l'interopérabilité et l'universalité des données vidéo.

(source : T. Imbert pratique de l'actionscript 3)

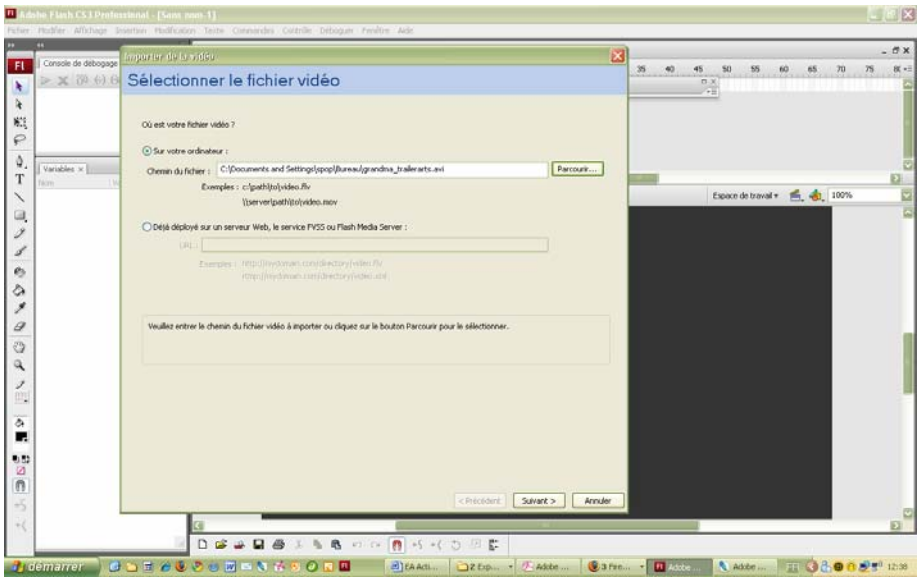
Les versions antérieures à flash cs3 ne permettaient de ne charger que des fichiers flv.

Mettre en place une vidéo dans l'environnement auteur.

Une note importante. Parce qu'une vidéo même compressée et de petite taille pèse rapidement très lourd, il n'est pas conseillé d'importer dans la bibliothèque plus de dix secondes de vidéo sans quoi on est sur de voir le swf de votre application monter bien au delà du méga, ce qui représente en terme d'ergonomie une vraie catastrophe.

Méthode 1

Dans le menu « fichier/ importer » sélectionner importer une vidéo, la boite de dialogue suivante apparait.



D'une manière générale on préfèrera importer des fichiers flv plutôt que d'autres formats. Que flash essaiera de modifier pour le rendre compatible.

Figure 1 : bien que la selection de fichier permettent un vaste choix de format, veillez à lui fournir de préférence des fichiers flv

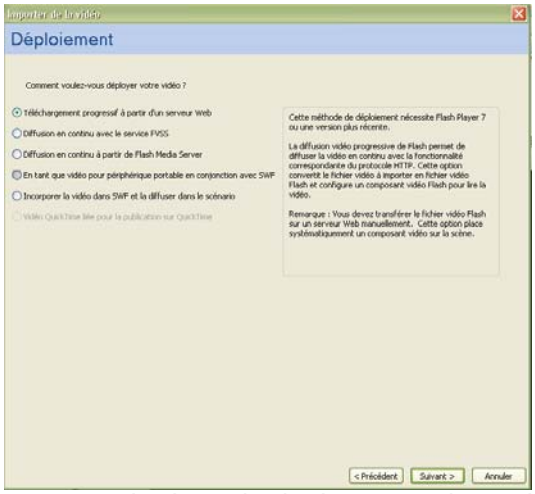


Figure 2 : le choix du déploiement dépend de la finalité du media : sert-il d'élément graphique dans l'application ou doit-il être lu et contrôlé par le biais d'un composant flvPlayback ?

Si l'on sélectionne le fichier pour l'intégrer à la bibliothèque, afin de s'en servir d'élément graphique on choisira la dernière option : « Incorporer la vidéo… ». La vidéo apparait donc dans la bibliothèque. Il suffit alors de la glisser sur la scène. Elle sera lue alors en boucle.

Si au contraire on souhaite l'intégrer et la doter de contrôle on choisira l'option « téléchargement progressif … »
Flash placera alors le fichier à la racine du répertoire où se trouve le swf et appellera un composant que vous pourrez configurer. (cf figure 3).
L'application se composera alors de votre fichier swf, du fichier flv et d'un swf comportant les contrôles. Ces trois fichiers seront nécessaires au bon fonctionnement de l'application.

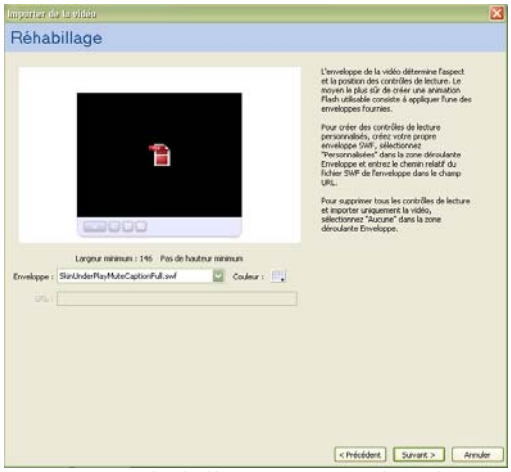


Figure 3 : L'habillage permet de sélectionner les contrôles et de modifier les couleurs de ceux-ci

Il est possible également d'importer directement le composant flvPlayback depuis le panneau des composants, de la configurer au travers de l'inspecteur des composants. Puis de le lier à une vidéo flv en renseignant le paramètre « source » dans l'inspecteur.

Le fichier flv servant d'exemple ici pèse 1,17mo, le swf incluant la vidéo 1,05Mo. Quand au swf contenant juste le composant il pèsera 49ko sans autres éléments.

Personnaliser un des habillages intégrés.
Il est possible également de vous servir de la source du composant pour personnaliser un des habillages et le conformer à la charte graphique de l'application que vous créez. Pour trouver le fichier fla du composant rendez-vous dans le dossier « program files » (windows) ou applications (« mac »).

Adobe\Adobe Flash CS4\Common\Configuration\Component Source\ActionScript 3.0\FLVPlayback

Mettre en place et contrôler une vidéo externe dans flash à l'aide du script.

Objets à étudier en particulier :

flash.net.NetConnection;
flash.net.NetStream;
flash.media.Video;

Pour des raisons évidentes de fluidité et de rendu final, certaines animations flash comportent des séquences vidéo permettant de mixer animations scriptées et rendu vidéo avec toutes les possibilités de post-production que le media permet. Cela permet des effets graphiques enrichis tout en conservant une possibilité d'interaction.
Il est donc bien utile parfois de pouvoir charger et contrôler une vidéo par le script sans pour autant avoir besoin d'une console pour l'utilisateur. Ne serait que pour presenter une video en tant que fond pour un site comme cela est actuellement courant dans nombre de sites.

Pour afficher une vidéo, un conteneur flash à besoin de réaliser les opérations suivantes :

Créer une connexion persistante avec le serveur à l'aide de l'objet NetConnection.

D'établir un flux vidéo entre l'application et le serveur pour permettre la visualisation couplé au téléchargement progressif. C'est le rôle de la classe NetStream

De créer un moteur de rendu recevant le flux et le traduisant en image et son insérable dans le liste d'affichage sous la forme de l'objet ActionScript : Vidéo.

Il est important de noter que la plupart des opérations de contrôles et de transformations de son s'effectue sur l'objet netStream soit le flux lui-même. L'objet vidéo possède quelques fonctions liées à l'objet en diffusion mais on l'utilise surtout pour les opérations graphiques commune

Utilisation de la Web Cam de l'utilisateur.

Qu'il s'agisse d'une application de chat ou d'un divertissement mettant en jeu l'image de l'utilisateur, il est utile de savoir comment accéder à la webcam de l'utilisateur. La manipulation est très simple et ne pose aucune difficulté particulière :

```
// Création d'une fenetre de diffusion video.
var fenetre : Video = new Video();

// La camera se récupère en appellant une function statique de la classe Camera
var camera:Camera = Camera.getCamera();

// Verification de la presence d'une camera.
if (camera != null)
{
	// Lissage de l'image
	fenetre.smoothing = true;
	// Transmission du flux à la fenêtre
	fenetre.attachCamera(camera);
	// Ajout à la liste d'affichage.
	addChild(fenetre);
}
else
{
	trace("pas de camera détectée");
}
```

Création d'un classe vidéo

L'utilisation de l'objet Video peut donner lieu à la création d'une classe spécialisée que nous pourrons :

- Soit utiliser simplement à des fins décoratives.
- Soit s'en servir pour une présentation synchronisée en manipulant les points de repères
- Soit la faire fonctionner de manière autonome et l'enrichir de fonctions utiles comme la diffusion multiple à l'aide d'une playlist la rendre autonome afin d'être déployée simplement sur un site. (player).

Organisation de la classe

Philosophie.

La classe que nous allons créer n'est pas aussi générique que celle que nous avons mise en place lors de la création d'un lecteur musical. Le but est ici de se familiariser avec la diffusion de contenu vidéo. Et au passage de voir comment synchroniser des actions d'un clip de la scène ou de tout autre produit de votre imagination. De nombreux enrichissements sont bien sur possibles en vous servant de cette base. Nous la rendrons toutefois assez souple pour être utilisée par toutes les possibilités offertes en terme d'instanciation (extension d'un clip de la bibliothèque, classe proposant la diffusion d'une vidéo…)

Difficultés particulière ou codes inhabituels :

La réception de certains « évènements » se fait de façon particulière en vertu du mode de diffusion de la vidéo. Les messages provenant du fichier vidéo lui-même (Métadonnées et point d'ancrage personnalisés) sont envoyés à trois fonction qui doivent porter un nom précis. (onCuePoint, onXMPData et onMetaData). Elle reçoivent toutes trois en paramètre un objet contenant les données concernées. Ces fonctions doivent nécessairement être trouvées par le flux vidéo. Elles ne necessitent pas d'être remplies. Vous pouvez simplement fournir cette fonction sans aucune instruction dans le corps.
La plus intéressante de ces fonctions est sans nul doute celle qui va recevoir les points de repères, car grâce aux points d'ancrage il est possible de synchroniser un flux vidéo et le déroulement d'une animation. Nous verrons comment construire notre classe de manière à ce qu'un évènement courant soit émis. De cette manière nous n'aurons pas à nous préoccuper des spécificités de la classe mais simplement de récupérer l'identifiant du point de repère emis par la video.
Il faut aussi indiquer au flux vidéo quelle classe contient ces fonctions. Il est par conséquent possible d'indiquer au flux d'envoyer ces informations à la classe appelante plutôt qu'à la classe FenetreVideo, ce qui représente un avantage plutôt qu'une difficulté supplémentaire. Il est possible de lisser de comportement par l'emploi d'une classe d'évènement personnalisée.

Les modifications depuis Flash CS4.

Depuis le flash player 10 quelques modifications ont été apportées au fonctionnement de la video dans flash. La première est l'ajout de la fonction de reception onXMPData aux deux autres mises en place avec l'utilisation d'AS3.

Ajout du fichier XMP

Cette fonction est déclenchée lorsque le fichier XMP inclus est reçu par le lecteur flash. Un fichier XMP est un fichier XML normalisé, qui se généralise dans la suite adobe et permet de placer des informations supplémentaires sur le fichier. Ce fichier permet selon le créateur de logiciel de faciliter la communication de données entre les différents logiciels de la suite. On retrouve ce fichier par exemple lorsque l'on travaille dans photoshop sur un fichier de type raw, toutes les modifications sont inscrites dans ce fichier, alors que les pixels de l'image ne sont pas modifiés. Lors de l'ouverture suivante les données XMP sont lues et appliquée au fichier raw.
Le fichier XMP video dans flash va contenir toutes les données concernant la vidéo, son encodage, ses points de repères… A ce jour aucune documentation n'explique clairement comment utiliser ce fichier. Il est probable qu'il prenne une importance essentielle dans les prochaines versions du player avec la modification du format de base des videos.

Vers le nouveau format F4V

Le format flv est en effet en train d'être remplacé par les fichier portant l'extension F4V. Ceux-ci sont maintenant encodés à l'aide de l'algorythme h264, qui est open source contrairement aux anciennes versions. Or ce fichier F4V ne transmet plus les points de repères de la même manière, comme nous allons le voir dans le tutoriel qui suit. Il est donc probable que la diffusion video soit profondément retravaillée afin d'unifier ces modes de diffusion avec le système évènementiel courant, et qu'a terme l'ensemble des infos transitent par le fichier XMP pour être traité directement par le lecteur flash et non plus incus dans la vidéo… A suivre de près si vos applications doivent faire appel à la vidéo avancée.

Création de la classe

La classe que nous créons étend MovieClip nous pourrons agir sur ses taille et emplacement afin de modifier l'aspect de la video que nous laisserons à sa taille par défaut.

Nous créons trois variables de classe privées qui nous servirons à manipuler le flux video :

```
private var chargeurVideo:NetConnection;
private var fluxVideo:NetStream;
private var ecranVideo:Video;
```

Par défaut le flux video à besoin de pouvoir appeler trois fonctions pour transmettre les message de la video. Il est donc obligatoire de mettre à disposition du flux trois corps de fonctions fussent-elles vides. Par défaut, le flux video ne va pas chercher ces fonctions dans le corps de la classe. Il est nécessaire de lui indiquer quel objet les renferme. Cela pourrait donc être une classe principale, ou même une classe personnalisée. Pour cela nous allons passer un paramètre dans le constructeur permettant d'indiquer l'emplacement de ces fonctions. Nous donnons une valeur par défaut : null de manière à ce que si nous ne passons pas de paramètres, le flux aille utiliser les fonctions « onXxxxXxxx » de la classe.

Nous ajoutons par conséquent une variable de classe pour stocker le containeur des fonctions. Nous assurons la mise en place dans la fonction constructeur :

```
private var client_fonctions:Object;
public function FenetreVideo( clientFonctions:Object = null)
{
	if (clientFonctions == null){ client_fonctions = this;}
	else { client_fonctions = clientFonctions };
}
```

Nous placerons dans le scénario trois fonctions différentes
La référence à ces fonctions doivent obligatoirement porter un nom précis

onCuePoint:Function
onMetaData:Function
onXMPData:Function

```
public function onMetaData ( pMeta:* ):void
{
/* Cette boucle sert de controle pour lister l'ensemble de propriétés de
l'objet reçu par la fonction */

	trace("-----------------------------------")
	trace("METADONNEES")
```

```
        for ( var p:* in pMeta )
        trace( p + " : " + pMeta[p] );
        trace("-----------------------------------")

        // mise en lecture
        jouerFlux();
}


/* cette fonction permet d'intercepter lefichier joint XMP lors de
l'encodage video du flv/f4v */
public function onXMPData ( pMeta:* ):void
{
        for ( var p:* in pMeta )
        {trace( p + " : " + pMeta[p] );}
}
```

La fonction onCuePoint permet d'intercepter les points de repères programmés lors de l'encodage du flv. Afin de ne pas rendre obligatoire le transfert de cette fonction nous allons nous en servir pour diffuser, grâce à l'héritage MovieClip de notre classe, un évènement générique à chaque fois qu'un point de repère est reçu.
Pour ce faire, nous allons ajouter une variable de classe à notre script qui stockera le nom du point de repère et sera accessible via la publicité de la variable ou par un « getter »

```
private var point_repere : String;


public function get repereActuel():String { return point_repere; }
```

En recevant notre point de repère dans la fonction nous le stockons dans la variable et diffusons un évènement générique pour prévenir de la réception.
Nous le recupèrerons en ayant placé l'ecouteur event.CHANGE sur l'instance de FenetreVideo et en appelant dans la fonction évènementielle : evenement.target.repereActuel.

```
public function onCuePoint ( pMeta:* ):void
{
        point_repere = pMeta.name;
        dispatchEvent(new Event(Event.CHANGE));

        for ( var p:* in pMeta )
        {trace( p + " : " + pMeta[p] );}
}
```

En dehors de ces figures imposées, le lancement à proprement parler de la vidéo est relativement simple. Il nous faut mettre en œuvre les objets que j'ai évoqués plus haut. Nous ne les plaçons pas dans la fonction constructeur mais dans une fonction dédiée à son déploiement, rendant la classe éligible pour une multi diffusion sans avoir à la recomposer entièrement. Cette façon de procéder permettra d'éviter de modifier la source d'un précédent swf qui utilise la même classe mais selon une architecture devenue obsolète.

```
public function chargerNouvelleVideo(adresse:String):void
{
        //Fonction de retrait des écouteurs s'ils existent (détaillé ci-
après)
        retraitEcouteurs() ;
        // instanciation de la connection
```

```
      chargeurVideo = new NetConnection();
// Si la connection s'effectue sur le même domaine on laisse l'argument à
null
      chargeurVideo.connect(null);

// instanciation du flux.
      fluxVideo = new NetStream ( chargeurVideo );
      fluxVideo.bufferTime = 10;
// Fonction d'ajout des écouteurs au flux (détaillé ci-après)
      ajoutEcouteurs();

      ecranVideo = new Video();

/* Tout comme une image on peut appliquer un lissage en cas de
redimensionnement. */
      ecranVideo.smoothing=true;
// on connecte le flux à l'écran vidéo
      ecranVideo.attachNetStream( fluxVideo );
// ajout à la liste d'affichage du clip en cours.
      addChild ( ecranVideo );
// lecture en chargement progressif du fichier vidéo flv
      fluxVideo.play (adresse);
/* mise en pause afin de synchroniser la diffusion a l'affichage des
informations */
      fluxVideo.pause();

// !! Le flux video envoie des appel de fonctions tels que onMetaData ou
onCuePoint pour les intercepter il faut donc créer les fonction
correspondantes et indiquer au flux à quel endroit sont stockées ses
fonctions */
      fluxVideo.client = client_fonctions;
      volume = 0;
      addChild ( ecranVideo );

}
```

L'objet NetStream diffuse les informations de lecture (début,fin) et les éventuelle erreurs. Pour différencier les différents messages il faut interroger la propriété code de l'objet info inclus dans les évènement de type NetStatusEvent.
J'ai donc choisi de placer deux écouteurs pour le même type d'évènement. Afin de ne pas finir avec une énorme fonction évènementielle mais aussi afin de séparer le traitement des erreurs et celui des messages de diffusion de la séquence vidéo.

```
private function ajoutEcouteurs ( ):void
{
fluxVideo.addEventListener( NetStatusEvent.NET_STATUS, gestionErreurs );
/* écoute de l'événementNetStatusEvent.NET_STATUS ecoute l'evenement de
début et de fin de diffusion de la video : me permet par exemple de mettre
en place une liste de lecture. */
fluxVideo.addEventListener( NetStatusEvent.NET_STATUS, etatVideo );
/* création de l'objet Video : le conteneur ou ecran qui va permettre de
gérer et d'interpréter le flux de données */

}
```

Toujours dans la visée de faire évoluer la classe dans le futur, je place également une fonction de retrait des écouteurs que j'activerai en cas d'erreur de chargement ou de fin de lecteur.

```
private function retraitEcouteurs (e:Event ):void
```

```
{
      if (fluxVideo.hasEventListener(NetStatusEvent.NET_STATUS))
      {
fluxVideo.removeEventListener( NetStatusEvent.NET_STATUS, gestionErreurs );
fluxVideo.removeEventListener( NetStatusEvent.NET_STATUS, etatVideo );
      }
}
```

La fonction de réception des erreurs coupe les écoutes d'évènements. Evitant ainsi de s'en préoccuper depuis la classe appelante si l'on souhaite détruire l'instance en cas d'erreur de diffusion.

```
private function gestionErreurs ( pEvt:NetStatusEvent ):void
{
      pEvt.target.removeEventListener( NetStatusEvent.NET_STATUS,
gestionErreurs );
      if ( pEvt.info.code == "NetStream.FileStructureInvalid" )
      {
            trace("fichier non compatible");
            retraitEcouteurs() ;
      }

      else if ( pEvt.info.code == "NetStream.NoSupportedTrackFound" )
      {
            trace("aucune piste trouvée");
            retraitEcouteurs() ;
      }
}
```

Dans la fonction gérant les signaux de fonctionnement normaux, nous retransmettons les évènements importants tels que les signaux de début et de fin de lecture en nous servant d'évènements génériques de la classe Event : le signal de début fera émettre à notre instance un Event.OPEN et celui de fin un Event.COMPLETE.

Je vous conseille ici de créer un évènement personnalisé qui vous offrira l'opportunité de transmettre directement une chaîne de caractère. En adaptant cette méthode à la diffusion des point de rrepère vous n'auriez plus alors qu'a poser un seul écouteur pour l'ensemble des signaux émis par une instance de cette classe.

```
private function etatVideo(even:NetStatusEvent):void
{
/* cette boucle sert de controle pour lister l'ensemble de propriétés de
l'objet reçu par la fonction */
      //trace("VIDEO")
      //trace("-----------------------------------");
      //for ( var p:* in even.info )
      //trace( p + " : " + even.info[p] );
      //trace("-----------------------------------");

/* diffusion d'un evenement de base "OPEN" si le code suivant est detecté,
une playlist pourra ecouter cet evenement et par exemple modifier
l'affichage du titre; */
      if(even.info.code=="NetStream.Play.Start")
      dispatchEvent(new Event(Event.OPEN));
/* diffusion d'un evenement de base "COMPLETE" si le code suivant est
detecté; */
      if(even.info.code=="NetStream.Play.Stop")
      dispatchEvent(new Event(Event.COMPLETE));
```

```
}
```

Contrôles sur la vidéo

Puis je donne accès à quelques fonctions basique afin de permettre la création d'un lecteur minimal, on fait appel pour les deux première à des fonctions déjà existantes dans l'objet de flux. De cette manière on ouvre la classe à la possibilité de la diffusion multiple :

```
public function jouerFlux():void
{
	fluxVideo.resume();
}

public function pauser():void
{
	fluxVideo.pause();
}
```

La fonction de volume est identique à celle utilisée dans le lecteur audio. On extrait l'objet SoundTransform du flux (NetStream), on modifie la propriété volume puis on lui réattribut.

```
public function set volume(vol:Number):void
{
//J'extraie l'objet soundTransform de l'objet video pour agir sur ses
paramètres
	var transformation:SoundTransform = fluxVideo.soundTransform;
// modification du volume (ex ici suppression du son).
		transformation.volume = 0;
// Puis réattribution à la video pour prendre en compte les modifications
		fluxVideo.soundTransform = transformation;
	}
```

La classe est maintenant prête à être instanciée et manipulée. Elle reste ouverte pour des développements ultérieurs tels que l'ajout d'une fonction de navigation dans la piste.

La classe finale

```
package
{
	import flash.display.MovieClip;
	import flash.events.Event;
	import flash.events.NetStatusEvent;
	import flash.media.Camera;
	import flash.media.Sound;
	import flash.media.SoundTransform;
	import flash.media.Video;
	import flash.net.NetStream;
	import flash.net.NetConnection;

public class FenetreVideo extends MovieClip
{
private var chargeurVideo:NetConnection;
private var fluxVideo:NetStream;
private var ecranVideo:Video;
private var client_fonctions:Object;
private var point_repere : String;

public function FenetreVideo( clientFonctions:Object = null)
```

```
{
      if (clientFonctions == null){ client_fonctions = this;}
      else { client_fonctions = clientFonctions };
}

public function get repereActuel():String { return point_repere; }

public function chargerNouvelleVideo(adresse:String):void
{
      retraitEcouteurs()
// instanciation de la connection
      chargeurVideo = new NetConnection();
// Si la connection s'effectue sur le même domaine on laisse l'argument à
null
      chargeurVideo.connect(null);

// instanciation du flux.
      fluxVideo = new NetStream ( chargeurVideo );
      fluxVideo.bufferTime = 10;
      ajoutEcouteurs();
      ecranVideo = new Video();

/* Tout comme une image on peut appliquer un lissage en cas de
redimensionnement. */
      ecranVideo.smoothing=true;
// on connecte le flux à l'écran vidéo
      ecranVideo.attachNetStream( fluxVideo );
// ajout à la liste d'affichage du clip en cours.
      addChild ( ecranVideo );
// lecture en chargement progressif du fichier vidéo flv
      fluxVideo.play (adresse);
/* mise en pause afin de synchroniser la diffusion a l'affichage des
informations */
      fluxVideo.pause();

// !! Le flux video envoie des appel de fonctions tels que onMetaData ou
onCuePoint pour les intercepter il faut donc créer les fonction
correspondantes et indiquer au flux à quel endroit sont stockées ses
fonctions */
      fluxVideo.client = client_fonctions;
      volume = 0;
      addChild ( ecranVideo );

}


private function ajoutEcouteurs ( ):void
{
fluxVideo.addEventListener( NetStatusEvent.NET_STATUS, gestionErreurs );
/* écoute de l'événementNetStatusEvent.NET_STATUS ecoute l'evenement de
début et de fin de diffusion de la video : me permet par exemple de mettre
en place une liste de lecture. */
fluxVideo.addEventListener( NetStatusEvent.NET_STATUS, etatVideo );
/* création de l'objet Video : le conteneur ou ecran qui va permettre de
gérer et d'interpréter le flux de données */

}

private function retraitEcouteurs (e:Event ):void
{
      if (fluxVideo.hasEventListener(NetStatusEvent.NET_STATUS))
```

```
	{
fluxVideo.removeEventListener( NetStatusEvent.NET_STATUS, gestionErreurs );
fluxVideo.removeEventListener( NetStatusEvent.NET_STATUS, etatVideo );
	}
}


/* cette fonction gére le debut et la fin et envoie un evenement de base à
la racine du swf pour pouvoir être eventuellement ecouté par une console
personnalisée.*/
private function etatVideo(even:NetStatusEvent):void
{
/* cette boucle sert de controle pour lister l'ensemble de propriétés de
l'objet reçu par la fonction */
	//trace("VIDEO")
	//trace("-----------------------------------");
	//for ( var p:* in even.info )
	//trace( p + " : " + even.info[p] );
	//trace("-----------------------------------");

/* diffusion d'un evenement de base "OPEN" si le code suivant est detecté,
une playlist pourra ecouter cet evenement et par exemple modifier
l'affichage du titre; */
	if(even.info.code=="NetStream.Play.Start")
	dispatchEvent(new Event(Event.OPEN));
/* diffusion d'un evenement de base "COMPLETE" si le code suivant est
detecté; */
	if(even.info.code=="NetStream.Play.Stop")
	{
		dispatchEvent(new Event(Event.COMPLETE));
		retraitEcouteurs()
}
}

public function onMetaData ( pMeta:* ):void
{
/* Cette boucle sert de controle pour lister l'ensemble de propriétés de
l'objet reçu par la fonction */

	trace("-----------------------------------")
	trace("METADONNEES")
	for ( var p:* in pMeta )
	trace( p + " : " + pMeta[p] );
	trace("-----------------------------------")

	// mise en lecture
	jouerFlux();
}

/* cette fonction permet d'intercepter les points de repères programmés
lors de l'encodage du flv. Nous plaçons le repère dans une variable
accessible et diffusons un évènement générique pour prévenir de la
réception */
public function onCuePoint ( pMeta:* ):void
{
	point_repere = pMeta.name;
	dispatchEvent(new Event(Event.CHANGE));
	for ( var p:* in pMeta )
	{trace( p + " : " + pMeta[p] );}
}
```

```
/* cette fonction permet d'intercepter lefichier joint XMP lors de
l'encodage video du flv/f4v */
public function onXMPData ( pMeta:* ):void
{
      for ( var p:* in pMeta )
      {trace( p + " : " + pMeta[p] );}
}

/* cette fonction sert à prévoir un comportement particulier et adapté en
cas d'erreur de la connexion. ex renvoi à une page html pouvant traiter le
format. */
private function gestionErreurs ( pEvt:NetStatusEvent ):void
{
      pEvt.target.removeEventListener( NetStatusEvent.NET_STATUS,
gestionErreurs );
      if ( pEvt.info.code == "NetStream.FileStructureInvalid" )
      {
            trace("fichier non compatible");
            retraitEcouteurs() ;
      }

      else if ( pEvt.info.code == "NetStream.NoSupportedTrackFound" )
      {
            trace("aucune piste trouvée");
            retraitEcouteurs() ;
      }
}

/* Ces fonctions simples me permettent de faire appel à aux methodes
qu'elles renferment depuis l'exterieur. Par exemple un bouton placé sur la
scéne var activer cette fonction en appelant cette methode. Ainsi toutes
les methodes propres à la video restent contenues à l'interieur de cet
objet au lieu d'être dispersées à travers l'animation. */
///----------------------------------------
public function jouerFlux():void
{
      fluxVideo.resume();
}

public function pauser():void
{
      fluxVideo.pause();
}

/*--Gestion du son de la video. --*/
public function set volume(vol:Number):void
{
//J'extraie l'objet soundTransform de l'objet video pour agir sur ses
paramètres
      var transformation:SoundTransform = fluxVideo.soundTransform;
// modification du volume (ex ici suppression du son).
            transformation.volume = 0;
// Puis réattribution à la video pour prendre en compte les modifications
            fluxVideo.soundTransform = transformation;
      }

}
}
```

Annexes

Source du code utilisé pour le script sur le scénario
- thibaut Imbert : pratique de l'actionScript 3

Logiciels gratuit pour la compression d'image
- Super : permet la conversion de multiples format (testé)
- free-flv converter : http://www.commentcamarche.net/telecharger/telecharger-34055397-free-flv-converter (non testé).
- Riva : http://www.commentcamarche.net/telecharger/telecharger-34055237-riva-flv-encoder (non testé).

La même … dans le panneau actions-image d'une animation de base :

```
//Script source :  Thibault Imbert "Pratique de l'actionscript 3" .
Chapitre 17 – Son et vidéo – version 0.1.1


var chargeurVideo:NetConnection = new NetConnection();
chargeurVideo.connect(null);
var fluxVideo:NetStream = new NetStream ( chargeurVideo );

// Je precise le nombre de seconde à mettre en tampon sur le flux
fluxVideo.bufferTime=10;

fluxVideo.addEventListener( NetStatusEvent.NET_STATUS, gestionErreurs );

fluxVideo.addEventListener( NetStatusEvent.NET_STATUS, etatVideo );

var ecranVideo:Video = new Video();
ecranVideo.smoothing=true;
ecranVideo.attachNetStream( fluxVideo );

function etatVideo(even:NetStatusEvent)
{

      if(even.info.code=="NetStream.Play.Start")
      stage.dispatchEvent(new Event(Event.OPEN));
      if(even.info.code=="NetStream.Play.Stop")
      stage.dispatchEvent(new Event(Event.COMPLETE));
}

addChild ( ecranVideo );

fluxVideo.play ("ADRESSE_VIDEO.FLV");
fluxVideo.pause();
fluxVideo.client = this;

var transformation:SoundTransform = fluxVideo.soundTransform;
transformation.volume = 0;
fluxVideo.soundTransform = transformation;


function onMetaData ( pMeta ):void
{
      trace("METADONNEES")
      trace("-----------------------------------")
      for ( var p in pMeta )
      trace( p + " : " + pMeta[p] );
```

```
	trace("-----------------------------------")

	ecranVideo.width = pMeta.width;
	ecranVideo.height = pMeta.height;

	if ( !contains ( ecranVideo ) )
	{
		// dans quel cas nous l'ajoutons
		addChild ( ecranVideo );
	}
	jouerFlux();

}


function onCuePoint ( pMeta ):void
{
	trace("CUE POINT nom = "+pMeta.name)
	for ( var p in pMeta )
	trace( p + " : " + pMeta[p] );

	if(pMeta.name=="pauser")
	{
		stage.dispatchEvent(new Event(Event.UNLOAD));
	}
}

function gestionErreurs ( pEvt:NetStatusEvent ):void
{
	pEvt.target.removeEventListener( NetStatusEvent.NET_STATUS,
gestionErreurs );

	if ( pEvt.info.code == "NetStream.FileStructureInvalid" )
	trace("fichier non compatible");
	else if ( pEvt.info.code == "NetStream.NoSupportedTrackFound" )
	trace("aucune piste trouvée");
}

function jouerFlux()
{
	fluxVideo.resume();
}

function pauser()
{
	fluxVideo.pause();
}
```