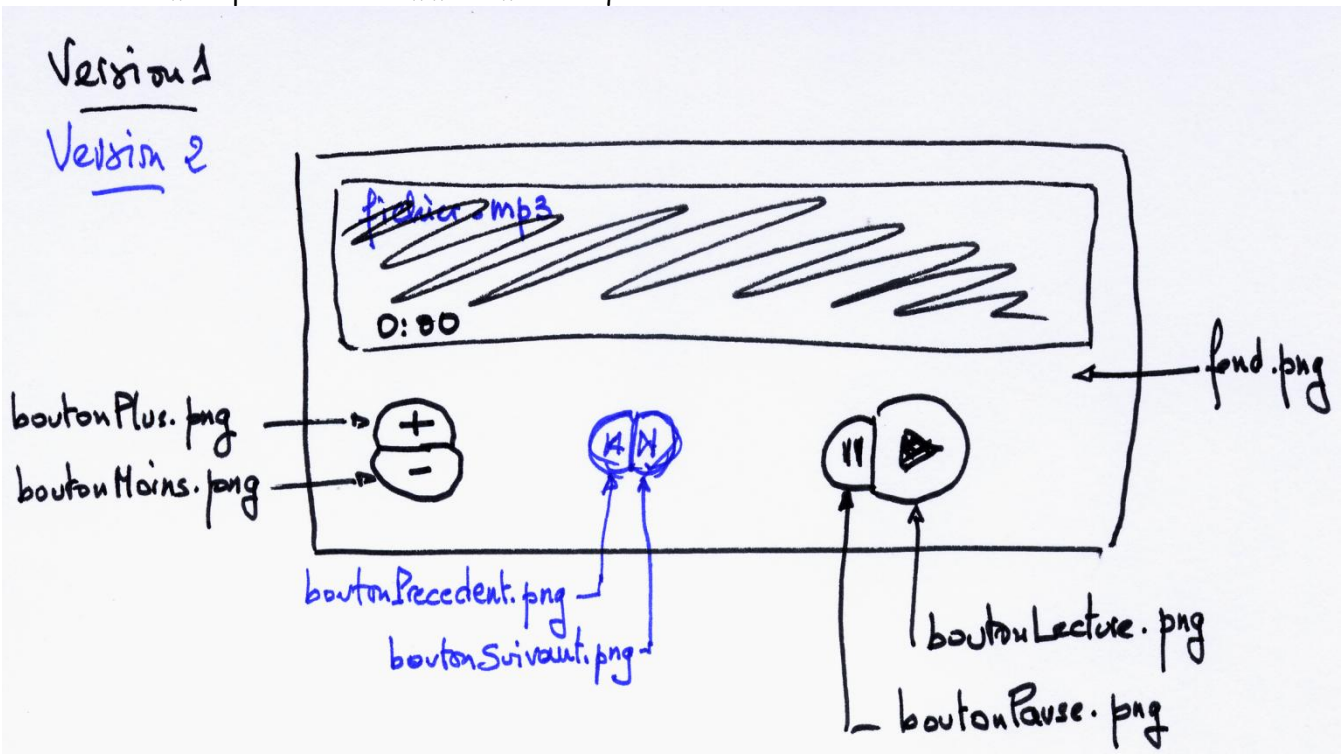
TP9 : FLASH ou HTML

Doit-on réaliser le projet avec Flash, ou utiliser les possibilités de la nouvelle version du HTML, assistée de CSS3 et de JS ?
Voici un TP qui va permettre de réaliser les 2 versions, de façon à mieux se rendre compte.
Le but est de réaliser un lecteur de fichiers audio, avec les éléments classiques de l'interface de ce type d'applications.

Habillage

L'interface sera constituée d'éléments déjà dessinés dans un traitement d'images (pas de dessins vectoriels pour ne pas favoriser Flash !).
Voici un schéma représentant sommairement ce que l'on désire obtenir :



Le nom des fichiers images est également indiqué. Tous les fichiers sont au format PNG pour pouvoir utiliser la « transparence ».
Les différents éléments prendront exactement la même place dans les deux versions (au pixel près !).
Le bouton « Lecture » permettra de démarrer la lecture du fichier MP3.
Le bouton « Pause » permettra de suspendre la lecture en cours.
Le bouton « + » permettra d'augmenter le niveau sonore.
Le bouton « - » permettra de réduire le niveau sonore.
L'affichage du temps de lecture et de la durée du fichier sera réalisé dans la fenêtre de l'écran.

En guise d'évolution, il serait souhaitable de pouvoir choisir entre plusieurs fichiers sons et de visualiser cette information. Et pourquoi ne pas changer « l'image » affichée sur l'écran ?
Et si on veut choisir soi-même les fichiers à lire ? Et si, et si …

TP9 : FLASH ou HTML

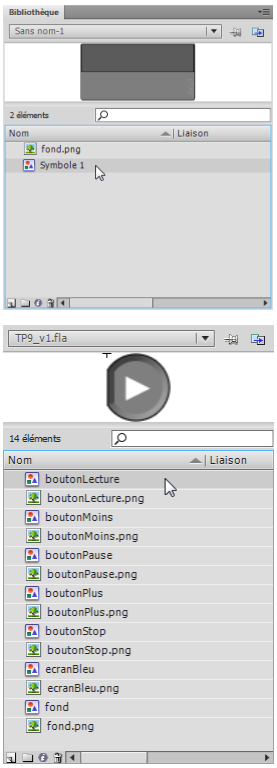
1 La version Flash

Ouvrez une nouvelle animation Flash, en AS3.
Donnez les dimensions suivantes à votre animation : largeur W=400 px et hauteur H=200 px. La couleur de fond de l'animation est indifférente, le blanc fera l'affaire, et une cadence de 12 ou 15 i/s également.
Enregistrez votre animation en TP9_v1.

1.1 Le décor de notre lecteur

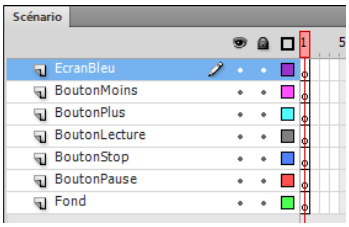
L'image du lecteur est prête, nommée fond.png, disponible dans le dossier Images de l'archive TP9.zip.
Importez cette image **dans la bibliothèque** par l'intermédiaire du menu **Fichier** .
L'image est bien chargée dans la bibliothèque, mais un autre élément est apparu, un élément de type symbole graphique comme l'indique la capture d'écran ci-contre.
Pour plus de clarté, renommez Symbole1 en fond.
Répétez ces opérations de façon à charger dans la bibliothèque les autres images nécessaires, à savoir :

- boutonLecture.png, dont le symbole graphique sera renommé en boutonLecture
- boutonPause.png, dont le symbole graphique sera renommé en boutonPause
- boutonStop.png, dont le symbole graphique sera renommé en boutonStop
- boutonPlus.png, dont le symbole graphique sera renommé en boutonPlus
- boutonMoins.png, dont le symbole graphique sera renommé en boutonMoins
- ecranBleu.png, dont le symbole graphique sera renommé en ecranBleu

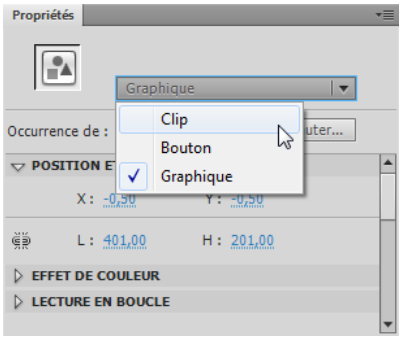


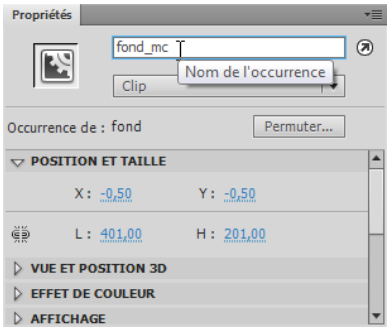
Votre bibliothèque devrait ressembler à cela :

Occupons-nous maintenant de la scène.
Renommez le Calque 1 en Fond, et ajoutez au-dessus 6 calques, nommés respectivement, de bas en haut, BoutonPause, BoutonStop, BoutonLecture, BoutonPlus, BoutonMoins et EcranBleu.
La barre de scénario s'étoffe !
Mais rien de visible …



Placez, dans l'image1 du calque Fond, une occurrence du symbole graphique fond. Centrez-le dans la scène.
Ouvrez la fenêtre des propriétés de cette occurrence et modifiez le type de Graphique à Clip, et renommez l'occurrence en fond_mc. Ce qui donne pour les propriétés de l'occurrence :



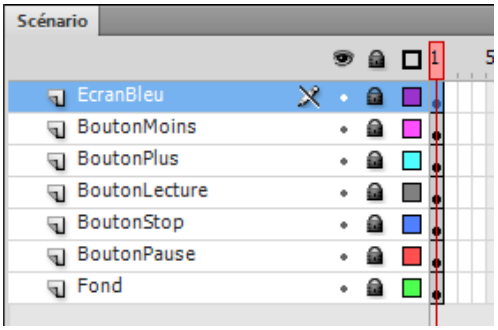


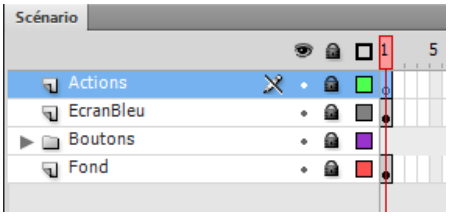
Courage, il reste 6 symboles graphiques encore inutilisés dans la bibliothèque et 6 calques vides !
Répétez les opérations précédents de façon à déposer sur chaque calque le symbole graphique correspondant avant de le transformer et de le renommer.
Ces correspondances sont fournies dans le tableau suivant.

TP9 : FLASH ou HTML

Calque	Symbole graphique	Nom occurrence Clip	Position x	Position y
BoutonPause	boutonPause	boutonPause_mc	228	139
BoutonStop	boutonStop	boutonStop_mc	228	139
BoutonLecture	boutonLecture	boutonLecture_mc	254	130
BoutonPlus	boutonPlus	boutonPlus_mc	38	129
BoutonMoins	boutonMoins	boutonMoins_mc	38	154
EcranBleu	ecranBleu	ecranBleu_mc	6	6

Travail un peu répétitif, mais qui permet d'obtenir une scène un peu plus fournie. Tous les éléments nécessaires sont maintenant en place, comme indiqué dans la fenêtre de scénario ci-contre.
Il n'y a plus qu'à saisir l'Actionscript ! Mais horreur, malheur, il me manque un calque Actions, placé en haut de la pile.
Pour les perfectionnistes, il est possible de placer tous les calques de boutons dans un dossier de calques, et surtout de verrouiller tous les calques.
Ce qui nous donnerez un truc comme ça :





1.2 Une pincée d'ActionScript

Le Flash Player est capable de lire des fichiers sons au format mp3. Il faut bien évidemment lui fournir le chemin d'accès au fichier et mettre en place quelques éléments. La saisie automatique de Flash permet d'intégrer les références aux différentes classes au fur et à mesure de leur utilisation. Pour cette application, nous aurons besoin des classes :

- flash.events.MouseEvents , pour l'utilisation de la souris sur les différents boutons ;
- flash.media.Sound , pour la manipulation de fichier son ;
- flash.media.SoundChannel , pour la manipulation de piste son ;
- flash.media.SoundTransform , pour la modification du niveau sonore des pistes.

Ces références aux classes utilisées, précédées de l'instruction import seront ajouter par Flash, automatiquement, en haut de la fenêtre script. A surveiller tout de même …

Les variables utilisées

Pour manipuler les sons dans Flash, par script, il faut utiliser des objets de type Sound (sons proprement dits, suite d'échantillons) et de type SoundChannel (piste sur laquelle un son peut être lu).
La piste va s'appeler piste1 et morceau sera le nom donné au fichier son. Ces éléments sont définis dans le corps du script de façon à être disponible pour toutes fonctions en ayant besoin. Ces « variables » sont donc de portée globale (toute l'animation).
Pour trouver le fichier son, Flash Player a besoin de connaitre le chemin d'accès et le nom de ce fichier. C'est le rôle des variables de type URLRequest. Il n'est pas nécessaire d'indiquer le chemin d'accès complet si les fichiers sont placés dans le même dossier que le fichier SWF, dans ce cas-là, seul le nom du fichier suffit.
La lecture d'une piste son s'obtient en exécutant la fonction intégrée play() d'un objet Sound, ce qui dans notre cas donne l'instruction :

- piste1 = morceau.play()

Le contenu du fichier son, associé à morceau, est transféré via le canal piste1.

TP9 : FLASH ou HTML

Si le chemin d'accès au fichier est défini à la valeur emplacement, il est facile de définir l'élément morceau par les instruction suivantes :

- var emplacement :URLRequest(" son1.mp3 ") ;
- morceau = new Sound(emplacement) ;

Mise en place des écouteurs

Pour que l'occurrence de clip boutonLecture_mc soit utilisable, il faut placer un écouteur, sur un événement souris par exemple, sur cet objet. Nous allons utiliser l'événement MOUSE_UP (bouton gauche relâché). Cet événement, lorsqu'il va survenir, lancera la fonction nommée lecture, qui sera définie ultérieurement.
Ce qui nous donne :

- Object(root).boutonLecture_mc.AddEventListener(MouseEvent.MOUSE_UP, lecture) ;

Mis en forme et en ordre, cela nous donne un premier script :

```
//***Déclarations des variables globales***//
var piste1:SoundChannel ;
var morceau:Sound ;


//***Mise en place des écouteurs***//
Object(root).boutonLecture_mc.addEventListener(MouseEvent.MOUSE_UP, lecture) ;


//***Définitions des fonctions***//
function lecture(evt:MouseEvent) {
    var emplacement:URLRequest = new URLRequest("son1.mp3");
    morceau = new Sound(emplacement) ;
    piste1 = morceau.play() ;
}
```

Placez le fichier son1.mp3 dans le même dossier que votre fichier TP9_v1.fla et testez votre animation.
Cela fonctionne, non ?
C'est juste un peu long jusqu'à la fin et un peu fort. Les autres boutons ne fonctionnent pas encore !
Pour rendre le bouton Stop opérationnel, il faut placer un écouteur capable de lancer l'arrêt de la lecture.
Soit, transcris en AS3 :

- Object(root).boutonStop_mc.AddEventListener(MouseEvent.MOUSE_UP, enStop) ;

Quant à la fonction enStop, l'instruction à utiliser est un stop() sur piste1 . Tout cela remis au propre donne les instructions suivantes :

```
//***Déclarations des variables globales***//
var piste1:SoundChannel ;
var morceau:Sound ;


//***Mise en place des écouteurs***//
Object(root).boutonLecture_mc.addEventListener(MouseEvent.MOUSE_UP, lecture) ;
Object(root).boutonStop_mc.addEventListener(MouseEvent.MOUSE_UP, enStop) ;

//***Définitions des fonctions***//
function lecture(evt:MouseEvent) {
    var emplacement:URLRequest = new URLRequest("son1.mp3");
    morceau = new Sound(emplacement) ;
    piste1 = morceau.play() ;
}

function enStop(evt:MouseEvent) {
    piste1.stop() ;
}
```

Testez votre animation, c'est mieux non ?

TP9 : FLASH ou HTML

Mais quelques améliorations s'imposent.
Pourquoi surveiller le bouton Stop tant que le fichier son n'est pas lu ? et même pourquoi ce bouton est-il actif, voire même visible ? Et si en plus on pouvait empêcher la lecture de se lancer à chaque fois que l'on utilise le bouton Play …
Pour la surveillance du bouton stop, il suffit de déplacer l'écouteur à l'intérieur de la fonction lecture.
Pour rendre le bouton Stop visiblement inactif, il suffit de lui appliquer une valeur d'alpha égale à 0.5.
Et il suffit de désactiver le bouton Play lors du premier appui, et tant que le bouton Stop n'est pas utilisé.

```
//***Déclarations des variables globales***//
var piste1:SoundChannel ;
var morceau:Sound ;

//***Apparence des objets***//
Object(root).boutonStop_mc.mouseEnabled = false ;
Object(root).boutonStop_mc.alpha = 0.5 ;

//***Mise en place des écouteurs***//
Object(root).boutonLecture_mc.addEventListener(MouseEvent.MOUSE_UP, lecture) ;

//***Définitions des fonctions***//
function lecture(evt:MouseEvent) {
    var emplacement:URLRequest = new URLRequest("son1.mp3");
    morceau = new Sound(emplacement) ;
    piste1 = morceau.play() ;
    Object(root).boutonStop_mc.addEventListener(MouseEvent.MOUSE_UP, enStop) ;
    Object(root).boutonStop_mc.mouseEnabled = true ;
    Object(root).boutonStop_mc.alpha = 1 ;
    evt.currentTarget.mouseEnabled = false ;
    evt.currentTarget.alpha = 0.5 ;
}

function enStop(evt:MouseEvent) {
    evt.currentTarget.mouseEnabled = false ;
    evt.currentTarget.alpha = 0.5 ;
    piste1.stop() ;
}
```

Testez votre animation, c'est de mieux en mieux, non ? Tiens le bouton Pause apparaît au travers du bouton Stop.
Cela se corrige facilement :

```
//***Apparence des objets***//
Object(root).boutonStop_mc.mouseEnabled = false ;
Object(root).boutonStop_mc.alpha = 0.5 ;
Object(root).boutonPause_mc.mouseEnabled = false ;
Object(root).boutonPause_mc.alpha = 0 ;
```

Et le réglage du niveau sonore ?
Voici le code AS3 permettant de faire fonctionner les boutons + et -.
Il s'agit de modifier la propriété volume de la variable niveauPiste1 de type SoundTransform. Ensuite c'est la propriété soundTransform (sans S majuscule) du SoundChannel piste1 qui est modifiée.
Un test simple est réalisé pour chaque bouton de façon à limiter le niveau sonore (0 au mini et 1.6 au max, 1.6 correspondant à une amplification acceptable).

TP9 : FLASH ou HTML

```
import flash.events.MouseEvent;
import flash.media.Sound;
import flash.media.SoundChannel;
import flash.media.SoundTransform;

//***Déclarations des variables globales***//
var piste1:SoundChannel ;
var morceau:Sound ;
var volumeSon:Number = 0.5 ;

//***Apparence des objets***//
Object(root).boutonStop_mc.mouseEnabled = false ;
Object(root).boutonStop_mc.alpha = 0.5 ;
Object(root).boutonPause_mc.mouseEnabled = false ;
Object(root).boutonPause_mc.alpha = 0 ;

//***Mise en place des écouteurs***//
Object(root).boutonLecture_mc.addEventListener(MouseEvent.MOUSE_UP, lecture) ;
Object(root).boutonPlus_mc.addEventListener(MouseEvent.MOUSE_UP, sonPlus) ;
Object(root).boutonMoins_mc.addEventListener(MouseEvent.MOUSE_UP, sonMoins) ;

//***Définitions des fonctions***//
function lecture(evt:MouseEvent) {
    var emplacement:URLRequest = new URLRequest("son1.mp3");
    morceau = new Sound(emplacement) ;
    piste1 = morceau.play() ;
    var niveauPiste1:SoundTransform = piste1.soundTransform ;
    niveauPiste1.volume = volumeSon ;
    piste1.soundTransform = niveauPiste1 ;
    Object(root).boutonStop_mc.addEventListener(MouseEvent.MOUSE_UP, enStop) ;
    Object(root).boutonStop_mc.mouseEnabled = true ;
    Object(root).boutonStop_mc.alpha = 1 ;
    evt.currentTarget.mouseEnabled = false ;
    evt.currentTarget.alpha = 0.5 ;
}

function enStop(evt:MouseEvent) {
    evt.currentTarget.mouseEnabled = false ;
    evt.currentTarget.alpha = 0.5 ;
    Object(root).boutonLecture_mc.mouseEnabled = true ;
    Object(root).boutonLecture_mc.alpha = 1 ;
    piste1.stop() ;
}

function sonPlus(evt:MouseEvent) {
    if (volumeSon<1.6) {
        volumeSon += 0.1 ;
    }
}

function sonMoins(evt:MouseEvent) {
    if (volumeSon>0) {
        volumeSon -= 0.1 ;
    }
}
```

Voila, cela fonctionne mais le bouton Pause n'est pas utilisé. Pourquoi ?
Dans l'ActionScript, il n'y a pas de méthode permettant de mettre en pause, suspendre la lecture d'un fichier son. Mais cela est assez facilement réalisable. Stocker la valeur du moment où la lecture est arrêtée est possible. Et comme il est possible de lire un fichier son à partir de l'instant choisi, le tour est joué ;-) La variable posTete, de type Number va s'en charger. Le temps écoulé pour une lecture de fichier son est exprimé en millisecondes.
Et remplacer le bouton Pause par le bouton Stop, et réciproquement n'est pas des plus compliqués.
Cela va nous donner le script suivant :

TP9 : FLASH ou HTML

```
import flash.events.MouseEvent;
import flash.media.Sound;
import flash.media.SoundChannel;
import flash.media.SoundTransform;

//***Déclarations des variables globales***//
var piste1:SoundChannel ;
var morceau:Sound ;
var volumeSon:Number = 0.5 ;
var posTete:Number = 0 ;

//***Apparence des objets***//
Object(root).boutonStop_mc.mouseEnabled = false ;
Object(root).boutonStop_mc.alpha = 0 ;
Object(root).boutonPause_mc.mouseEnabled = false ;
Object(root).boutonPause_mc.alpha = 0.5 ;

//***Mise en place des écouteurs***//
Object(root).boutonLecture_mc.addEventListener(MouseEvent.MOUSE_UP, lecture) ;
Object(root).boutonPlus_mc.addEventListener(MouseEvent.MOUSE_UP, sonPlus) ;
Object(root).boutonMoins_mc.addEventListener(MouseEvent.MOUSE_UP, sonMoins) ;

//***Définitions des fonctions***//
function lecture(evt:MouseEvent) {
    var emplacement:URLRequest = new URLRequest("son1.mp3");
    morceau = new Sound(emplacement) ;
    piste1 = morceau.play(posTete) ;
    var niveauPiste1:SoundTransform = piste1.soundTransform ;
    niveauPiste1.volume = volumeSon ;
    piste1.soundTransform = niveauPiste1 ;
    Object(root).boutonPause_mc.addEventListener(MouseEvent.MOUSE_UP, enPause) ;
    Object(root).boutonPause_mc.mouseEnabled = true ;
    Object(root).boutonPause_mc.alpha = 1 ;
    Object(root).boutonStop_mc.mouseEnabled = false ;
    Object(root).boutonStop_mc.alpha = 0 ;
    evt.currentTarget.mouseEnabled = false ;
    evt.currentTarget.alpha = 0.5 ;
}
function enPause(evt:MouseEvent) {
    posTete = piste1.position ;
    piste1.stop() ;
    evt.currentTarget.mouseEnabled = false ;
    evt.currentTarget.alpha = 0 ;
    Object(root).boutonStop_mc.mouseEnabled = true ;
    Object(root).boutonStop_mc.alpha = 1 ;
    Object(root).boutonLecture_mc.mouseEnabled = true ;
    Object(root).boutonLecture_mc.alpha = 1 ;
    Object(root).boutonStop_mc.addEventListener(MouseEvent.MOUSE_UP, enStop) ;
}

function enStop(evt:MouseEvent) {
    evt.currentTarget.mouseEnabled = false ;
    evt.currentTarget.alpha = 0 ;
    Object(root).boutonPause_mc.mouseEnabled = false ;
    Object(root).boutonPause_mc.alpha = 0.5 ;
    Object(root).boutonLecture_mc.mouseEnabled = true ;
    Object(root).boutonLecture_mc.alpha = 1 ;
    piste1.stop() ;
    posTete = 0 ;
}

function sonPlus(evt:MouseEvent) {
    if (volumeSon<1.6) {
        volumeSon += 0.1 ;
    }
}

function sonMoins(evt:MouseEvent) {
    if (volumeSon>0) {
        volumeSon -= 0.1 ;
    }
}
```

Ce serait bien de détecter la fin du fichier son, pour pouvoir réactiver le bouton Play (seuls les plus patients ont pu s'en rendre compte !)
Il faut pour cela ajouter un écouteur d'événements à la piste son, l'événement SOUND_COMPLETE est prévu pour. Cela nous donne les ajouts suivants :
Une nouvelle fonction :

```
function finPiste(evt:Event) {
    Object(root).boutonLecture_mc.mouseEnabled = true ;
    Object(root).boutonLecture_mc.alpha = 1 ;
}
```

TP9 : FLASH ou HTML

Et dans la fonction lecture :

```
function lecture(evt:MouseEvent) {
    var emplacement:URLRequest = new URLRequest("son1.mp3");
    morceau = new Sound(emplacement) ;
    piste1 = morceau.play(posTete) ;
    var niveauPiste1:SoundTransform = piste1.soundTransform ;
    niveauPiste1.volume = volumeSon ;
    piste1.soundTransform = niveauPiste1 ;
    Object(root).boutonPause_mc.addEventListener(MouseEvent.MOUSE_UP, enPause) ;
    Object(root).boutonPause_mc.mouseEnabled = true ;
    Object(root).boutonPause_mc.alpha = 1 ;
    Object(root).boutonStop_mc.mouseEnabled = false ;
    Object(root).boutonStop_mc.alpha = 0 ;
    evt.currentTarget.mouseEnabled = false ;
    evt.currentTarget.alpha = 0.5 ;
    piste1.addEventListener(Event.SOUND_COMPLETE, finPiste) ;
}
```

Y'avait pas aussi une demande d'affichage de la durée du fichier et de la lecture ?
Il va falloir retourner à nos crayons pour dessiner, sur 2 nouveaux calques, placés au-dessus du calque EcranBleu, nommés DureeMorceau et DureeLecture.
Sur le calque DureeMorceau, dessinez un bloc de texte dynamique, nommez-le dureeMorceau_txt, avec une taille de police de 30 minimum. Et sur le calque DureeLecture, pareil, mais l'occurrence du champ de texte doit être nommée dureeLecture_txt.
Un écouteur, placé sur l'occurrence fond_mc, va surveiller chaque changement d'images (lié à la cadence de l'animation). C'est l'événement ENTER_FRAME qui est fort utile !
A la fonction lecture, il suffit d'ajouter, après l'écouteur de fin de piste, l'instruction suivante :

```
    evt.currentTarget.alpha = 0.5 ;
    piste1.addEventListener(Event.SOUND_COMPLETE, finPiste) ;
    Object(root).fond_mc.addEventListener(Event.ENTER_FRAME, bougerBarre) ;
}
```

Et de définir la fonction bougerBarre comme cela :

```
function bougerBarre(evt:Event) {
    Object(root).dureeLecture_txt.text = String(Math.floor(Math.floor(piste1.position/1000)/60)) + ":" + String(Math.floor((piste1.position/1000)%60))  ;
    Object(root).dureeMorceau_txt.text = String(Math.floor(Math.floor(morceau.length/1000)/60)) + ":" + String(Math.floor((morceau.length/1000)%60))  ;
}
```

Facile pour les matheux !
Je rappelle que les durées relatives aux fichiers sons s'expriment en millisecondes. D'où les formules !
Enregistrez, testez, c'est beau, hein ?

Publiez votre animation, attention à bien déplacer votre fichier son1.mp3 avec le fichier SWF, et le fichier HTML aussi pour mettre en ligne.

Et en HTML5, ça donnerait quoi ?

TP9 : FLASH ou HTML

2 La version HTML5

HTML a beaucoup évolué, il s'appuie de plus en plus sur des éléments extérieurs, des bibliothèques entières utilisables pour faciliter la programmation en javascript , et CSS 3 pour la mise en forme rapide des éléments de la pages. A tel point qu'il ne reste plus grand-chose en HTML parfois comme nous allons le voir.

2.1 Mise en place du cadre

Il nous faut travailler sur deux fichiers, indissociables, le fichier index.html et le fichier style.css. Pour ces fichiers, un simple éditeur de texte peut faire l'affaire.
Les captures d'écran sont faites avec le Notepad++, mais ce n'est pas le plus important.
Vous devez créer un nouveau fichier, le nommer index.html et lui donner les attributs d'un fichier HTML :

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
</body>
<script>
</script>
</html>
```

à savoir une définition de type : < !DOCTYPE html> et bien évidemment le jeu de balises <html> encadrant les balises <head>, <body> et <script>.

Le fichier style.css est un simple fichier texte, sans élément caractéristique. Enregistrez ces deux fichiers dans votre espace de travail.
La référence au fichier css doit être présente dans le fichier html, dans la balise <head> pour être précis.
Et tant qu'à traiter l'entête du fichier html, autant en profiter pour lui donner un titre, et surtout pour lier la bibliothèque jQuery qui va nous servir pour la partie « interactive ».

```
<!DOCTYPE html>
<html>
<head>
    <title>Zmp3</title>
    <link rel="stylesheet" href="style.css" />
    <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
</head>
<body>
</body>
<script>
</script>
</html>
```

Le lien est fait sur la bibliothèque jQuery, version 1.8.3, dernière version disponible sur le site de Google.
Il nous faut maintenant créer une « boite » devant contenir notre lecteur mp3.
Dans la balise <body>, insérez une balise nommée scene.

```
</head>
<body>
<div id="scene">
</div>
</body>
```

Mais il faut aussi travailler dans le fichier css pour « habiller » cette boite !

Basculez dans la fenêtre d'édition de votre fichier style.css et complétez la définition de votre « scene » comme indiqué dans la capture d'écran.

```
#scene {
position : relative;
height : 200px;
width : 400px;
background-image: url('images/fond.png');
margin-left:50%;
left:-200px;
}
```

Votre « scene » sera placée au centre (horizontalement) de la fenêtre du navigateur, et l'image fond.png va servir de fond pour la boite (div) nommée scene.
Enregistrez vos deux fichiers et visualisez index.html dans votre navigateur internet préféré pour voir votre travail.

C'est bon, l'image est bien centrée, quelle que soit la taille de la fenêtre ? Donc on continue …

TP9 : FLASH ou HTML

2.2 Mise en place des éléments

Maintenant que la première boite est posée, c'est parti pour le jeu de brique !
Il nous faut une boite pour ranger les boutons de lecture, pause et arrêt, ainsi qu'une autre pour les contrôles de volume. Dans la fenêtre du fichier html, dans la balise div nommée scene, ajoutez une nouvelle balise div, nommée controleLecture, puis une autre balise div nommée controleSon. Chacune de ces boites va recevoir les éléments appropriés :
Dans la balise controleLecture, ajoutez les trois images correspondantes, boutonLecture.png, boutonPause.png et boutonStop.png, ces trois objets recevant respectivement les noms suivants : play, pause et stop.
De même pour la boite controleSon, il s'agit des images boutonPlus.png, référencée sonP et boutonMoins.png avec l'identificateur sonM.
Ce qui devrait vous donner un code proche de celui de la capture d'écran suivante :

```
9   <div id="scene">
10  <div id="controleLecture">
11          <img id="play" src="images/boutonLecture.png">
12          <img id="pause" src="images/boutonPause.png">
13          <img id="stop" src="images/boutonStop.png">
14      </div>
15      <div id="controleSon">
16          <img id="sonP" src="images/boutonPlus.png">
17          <img id="sonM" src="images/boutonMoins.png">
18      </div>
19  </div>
```

Enregistrez votre fichier puis testez-le dans votre navigateur.
Pas terrible !!! Les images sont bien chargées, mais pas au bon endroit.
Demandons à CSS de s'en charger …

Dans la fenêtre du fichier style.css, il faut définir la position et l'aspect des différents éléments que nous venons de déposer dans la page HTML.
Ajoutez les définitions de la boite de controleLecture et de celle de controleSon comme indiquées ci-desssous :

```
7   left:-200px;
8   }
9
10  #controleLecture {
11  position : absolute;
12  height : 52px;
13  width : 81px;
14  left: 226px;
15  top: 130px;
16  }
17
18  #controleSon {
19  position : absolute;
20  height : 50px;
21  width : 34px;
22  left: 38px;
23  top: 129px;
24  }
25
26
```

Enregistrez ce fichier et testez le fichier index.html dans votre navigateur (ou rafraichissez votre affichage si le navigateur est toujours ouvert).
C'est bien mieux, mais ce n'est pas encore ça ! Les boutons sont bien rangés dans leur boites, mais un peu en vrac !!!
Il va falloir ranger mieux que ça.
Et toujours dans le code CSS …

Dans le fichier style.css, à la suite, déclarez les éléments pause, play et stop et affectez-les des éléments correspondants à leur emplacement :

TP9 : FLASH ou HTML

```
23      top: 129px;
24    }
25
26  #pause{
27    position : absolute;
28    height: 35px;
29    width: 29px;
30    left: 1px;
31    top: 9px;
32    }
33  #play{
34    position : absolute;
35    height: 52px;
36    width: 48px;
37    left: 29px;
38    top: 0px;
39    }
40
41  #stop{
42    position : absolute;
43    height: 35px;
44    width: 29px;
45    left: 1px;
46    top: 9px;
47    }
48
```

Cela positionne les trois boutons correctement dans leur boite, les dimensions en pixels faisant référence de façon absolue au contenu de la boite controleLecture. Vous pouvez le vérifier en enregistrant le fichier et en visualisant le fichier index.html dans votre navigateur.
Ca progresse !
Pour les boutons de contrôle du volume, même traitement, dans la boite controleSon :

```
48
49  #sonP {
50    float:top ;
51    }
52
53  #sonM {
54    position : absolute;
55    height : 28px;
56    width : 34px;
57    left: 0px;
58    top: 27px;
59    }
60
```

La position du bontonPlus est définie juste par un top pour la propriété float. Le bouton sera donc placé avec le haut de l'image contre la limite de la boite controleSon, plus simple que de définir tous les éléments comme pour le sonM.

Testez votre travail après avoir enregistrer votre fichier. Voilà qui est beaucoup mieux.
Mais où est passé le bouton pause ?
Bien rangé dans sa boite, au même endroit que le bouton stop ! Mais en dessous de celui-ci. Une ligne peut arranger cela, la propriété z-index, qui permet de définir l'ordre de plusieurs éléments.
Ajoutez z-index : 1 pour le stop et z-index : 2 pour le pause.
Enregistrez votre fichier et testez index.html : c'est bon, le bouton pause est de nouveau visible, mais plus le bouton stop.
Si vous modifiez la propriété z-index de pause en lui affectant la valeur 0, après enregistrement et rafraichissement, c'est le bouton stop qui est de nouveau visible. Il est donc possible d'avoir 2 images superposées (ou plus) et de choisir celle qui sera visible, par modification de la propriété z-index. A retenir !
Et maintenant, jouons …

2.3 Interactivité ?

Si les boutons sont bien en place, il ne se passe pour l'instant rien, lors d'un clic souris sur ces boutons. Il va falloir maintenant faire intervenir le code JavaScript pour faire « bouger » les choses.
Pour s'assurer du bon fonctionnement de la page, il est plutôt conseiller de ne lancer les éléments du script que lorsque tous les éléments de la page html sont en place. C'est le rôle de l'instruction suivante : $(document).ready(function(){}) qui ne va autoriser la suite que lorsque le document (page html) sera prêt (tous les éléments chargés). Bien évidemment, cette instruction doit être placée à l'intérieur de la balise script.

```
19  </div>
20  </body>
21  <script>
22  $(document).ready(function(){
23  }
24  </script>
25  </html>
```

Et pour préparer l'interactivité, il faut encore créer une variable JS qui sera associé à l'objet Audio qui sera joué par notre player de MP3 :
mplayer=new Audio() (pour en savoir plus : http://www.w3.org/TR/2012/CR-html5-20121217/embedded-content-0.html#the-audio-element)

TP9 : FLASH ou HTML

Mais il ne s'agit pas de créer une variable pour un objet audio pour pouvoir le lire. Il faut également indiqué le chemin d'accès au fichier audio à charger. Et là, petit problème : quel format de fichier audio est géré par HTML5. Ce n'est en réalité pas tout à fait le nœud du problème, HTML5 est heureusement capable de gérer plusieurs formats de fichiers audio (MP3, OGG, wave) mais les navigateurs ne sont pas tous capables de lire tous ces formats. Pour des raisons évidentes, le lecteur ne va proposer que la lecture d'un fichier mp3 ou ogg, afin de couvrir l'ensemble des navigateurs et des plateformes. Il va donc falloir effectuer un test pour déterminer quel format de fichier sera utilisé. Cela se réalise avec la méthode canPlayType (pour plus d'infos : http://www.w3schools.com/tags/av_met_canplaytype.asp).

```
20  </body>
21  <script>
22  $(document).ready(function(){
23      var mplayer = new Audio();
24      if (mplayer.canPlayType('audio/mp3')!=""){
25          mplayer.src="sons/son1.mp3";
26      }
27      else {mplayer.src="sons/son1.ogg";}
28
29  })
30  </script>
31  </html>
```

Voila à quoi doit ressembler votre page HTML maintenant.
Attention au point d'exclamation avant le signe égal !
Enregistrez et testez …

Mais cela ne permet pas encore d'obtenir la lecture du fichier son.

Pour obtenir cette lecture il va falloir écrire la réponse à un événement click sur le l'objet nommé play :

```
21  <script>
22  $(document).ready(function(){
23      var mplayer = new Audio();
24      if (mplayer.canPlayType('audio/mp3')!=""){
25          mplayer.src="sons/son1.mp3";
26      }
27      else {mplayer.src="sons/son1.ogg";}
28
29      $('#play').click(function(){
30          mplayer.play();
31          return;
32      })
33  })
34  </script>
35  </html>
```

Enregistrez et testez …
Au clic sur l'image boutonLecture, la lecture du fichier audio se met en route, nickel !
Mais pour l'arrêter … il faut fermer la page, pour l'instant …
Programmons le bouton pause !

Comme pour l'objet play, c'est maintenant le tour de l'objet pause :

```
29      $('#play').click(function(){
30          mplayer.play();
31          return;
32      })
33
34      $('#pause').click(function(){
35          mplayer.pause();
36          return;
37      })
38  })
39  </script>
```

Une fois la saisie terminée, enregistrez et testez le fichier HTML.
Le bouton de lecture est toujours opérationnel, mais le bouton de pause l'est maintenant lui-aussi.
Encore un petit effort, les boutons de contrôle du volume ne sont pas actifs !
Mais avant de créer les réponses aux événements clic de ces boutons, il serait judicieux de définir le niveau sonore de la lecture à 50% par exemple, afin de pouvoir l'ajuster par la suite. La propriété volume de l'objet audio permet de réaliser ce changement. C'est cette même propriété qui sera modifiée par les boutons plus et moins de contrôle du volume. A insérer à la suite de la déclaration de l'objet mplayer.

TP9 : FLASH ou HTML

```
<script>
$(document).ready(function(){
    var mplayer = new Audio();
    mplayer.volume=0.5;

    if (mplayer.canPlayType('audio/mp3')!=""){
```

Enregistrez et testez votre page …
Le son est moitié moins fort, trop fort !

En réponse au clic sur un des boutons de réglage, le volume sera modifié de 10%, le volume pouvant varier de 0 à 1. Un test est mis en place de façon à rester dans les limites de la propriété.

```
    $('#pause').click(function(){
        mplayer.pause();
        return;
    })

    $('#sonM').click(function(){
        if (mplayer.volume-0.1>=0){
            mplayer.volume-=0.1;
        }
    })

    $('#sonP').click(function(){
        if (mplayer.volume+0.1<=1){
            mplayer.volume+=0.1;
        }
    })
})
</script>
</html>
```

Enregistrez et testez votre page …
OK, tout fonctionne !

Mais où est passé le bouton stop ? Et puis, ça fonctionne, mais ce n'est pas encore achevé. Il manque un affichage de la lecture, de la durée, pourquoi pas du titre.

Cela risque d'amener quelques modifications profondes du code. Enregistrez votre page HTML sous un autre nom, de façon à conserver la version initiale (index1.html fera l'affaire).

2.4 Améliorations

La version Flash contient un « écran »bleu pour l'affichage des infos, essayons de doter notre lecteur html des mêmes fonctionnalités.
Tout d'abord, il nous faut créer une boite pour recevoir l'image ecranBleu.png , une autre, à l'intérieur, pour contenir du texte qui sera transmis « au lecteur » et une seconde pour recevoir les informations liées au temps.
Ce qui dans le fichier index2.html nous donne :

```
    <div id="ecran">
        <img id="ecran" src="images/ecranBleu.png">
        <div id="playerInfo">
        </div>
        <div id="playerTemps">
        </div>

    </div>
```

ecranB

Bien évidemment, enregistrez votre fichier et testez-le.
L'écran bleu est bien présent, mais pas au bon endroit, c'est le rôle du fichier style.ccs qui lui aussi doit être modifié.

Dans le style.css, les objets ecran, playerInfo et playerTemps doivent être définis ainsi :

ecranB

```
#ecran {
position : absolute;
height : 95px;
width : 389px;
left: 3px;
top: 3px;
z-index:1;
}
```

```
#playerInfo {
position : absolute;
height : 26px;
width : 126px;
left: 10px;
top: 10px;
color:white;
z-index:2;
}
```

```
#playerTemps {
position : absolute;
height : 26px;
width : 126px;
left: 10px;
top: 46px;
color:white;
z-index:3;
}
```

TP9 : FLASH ou HTML

C'est bon pour l'écran, mais quid des infos ?
Cela va se passer dans le fichier index2.html … Il faut demander un affichage des données, mais quelles données ? Et où ? Dans la balise script, à la suite de la définition de la propriété volume, il suffit de créer 2 nouvelles variables, mplayerEtat, qui sera initialisée à stop et mplayerTemps, initialisée à 0.
Et dans la foulée, il faut demander l'affichage, dans l'objet nommé playerInfo, de la valeur de la variable mplayerEtat. Et même traitement pour la valeur de mplayerTemps qui sera affichée dans la boite nommée playerTemps. Ce qui en JS nous donne :

```
30  <script>
31  $(document).ready(function(){
32      var mplayer = new Audio();
33      mplayer.volume=0.5;
34      var mplayerEtat="stop";
35      var mplayerTemps="0";
36      $('#playerInfo').text(mplayerEtat);
37      $('#playerTemps').text(mplayerTemps);
38
39      if (mplayer.canPlayType('audio/mp3')!=""){
```

Enregistrez, testez …
Ecran bleu, un stop et, en dessous, un 0, blancs, OK, plutôt bon signe !
On ne va pas s'arrêter en si bon chemin ?
Il faut maintenant modifier l'état du lecteur lorsque les boutons de lecture sont utilisés et faire défiler la durée de lecture.
Il vous faut compléter le script des événements click pour les deux boutons de la façon suivante :

```
44      $('#play').click(function(){
45          mplayer.play();
46          mplayerEtat="lecture";
47          $('#playerInfo').text(mplayerEtat);
48          return;
49      })
50
51      $('#pause').click(function(){
52          mplayer.pause();
53          mplayerEtat="pause";
54          $('#playerInfo').text(mplayerEtat);
55          return;
56      })
```

Pour ce qui est de faire défiler le temps de lecture, il faut faire appel à un évènement régulier, or la méthode setInterval semble toute désignée (+ d'infos à http://www.toutjavascript.com).
En utilisant un délai de 100 ms, l'affichage sera mis à jour 10 fois par seconde. Mais il faut connaitre la durée du fichier son utilisé. La propriété duration de l'objet audio renvoie le nombre de secondes correspondant à cette durée. Une petite remise en forme et la durée peut être exprimée en minutes et secondes. De même, la propriété currentTime renvoie le nombre de secondes écoulées depuis le début de la lecture. Il serait intéressant de charger 2 fonctions de réaliser les calculs. En voici le script :

```
73      function dureeCalcul(){
74          var minutes = Math.floor(mplayer.duration / 60);
75          var seconds = Math.round(mplayer.duration - minutes * 60);
76          return minutes+' : '+seconds;
77      }
78
79      function timedCount() {
80          var minutes = Math.floor(mplayer.currentTime / 60);
81          var seconds = Math.round(mplayer.currentTime - minutes * 60);
82          $('#playerTemps').text(minutes+' : '+seconds+' // '+dureeCalcul());
83          return;
84      }
```

Bien sûr, pour pouvoir fonctionner, il faut ajouter l'appel à la fonction setInterval en début de script :

```
36      $('#playerInfo').text(mplayerEtat);
37      $('#playerTemps').text(mplayerTemps);
38      var timerAff = setInterval(timedCount,100);
39
40      if (mplayer.canPlayType('audio/mp3')!=""){
```

Enregistrez et testez votre travail.
C'est bon ? On continue …
Petite remarque, la ligne 37 est rendue obsolète par la ligne 38 (le 0 est immédiatement remplacé par 0 :0 // 0 : 0 . Elle n'est plus nécessaire. Et la variable mplayerTemps ? Aussi, car il est plus rapide de directement affecter un texte à une boite que de le stocker dans une variable puis de faire afficher la valeur de la variable. Donc exit la ligne 35 !

TP9 : FLASH ou HTML

```
<script>
$(document).ready(function(){
    var mplayer = new Audio();
    mplayer.volume=0.5;
    var mplayerEtat="stop";
    //var mplayerTemps="0";
    $('#playerInfo').text(mplayerEtat);
    //$('#playerTemps').text(mplayerTemps);
    var timerAff = setInterval(timedCount,100);

    if (mplayer.canPlayType('audio/mp3')!=""){
```

C'est pas mal, comme vous pouvez en juger en enregistrant et testant le fichier.
Un truc subsiste : le bouton stop … Pour l'instant, il ne sert pas !
Pour le faire fonctionner, un événement de type click, comme les autres, et le tout et joué !
Oui, mais que faire faire à ce bouton ?
Un bouton stop doit arrêter la lecture du fichier et ramener la tête de lecture au point de départ. L'arrêt de la lecture est obtenu avec le bouton pause. Pas de problème, mais pour mettre au départ ?
Deux solutions s'offrent à nous. La première, simple, consiste à recharger le fichier. La seconde, plus élégante, à positionner la tête au début du fichier, en mettant la propriété currentTime de l'objet audio à 0. Optons pour la seconde, ce qui nous donne si on insère le code du bouton stop après celui du bouton pause :

```
    $('#stop').click(function(){
        mplayer.pause();
        mplayerEtat="arrêt";
        $('#playerInfo').text(mplayerEtat);
        mplayer.currentTime=0 ;
        return;
    })

    $('#sonM').click(function(){
```

Enregistrez et testez votre page …
Y'a toujours pas de bouton stop visible, il fonctionne (ou pas …).
Pour faire apparaître le bouton, il faudrait le faire passer au-dessus du bouton pause ! On a déjà fait un truc comme ça !!! Dans le fichier style.css, un réglage de z-index !!! Passez à 3 la valeur de z-index du bouton stop. Enregistrez et rafraichissez votre page html. Le bouton stop est visible, et opérationnel !!!
Super, mais on a plus de bouton pause !!! Encore un petit effort …
Pour que le bouton stop soit accessible, il faut qu'il remplace le bouton pause : lors de la lecture, le bouton pause est visible, si on met en pause, c'est le bouton stop qui doit être visible. Et si on utilise le bouton lecture, c'est le bouton pause qui doit être visible. Logique ! Reste à mettre cela en musique … en js plutôt !

```
    $('#play').click(function(){
        mplayer.play();
        mplayerEtat="lecture";
        $('#playerInfo').text(mplayerEtat);
        document.getElementById("stop").style.zIndex="1";
        return;
    })

    $('#pause').click(function(){
        mplayer.pause();
        mplayerEtat="pause";
        $('#playerInfo').text(mplayerEtat);
        document.getElementById("stop").style.zIndex="3";
        return;
    })

    $('#stop').click(function(){
        mplayer.pause();
        mplayerEtat="arrêt";
        $('#playerInfo').text(mplayerEtat);
        mplayer.currentTime=0 ;
        document.getElementById("stop").style.zIndex="1";
        return;
    })
```

Attention à bien remettre la valeur de z-index à 1 dans style.css pour le bouton stop avant de tester votre page.
Tout fonctionne ? Les boutons jouent à cache-cache ? Bravo, voici votre première version HTML5 d'une application Flash.
Votre page est maintenant visible sur tous les supports et tous les navigateurs.
Eventuellement, proposer à l'utilisateur un choix pour la version à utiliser …
D'autres améliorations sont envisageables : Le titre du morceau dans l'écran, la valeur du volume sur l'écran, des effets sur les boutons …
A vous de voir jusqu'où vous voulez aller !
Des exemples à http://www.zoliv.fr/HTML5/zmp3.html