Partie 5

Maîtriser La gestion des données entrantes

L'objet loaderInfo.

L'objet loaderInfo est un objet qui va rassembler l'ensemble des informations et des classes du fichier auquel il appartient.
Bien que l'on trouve cet objet a la racine des DisplayObject, il ne concerne que l'objet root (ou stage) de l'application, ou un graphique (swf ou image) chargé extérieurement. Un Sprite ou un MovieClip créés dans le code renvoient null si l'on appelle cette propriété.
Parmi les diverses propriétés et méthodes de cet objet nous pouvons connaître entre autres :

- la version d'actionScript utilisée : loaderInfo.actionScriptVersion.
- Le contenu du fichier chargé : loaderInfo.content.
- Le framerate du clip chargé :loaderInfo.framerate
- …

Cet objet nous donne accès aussi à deux possibilités particulièrement intéressantes :

Récupération et utilisation de variables issues de la page html.

En nous servant du paramètre flashVars lors de l'intégration du swf il est possible de définir des variables qui seront transmises à l'application par le biais de l'objet loaderInfo.parameters.
Les variables placées sur la page html devront nécéssairement être des chaines. Les nombres seront donc aussi placés entre guillemets.
La démarche de récupération de ces variables est fort simple :

Dans le fichier html , Dans la balise script qui gère le lancement grace à swfObject, le paramètre flashvars définit une variable contenant l'ensemble des paramètres que nous souhaitons faire passer à flash :

```
var flashvars =
        {
                parametre_quelconque: "false",
                parametre_langage : "en"
        };
```

Nous cherchons à récupérer les variables passées à la racine de notre swf.

Nous sommes dans le fichier servant de classe de document :

```
var parametre_utilisateur :String ;

// je verifie que cette  variable se trouve bien à la racine de mon application
// à l'aide d'un test
if(root.loaderInfo.parameters.parametre_quelconque != undefined)
{
       parametre_utilisateur = root.loaderInfo.parameters.parametre_quelconque ;
       parametre_langage = root.loaderInfo.parameters.parametre_langage;

}
else
```

```
{
	// je m'assure de fournir à mon script un valeur par défaut
	// pour ne pas perturber l'animation en la testant hors navigateur.

	parametre_utilisateur = « false » ;
	parametre_langage = « false » ;

}
```

Exploiter les classes de la bibliothèque d'un swf externe.

Une des fonctions des plus intéressantes de l'objet loaderInfo, lorsque nous chargeons un swf externe, est de nous donner accès à tous les éléments de sa bibliothèque. Bien entendu, ceux qui auront été exporté pour actionScript dans le menu liaison des symboles de bibliothèque. Cela nous donne donc la possibilité de déléguer une partie conséquente du poids de notre application en chargeant les éléments graphiques et/ou polices dont nous aurons besoin depuis une bibliothèque externe. Dès lors, le poids de l'application pourra être consacré aux graphique et processus de l'intro du site et fournir à l'internaute une ouverture quasi-immédiate de la page.

C'est cette fois ci le sous-objet applicationDomain de loaderInfo que nous allons explorer, c'est a cet endroit que se trouve la bibliothéque des objets exportés, nous cherchons un clip auquel nous avons donné le nom de liaison « slider » dans le swf chargé :

- Récupération de classes de clips et autres boutons

```
// un swf contenant mes éléments de bibliothèque
var clip_charge : MovieClip
// l'objet qui sera renvoyé losque nous interrogerons la
// bibliothèque sera une classe générique qu'il nous faudra
// transtyper
var Reference : Class;
// stocke l'occurence de la classe
var instance:*;

// nom de classe des différents éléments déposés dans la
// bibliothèque
var identifiant:String;
 identifiant='helvetica'; // une police de la bibliothèque
// identifiant='slider'; // un MovieClip
// identifiant='bouton'; // un bouton

// verification de l'existence de la classe
if(root.loaderInfo.applicationDomain.hasDefinition(identifiant))
{
	// je stocke l'élément en le transtypant (as Class)
	Reference =
clip_charge.loaderInfo.applicationDomain.getDefinition(identifiant) as Class;
	trace ("création de l'instance : " + Reference);
	// je crée une instance de ma classe
	instance = new Reference();

	// les tests suivants me permettent de connaitre la
	// nature de l'instance créée
	if(instance is MovieClip)
	{
		trace("MovieClip : "+(instance is MovieClip) );
	}
	if(instance is Sprite)
	{
		trace("Sprite : "+(instance is Sprite) );
	}
	if(instance is SimpleButton)
	{
```

```
                        trace("Bouton : "+(instance is SimpleButton) );
                }
                if(instance is Font)
                {
                        trace("Font : "+(instance is Font) );
                        //trace((Font.enumerateFonts()[0].fontName))

                }
        }
        else
        {
                trace(" L'élément invoqué n'est pas répertorié en
librairie ")
        }
```

Récupération et incorporation de polices

Les polices étant intégrées sous forme de classes, on suit exactement le même processus que précédemment en donnant le nom de classe de la police.
Nous devrons simplement enregistrer la classe de la police (non son instance) à l'aide de la méthode statique de la classe Font :

```
Font.registerFont(fontechargée : Class);
```

Nous faisons au passage aussi la connaissance de la méthode statique :

```
Font.enumerateFonts(policeOrdiClient :Boolean = false) : Array
```
Renvoie un tableau contenant l'ensemble des polices disponibles dans l'application ou sur la machine client.

Puis nous modifions le code pour charger la police dans l'animation racine :

```
if(instance is Font)
{
        trace(("polices avant intégration : " + Font.enumerateFonts().length))

        // cette instruction transfère la police du swf chargé à notre application
        Font.registerFont(Reference);

        trace(("polices après intégration : " + Font.enumerateFonts().length))

        if(Font.enumerateFonts().length > 0 )
        trace((Font.enumerateFonts()[0].fontName))// nom veritable de la police

}
```

La police est désormais disponible dans l'application. Cela signifie que nous allons pouvoir utiliser des interpolations et des rotations sur nos champs texte sans risquer de voir le texte disparaître !
Il suffit pour cela d'appeler la police par son nom véritable, et de passer le paramètre embedFont à true ;

```
var texte : TextField = new TextField();
texte.htmlText = "<p>L'appli est testée avec succès</p>";
var format:TextFormat = new TextFormat();
// la police chargée doit être appelée par son nom officiel //
format.font = "HelveticaNeue LightExt";
// permet d'inclure le tracé des fontes dans le champ texte//
texte.embedFonts = true;
texte.multiline = true;
texte.wordWrap = true;
```

```
format.color = 0xDD0000;
format.size = 50;
texte.setTextFormat(format);
texte.width = 400;
texte.height = 150;
texte.autoSize = TextFieldAutoSize.RIGHT;
texte.selectable = false;
texte.rotation = 45;
texte.x = 200;
this.addChild(texte);
```