# 10 La 3D et *PaperVision*

## Introduction

Si nous pouvons créer des interfaces 3D à partir d'objets 2D, dans Flash, il n'est pas encore possible d'y importer des objets 3D réels et d'interagir nativement avec eux. Cependant, des classes ActionScript libres sont disponibles pour la gestion de la 3D réelle au sein de l'environnement Flash. Certaines – comme `PaperVision` – sont compatibles avec d'anciennes versions de Flash, pour AS3 et AS2 et Flash 9 (et les versions suivantes).

Dans ce chapitre, nous allons voir comment installer `PaperVision` pour AS3 sous Windows et Mac OS X, ainsi que la manière d'interagir sur des animations d'objets 3D réels exportés depuis l'application libre Google Sketchup ou tout autre logiciel 3D. Nous allons également découvrir comment contrôler des environnements 3D, créés à partir de simples primitives avec le clavier pour simuler la navigation dans un espace en trois dimensions.

À l'issue de ce chapitre, vous serez en mesure de créer des visionneuses d'objets 3D et des galeries 3D.

## Installer *PaperVision*

La classe `PaperVision` rassemble une série de fonctionnalités 3D, regroupées dans un même répertoire. Pour en disposer, nous devons d'abord télécharger cette classe. Une fois chargée sur votre ordinateur, elle doit être copiée dans le répertoire de votre site, au même niveau que les documents Flash qui constituent votre projet. Si, pour automatiser le processus, nous souhaitons centraliser la gestion des classes, nous devons redéfinir les préférences de Flash. Nous détaillons cette procédure plus loin dans ce chapitre.

Une fois dans Flash, pour invoquer les sous-classes `PaperVision`, qu'elles aient été placées localement ou intégrées à l'API, nous y faisons référence directement depuis la fenêtre des Actions. À la compilation, Flash intègre les scripts utilisés, requis pour la gestion des commandes 3D. Une fois déployées, elles n'ont donc pas besoin d'être placées sur le serveur. Seuls les objets 3D et les textures éventuellement appelées par le code doivent accompagner le document SWF.

Dans ce chapitre, nous développons les commandes 3D de `PaperVision` directement dans le scénario. De nombreux développeurs adoptent la méthode externalisée sous la forme de *packages* (classes `.as`). Nous verrons, dans ce chapitre, qu'il n'est pas nécessaire de recourir à une externalisation du code pour exécuter `PaperVision`.

La classe `PaperVision` étant constituée de nombreux fichiers et de sous-dossiers, vous trouverez plus facile d'accéder à une version zippée de cette classe. Vous en trouverez une sur le site Googlecode à l'adresse : **http://code.google.com/p/papervision3d/downloads/list**.

Vous pouvez aussi la télécharger en suivant directement le lien qui correspond à la dernière version publiée au moment où nous éditons cet ouvrage : **http://papervision3d.google-code.com/files/Papervision3D_2.1.920.zip**.

Bien que cette méthode de chargement soit simple, de nombreux utilisateurs préfèrent gérer le téléchargement des classes à l'aide d'utilitaires capables d'en assurer la mise à jour, lorsque la source distante a été modifiée par exemple.

Nous vous proposons, pour répondre à de nombreuses questions en rapport avec cette technique de chargement, de présenter ces utilitaires dans les deux sections suivantes (Tortoise SVN pour Windows et SVNX pour Mac OS X). Pour les lecteurs qui auront téléchargé `PaperVision` à partir du format zippé, vous pouvez passer directement à la section "Intégrer la classe `PaperVision` à Flash".

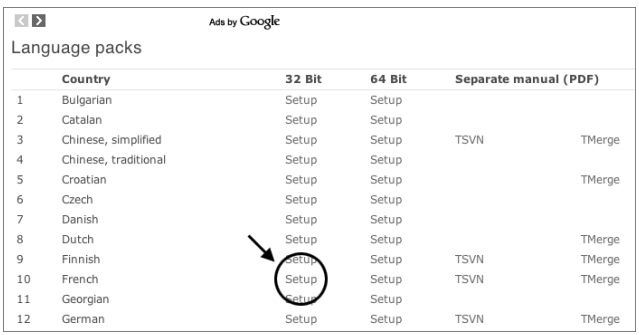
## Téléchargement avec Tortoise SVN pour Windows

L'utilitaire de chargement de dossiers nommé Tortoise SVN, compatible Windows, est disponible gratuitement à l'adresse suivante : **http://tortoisesvn.tigris.org/**.

Pour installer cet utilitaire, procédez comme suit :

1. Sur le site **http://tortoisesvn.tigris.org/**, cliquez directement sur l'onglet Downloads. En bas de la page de chargement, apparaît un tableau affichant toutes les versions disponibles, filtrées par langue et version de système (32 bits ou 64 bits)

**Figure 10.1**
*Téléchargement de l'utilitaire Tortoise SVN pour Windows.*



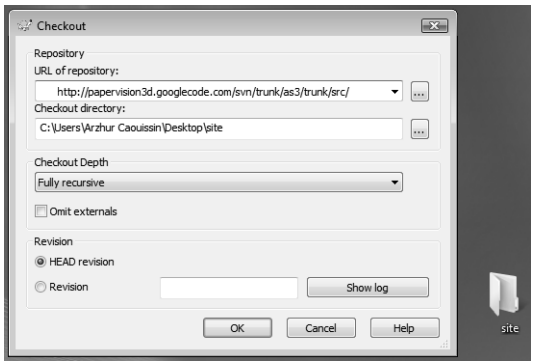

2. Cliquez sur le lien Setup correspondant à votre version de système. Le chargement s'effectue. Des supports au format PDF sont également disponibles en français : Separate manual (PDF). Vous pouvez, si besoin, les télécharger.
3. Installez le logiciel en double-cliquant sur l'icône d'installation chargée sur votre poste de travail. Puis validez toutes les étapes.

   Une fois installé, vous le retrouvez dans le dossier Program files et dans le menu Démarrer > Applications. Le principe de l'utilitaire Tortoise SVN est de rendre ses fonctions accessibles par un simple clic-droit sur le répertoire de votre choix. Vous pouvez alors y importer un dossier distant à partir d'une URL, mettre à jour le contenu éventuellement déjà chargé :
4. Directement, sur le dossier racine de votre projet, faites clic-droit > SVN CheckOut.

5. Dans la boîte de dialogue, saisissez l'URL de la classe `PaperVision` – ou d'une autre classe à importer : **http://papervision3d.googlecode.com/svn/trunk/as3/trunk/src/** (voir Figure 10.2).
6. Puis cliquez sur OK.

**Figure 10.2**

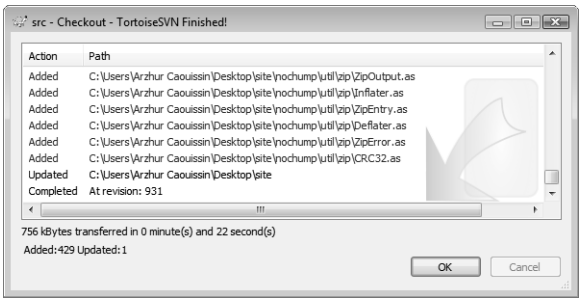
*Importer la classe avec TortoiseSVN sous Windows.*




7. Une fenêtre de chargement s'affiche. Patientez. Une fois le chargement terminé, cliquez à nouveau sur OK (voir Figure 10.3).

**Figure 10.3**

*Chargement de la classe `PaperVision`.*




8. À l'intérieur de votre dossier, la classe `PaperVision` a été importée. Elle apparaît sous la forme de deux répertoires : "nochump" et "org" (voir Figure 10.4).

**Figure 10.4**

*Classe chargée.*

La classe est disponible dans votre projet. Vous pouvez l'invoquer directement depuis votre document Flash.

## Téléchargement avec SVNX pour Mac OS X

L'utilitaire de chargement de dossiers, nommé SVNX et compatible Mac OS X, requiert deux installations : le plug-in SCPlugin et l'utilitaire SVNX.

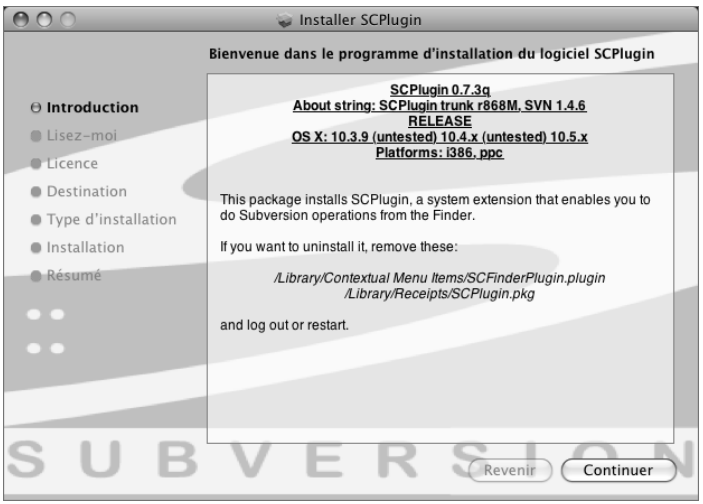
Le plug-in SCPlugin est disponible à cette adresse : **http://scplugin.tigris.org/servlets/ProjectDocumentList**. L'utilitaire SVNX ici : **http://www.lachoseinteractive.net/en/community/subversion/svnx/download/**.

1. Pour installer d'abord le plug-in, saisissez dans votre navigateur l'adresse : **http://scplugin.tigris.org/servlets/ProjectDocumentList**.
2. Puis, cliquez sur la dernière version de l'installeur proposée dans la liste de téléchargement (voir Figure 10.5).

**Figure 10.5**
*Téléchargement du plug-in SCPlugin pour Mac OS X.*

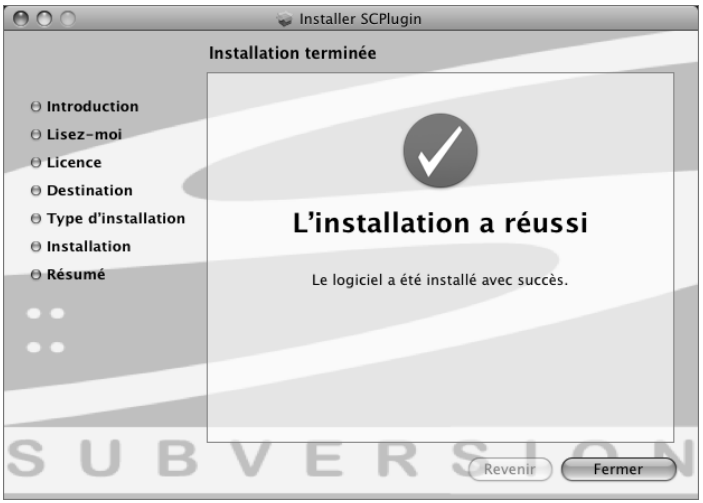

3. Le *package* téléchargé ouvre une fenêtre d'installation (voir Figure 10.6).

**Figure 10.6**
*Installation du plug-in SCPlugin pour Mac OS X.*



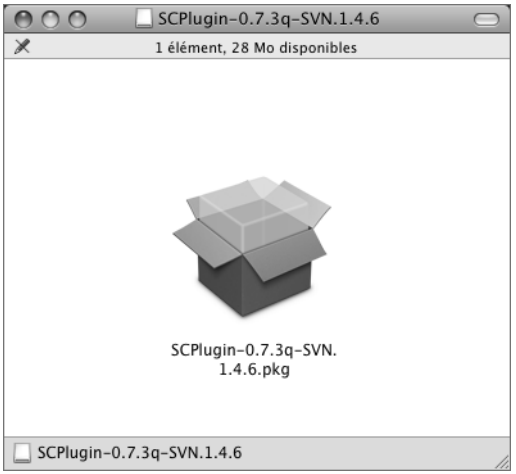

5. Validez toutes les étapes jusqu'à l'écran final (voir Figure 10.7).

**Figure 10.7**
Confirmation de l'installation.




Le dossier du plug-in apparaît sur votre bureau (voir Figure 10.8).

**Figure 10.8**
Plug-in installé.

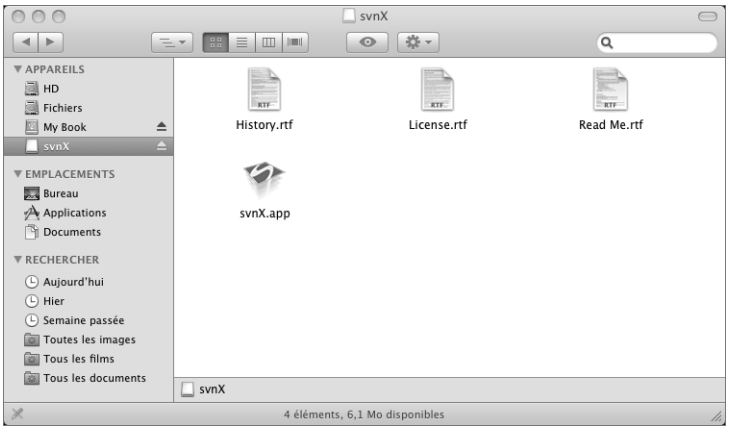



Vous pouvez maintenant installer l'utilitaire SVNX disponible à l'adresse suivante : **http://www.lachoseinteractive.net/en/community/subversion/svnx/download/**.

6. Dans la page de chargement, cliquez directement sur l'icône située en bas de page pour le chargement de l'application (voir Figure 10.9).
7. Une fois le chargement effectué, glisser-déposez directement l'application du répertoire, ouvert automatiquement, vers le dossier Applications de votre système. Pour organiser les fichiers, déposez-les dans un nouveau répertoire que vous nommerez SVNX (voir Figure 10.10).

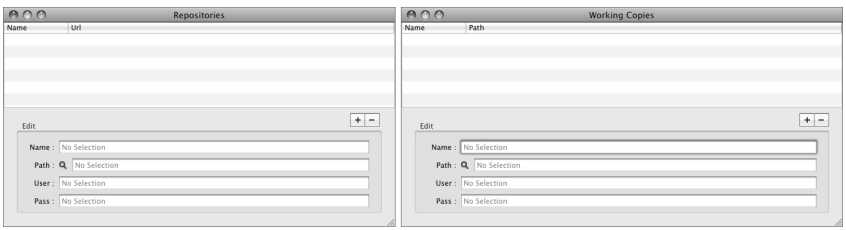
**Figure 10.9**
Téléchargement de l'utilitaire SVNX pour Mac OS X.


**Figure 10.10**
Téléchargement de l'utilitaire SVNX pour Mac OS X.




Une fois copié dans Applications, l'utilitaire est fonctionnel. Pour charger la classe `PaperVision`, *via* l'utilitaire SVNX : lancez l'utilitaire SVNX en double-cliquant dessus. Deux fenêtres apparaissent (voir Figure 10.11).

**Figure 10.11**
Ouverture de l'utilitaire SVNX.

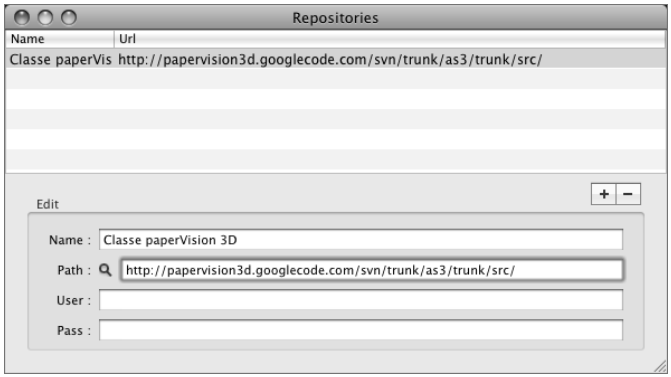



Seule la première fenêtre nous intéresse. Elle désigne l'URL à télécharger :

1. Pour définir un nouveau chargement, dans la fenêtre gauche, cliquez sur le bouton Plus.
2. Puis, renseignez un Nom (champ Name). Saisissez par exemple PaperVision3D.

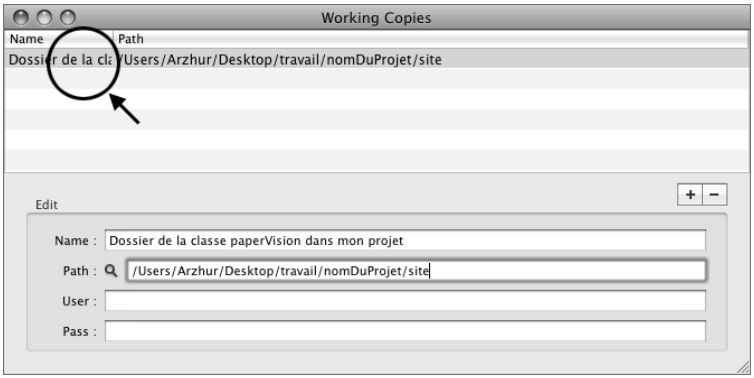
3. Dans le champ Path, inscrivez l'adresse de la classe `PaperVision` à importer (voir Figure 10.12) : **http://papervision3d.googlecode.com/svn/trunk/as3/trunk/src/.**

**Figure 10.12**
*Définition du contenu à importer (fenêtre de gauche).*




Le chargement du contenu appelé par l'URL se fait en double-cliquant sur la référence du chargement que nous venons de créer, dans la liste située au sommet de la première fenêtre (voir Figure 10.13).

**Figure 10.13**
*Ouverture de la fenêtre de chargement.*




1. Double-cliquez sur la référence pour lancer le chargement. Une nouvelle fenêtre apparaît (voir Figure 10.14).
2. Pour définir un dossier cible et y importer la classe, cliquez sur le lien "svn checkout". Une fenêtre de sélection de dossier apparaît.
3. Sélectionnez le répertoire de destination, soit le dossier racine du site pour lequel vous importez la classe ou un dossier distinct utilisé pour une gestion globale et native des classes, comme nous le détaillons plus loin dans ce chapitre (voir Figure 10.15).
4. Validez pour activer directement le chargement des fichiers. Le chargement s'effectue jusqu'à ce que la barre de chargement située en bas de la fenêtre s'arrête (voir Figure 10.16).

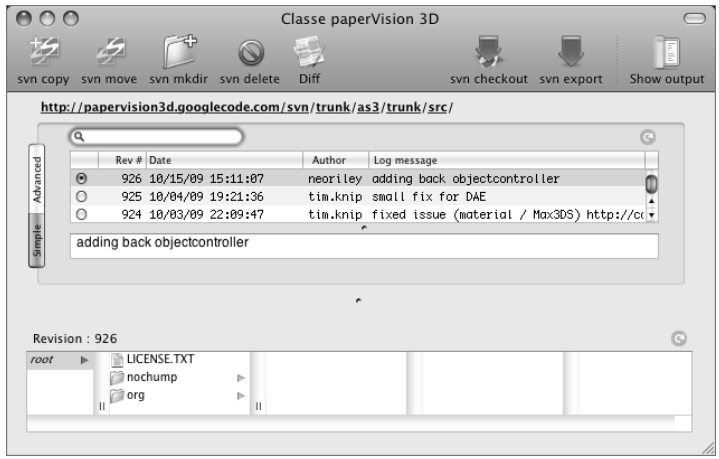



**Figure 10.14**
*Activation du chargement.*

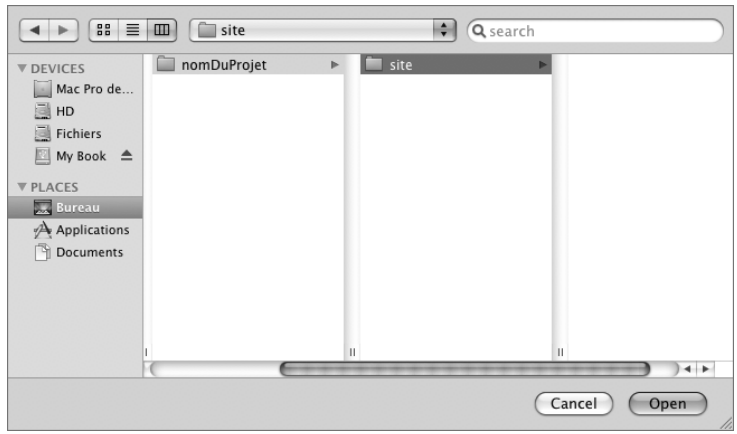


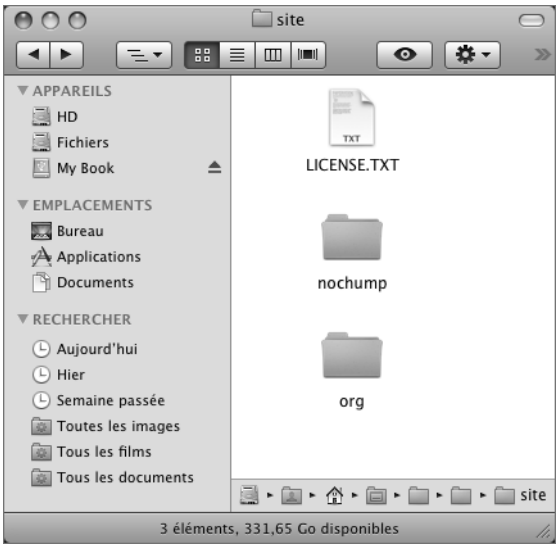

**Figure 10.15**
*Fenêtre de sélection du dossier cible.*


**Figure 10.16**
*Chargement en cours.*

5. Lorsque le chargement est terminé, quittez l'application. À l'intérieur de votre dossier, la classe `PaperVision` a été importée. Elle apparaît sous la forme de deux répertoires : nochump et org (voir Figure 10.17).

Vous pouvez maintenant l'appeler directement depuis un document Flash ou l'intégrer à l'API de Flash.

**Figure 10.17**
Classe chargée.




## Intégrer la classe *PaperVision* à Flash

Vous pouvez placer la classe `PaperVision`, dans Flash, selon deux méthodes. Soit, vous intégrez le chemin de la classe dans le moteur de Flash pour la rendre accessible à partir de tout nouveau document. Soit vous la copiez ponctuellement et manuellement dans chaque nouveau projet.

### Intégration native

Lorsque vous intégrez une classe nativement dans Flash, vous ne devez plus, par la suite, modifier l'emplacement du répertoire appelé en référence. Aussi, nous vous conseillons de bien choisir son emplacement avant de procéder à la liaison :

1. Par exemple, dans le dossier Flash du répertoire Applications de votre système (ou dans Program files pour Windows), créez un nouveau répertoire et nommez-le "Classes-Persos".
2. Dans ce nouveau dossier, créez un répertoire et intitulez-le à présent "papervision".
3. Copiez-y intégralement les dossiers que nous venons de télécharger ("org" et "nochump"). Attention toutefois, pour ne pas perdre la connexion avec Tortoise, préférez cibler directement ce nouveau répertoire lors du téléchargement de la classe.
4. Revenez dans Flash.
5. Pour intégrer la classe nativement, dans Flash, affichez les préférences (Cmd+U sur Mac ou Ctrl+U sur Windows).
6. Dans la catégorie ActionScript, cliquez sur le bouton Paramètres d'ActionScript 3.

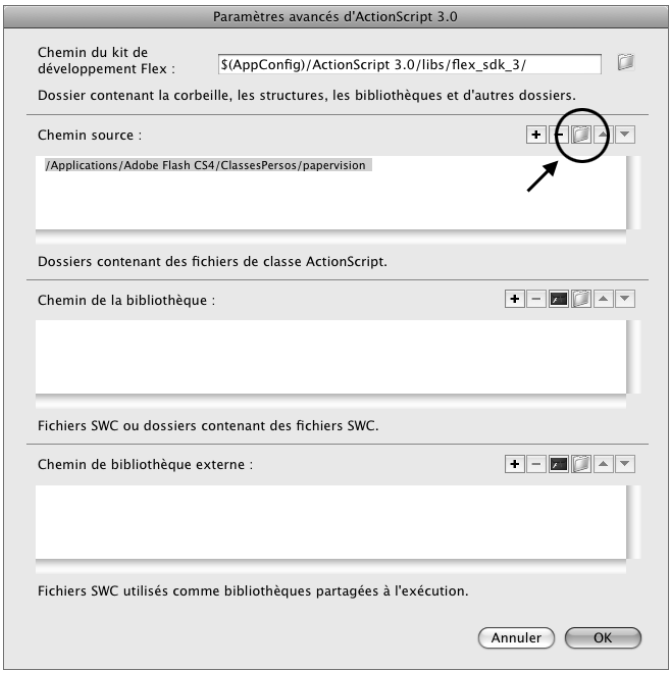
Dans la boîte de dialogue, trois zones de définition de chemin apparaissent. La première zone concerne l'intégration de classes natives (les trois zones sont confondues dans les anciennes versions de logiciel).

Pour la classe `PaperVision`, le dossier intitulé "org" contient l'ensemble des éléments requis. Plus précisément, c'est le dossier "papervision3d", contenu dans le dossier "org", contenu à son tour dans "papervision", qui devra plus tard être invoqué par ActionScript.

1. Dans la fenêtre de dialogue de Flash, à droite, cliquez sur le bouton de sélection de dossier. Puis, sélectionnez le répertoire "papervision" que nous venons de créer. En validant, le chemin de liaison apparaît dans la liste des classes intégrées (voir Figure 10.18).

**Figure 10.18**

*Fenêtre d'intégration des classes natives dans Flash CS4.*




2. Validez en refermant toutes les fenêtres. La classe est désormais intégrée nativement dans l'application et disponible pour tous les nouveaux documents.
3. Relancez de préférence l'application pour garantir la prise en compte de ces nouveaux éléments.

### Intégration ponctuelle

Vous pouvez aussi placer ponctuellement la classe dans chaque projet pour lequel elle est requise. Il n'est donc, dans ce cas, pas nécessaire de l'intégrer nativement comme vu à la section précédente :

1. Copiez le répertoire "org" que nous venons de télécharger.
2. Collez ce répertoire directement à la racine de votre site.

La classe est prête à être utilisée pour les documents Flash placés uniquement à la racine de ce dossier. Vous pouvez commencer à réaliser des interfaces avec des objets 3D.

**À retenir**

- `PaperVision` est une classe externalisée qu'il faut télécharger avant de pouvoir l'exploiter. Des utilitaires permettent de charger cette classe dynamiquement : Tortoise SVN pour Windows et SVNX pour Mac OS X.
- Il est possible d'intégrer une liaison native de Flash avec avec `PaperVision`, ou toute autre classe en modifiant les paramètres d'ActionScript 3, depuis les préférences du logiciel. Il faut pour cela cibler le répertoire qui les contient depuis la fenêtre de liaison de classes.
- Si vous n'intégrez pas les classes nativement, vous devez les copier, à chaque utilisation, au même niveau d'arborescence que le document qui y fait référence.
- Les classes sont compilées dans le SWF à la publication. Il n'est pas nécessaire de les placer sur le serveur d'hébergement.

## Modéliser en 3D pour Flash et *PaperVision*

Avant d'importer dans Flash des objets 3D réels, grâce à la classe `PaperVision`, vous devez naturellement disposer de ces objets. Vous pouvez les réaliser à partir de logiciels professionnels ou gratuits. Nous présentons ici différentes solutions pour rendre les créations 3D compatibles avec Flash et `PaperVision`. Nous proposons notamment l'outil Google Sketchup, logiciel 3D libre, compatible avec Flash et `PaperVision`.

### Exporter la 3D avec les logiciels standard

Tous les logiciels 3D sont compatibles Flash et `PaperVision` dès lors qu'ils peuvent exporter au format `DAE` (Maya, 3DS Max, Cinema 4D, Blender, Swift,…). Pour de nombreux logiciels toutefois, le plug-in *Collada* est requis. Il est disponible sur Internet. Pour chaque logiciel, reportez-vous au lien en rapport :

- Blender requiert un plug-in. L'aide à l'installation de Collada pour Blender est disponible à cette adresse : **http://colladablender.illusoft.com/cms/content/blogcategory/25/29/**.
- Swift 3D gère nativement l'export au format `DAE`. Consultez le site suivant pour en savoir plus sur Swift 3D : **http://www.erain.com/products/swift3d/?erain=v6pr**.
- Cinema 4D R11 accepte également le format `DAE` de Collada : **http://www.maxon.net/products/cinema-4d.html**.
- 3DSMax et Maya disposent d'un plug-in Collada spécifique. Retrouvez-le à : **http://sourceforge.net/projects/colladamaya/**.
- Une aide plus générale sur l'exportation de fichiers 3D avec Collada, pour tous logiciels 3D, est disponible à l'adresse : **http://www.australopitech.com/883-collada-papervision3d**.

Info

***Limite de taille des fichiers 3D.*** *La gestion de la 3D dans Flash avec* `PaperVision` *suppose que les fichiers puissent être consultés par tout type d'utilisateur et avec un chargement le plus rapide possible. Il est recommandé d'éviter de dépasser les scènes 3D composées de plus de 1 500 à 3 000 polygones. Au-delà, veillez à bien organiser le chargement du contenu par étapes successives pour éviter un chargement trop long, et éviter que le moteur de rendu ne peine à calculer entièrement le volume importé (présence de trous, points qui décrochent, etc.).*

### Exporter la 3D avec Google Sketchup

Google Sketchup est un logiciel de modélisation 3D, gratuit, si l'on s'en tient à une version basique. Cet outil initialement développé pour promouvoir la technologie GoogleEarth, offre une solution accessible et simple à utiliser pour créer des objets 3D pour l'environnement de Flash avec `PaperVision`, d'autant que Sketchup est très bien documenté – et en français. Google propose aussi une banque d'objets 3D déjà modélisés et libres de droits, simples à charger, pour un usage toutefois non commercial.

Info

*Les conditions d'utilisation de la banque d'objets 3D Google Sketchup sont disponibles à cette adresse :* **http://sketchup.google.com/intl/fr/3dwh/tos.html**.

Pour être importés dans Flash, les objets 3D de Google doivent être exportés au format `DAE`. Mais, Sketchup ne permet d'exporter qu'aux formats `KMZ` (optimisé) ou `SKP` (format natif).

Afin d'exporter un objet 3D Sketchup pour Flash, il suffit de le publier au format KMZ et de modifier son extension `.kmz` en `.zip`. Le dossier compressé obtenu contient le fichier du modèle 3D au format `DAE` et un fichier XML de métadonnées. Seul le fichier DAE et les textures éventuellement associées sont requis.

Dans cette section, nous présentons comment récupérer un objet 3D de la banque d'objets de Google Sketchup pour le convertir du format natif Sketchup en DAE pour Flash et `PaperVision`.

Info

*Pour accéder directement à une bibliothèque d'objets 3D KMZ, sans utiliser Sketchup, reportez-vous à l'adresse :* **http://sketchup.google.com/3dwarehouse/**.

Pour convertir des fichiers 3D natifs de Google dans un format compatible Flash, procédez comme suit :

1. Téléchargez la version gratuite de Sketchup à : **http://sketchup.google.com/intl/fr/download/gsu.html**.
2. Validez toutes les étapes de l'installation.
3. Puis lancez l'application. Au démarrage, Sketchup affiche un écran avec différentes aides de grande qualité (voir Figure 10.19).
4. Cliquez directement sur Commencer à utiliser Sketchup. Une scène 3D s'ouvre (voir Figure 10.20).

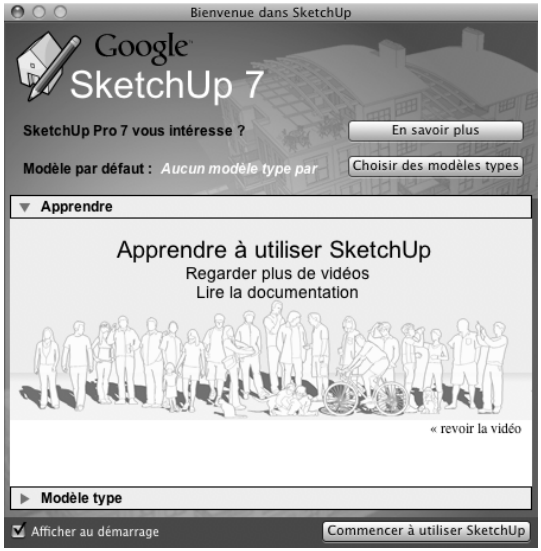



**Figure 10.19**
Lancement de Google Sketchup 7.

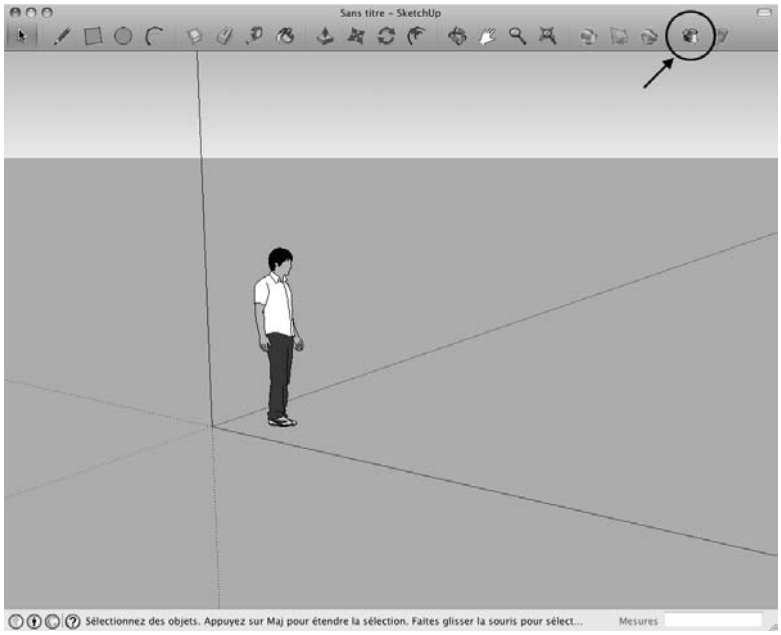
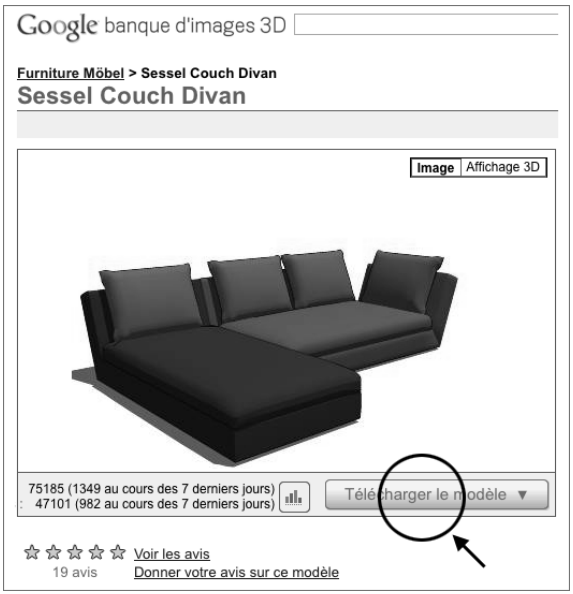



**Figure 10.20**
Nouvelle scène dans Sketchup.

5. En haut et à droite de la fenêtre, cliquez sur le bouton Télécharger des modèles (voir Figure 10.20). Une fenêtre de navigateur s'ouvre sur la page de la banque d'objets 3D de Google.
6. Saisissez une requête. Par exemple, sofa ou Sessel Couch Divan. Puis validez. Vous accédez à une liste d'objets.

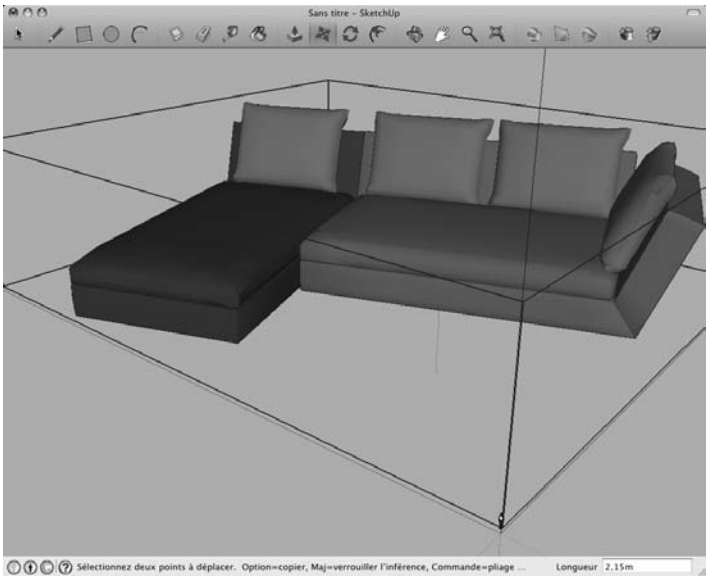
7. Sélectionnez celui de votre choix en le cliquant. Puis, dans la fiche descriptive de l'article, en bas de l'image de présentation, cliquez sur le lien Télécharger le modèle (voir Figure 10.21).

**Figure 10.21**
Télécharger le modèle.




8. Un message demande de confirmer le chargement du modèle dans le nouveau document Sketchup.
9. Confirmez et l'objet apparaît instantanément dans la scène. Vous pouvez l'exporter (voir Figure 10.22).

**Figure 10.22**
Modèle importé.



10. Si l'objet n'apparaît pas directement au centre de la scène, zoomez en arrière avec la roulette de la souris ou avec l'outil Loupe.
11. Repositionnez l'objet vers le point d'origine de la scène, à l'aide de l'outil de déplacement (Outil Déplacer/Copier en forme de 4 flèches cardinales rouges).
12. Supprimez le personnage de la scène en cliquant dessus et en faisant retour ou Suppr, au clavier.
13. Faites Fichier > Exporter > Modèle 3D. Nommez le document "sofa.kmz".

    Une fenêtre affiche des informations sur les propriétés de l'objet.
14. Refermez la fenêtre pour continuer. Vous pouvez conserver éventuellement le document au format natif pour le modifier si nécessaire en faisant Fichier > Enregistrer, au format SKP.
15. Puis, quittez Sketchup.

Info

*Si vous souhaitez pouvoir identifier des groupes d'objets pour* `PaperVision`*, en vue de modifier depuis* `PaperVision` *les teintes des composantes de forme des objets, utilisez la fenêtre Structure de Sketchup. Puis renommez chaque composante de l'objet 3D affiché dans cette liste. Aidez-vous éventuellement de l'option Créer un groupe, disponible par un clic-droit sur les formes graphiques composant les objets dans la scène.*

Une fois le document exporté au format `MKZ`, vous pouvez en extraire le fichier Collada DAE.

Modifiez l'extension du fichier "sofa.kmz" en "sofa.zip". Confirmez éventuellement la boîte de dialogue d'avertissement qui apparaît. Puis, décompressez le fichier obtenu. Un dossier qui porte le même nom que le document apparaît. À l'intérieur, nous distinguons le fichier DAE isolé dans un répertoire nommé "models" (voir Figure 10.23).

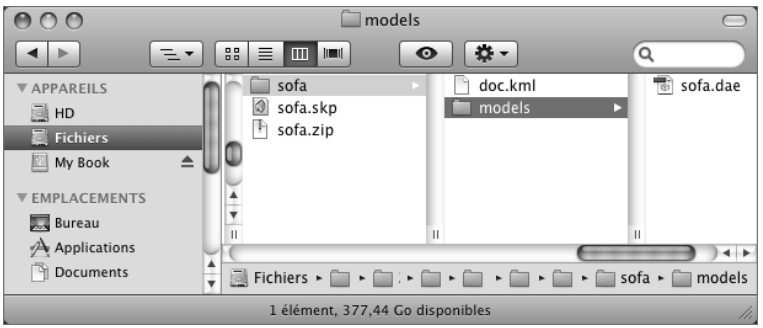



**Figure 10.23**
*Extraction du fichier DAE pour Flash et* `PaperVision`.

Nous pouvons conserver la structure des sous-dossiers. Il n'en sera que plus simple pour gérer les éventuelles modifications par la suite. Le fichier DAE est maintenant exploitable directement dans Flash, en ActionScript.

Info

*Pour en savoir plus sur Sketchup, consultez cette adresse :* **http://sketchup.google.com/intl/fr**.

*Pour accéder à une documentation détaillée, en texte et sous la forme de tutoriels vidéo sur la modélisation avec le logiciel Sketchup, consultez les adresses suivantes :*

- **http://sketchup.google.com/intl/fr/training/videos.html**.
- **http://sketchup.google.com/support/**.

**À retenir**

- *Vous pouvez exporter des objets 3D pour Flash et* `PaperVision` *si celui-ci est au format Collada DAE.*
- *Google Sketchup est une application 3D libre qui donne accès à des modèles 3D dans un format compatible avec Flash et* `PaperVision`.
- *Pour convertir un fichier KMZ en fichier DAE, il suffit de remplacer l'extension* `.kmz` *en* `.zip` *et décompresser le fichier obtenu.*

## Programmer les mouvements de caméra 3D

Au cours de cette section, nous utilisons un modèle d'objet 3D de référence Sessel Couch Divan, téléchargé depuis la banque d'objets 3D de Google. Nous allons voir comment ajouter dans le scénario de l'interface auteur de Flash, les commandes ActionScript 3D de `PaperVision` pour animer une caméra et l'objet 3D.

Dans cet exemple, nous utilisons des transitions de type `TweenMax` que nous appliquons à l'objet `Camera` en vue de réaliser plusieurs dispositifs de présentation.

Nous commençons par créer une animation autonome de la caméra 3D pour approcher l'objet et donner l'illusion que c'est lui qui s'approche de l'écran. Ensuite, nous associons ces actions à une occurrence de bouton pour effectuer un zoom de caméra, afin de permettre à l'utilisateur de contrôler lui-même le mouvement de la caméra. Nous ajoutons enfin un comportement à l'objet 3D, afin de le faire évoluer devant la caméra, une fois celle-ci fixée.

Pour cela, nous utilisons la classe `PaperVision` accessible depuis Flash.

*Exemples > ch10_3DPaperVision_1.fla*

Dans le document "ch10_3DpaperVision_1.fla", nous pouvons voir un bouton en bas et à droite, en forme de loupe, qui est associé à une action sur caméra. Au-dessous, un habillage gris matérialise la limite de la zone d'affichage réservée pour la 3D telle que définie en ActionScript. Une fois publié, le sofa part du fond de la scène et arrive progressivement au

premier plan, tout en tournant sur lui-même. En cliquant sur le bouton Loupe, la caméra se rapproche un peu plus encore à chaque clic (voir Figures 10.24 à 10.26).


**Figure 10.24**
Aperçu du document publié au début de l'animation.

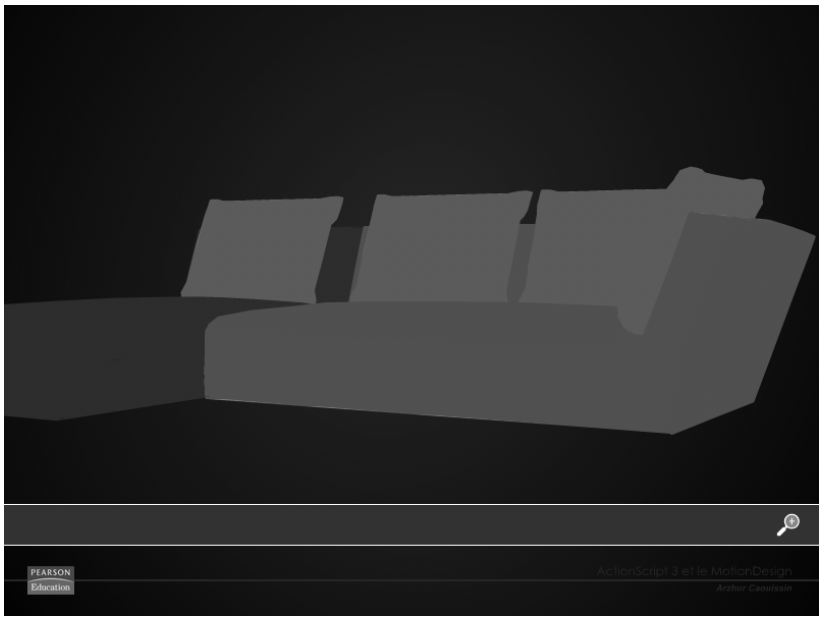



**Figure 10.25**
Aperçu du document publié au milieu de l'animation.

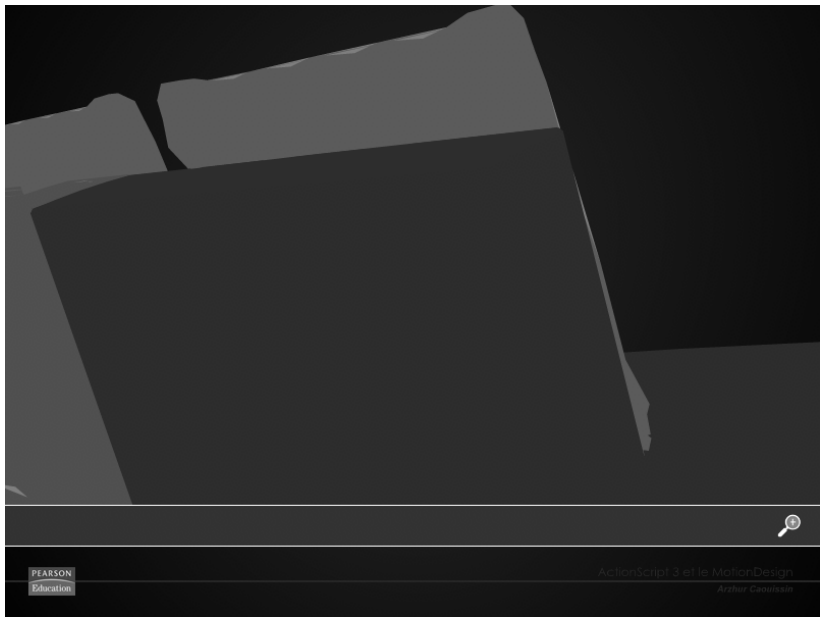



**Figure 10.26**
*Aperçu du document publié à la fin de l'animation.*

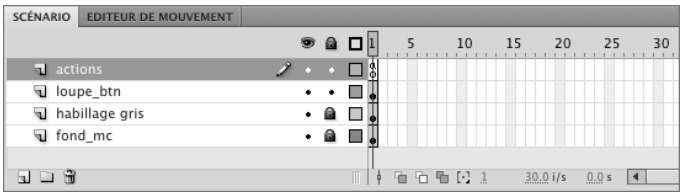



**Figure 10.27**
*Scénario de la scène principale.*

Dans la fenêtre de scénario de la scène principale, au-dessus du calque `fond_mc`, nous trouvons le bouton `loupe_btn`, l'habillage gris et le calque `Actions` (voir Figure 10.27). Dans le calque `Actions`, nous pouvons lire le code suivant :

```
//-------------------- initialisation
import gs.TweenMax;
import gs.easing.*;

import org.papervision3d.view.*;
import org.papervision3d.cameras.*;
import org.papervision3d.scenes.*;
import org.papervision3d.render.*;
import org.papervision3d.objects.primitives.*;
import org.papervision3d.events.*;
import org.papervision3d.materials.*;
import org.papervision3d.objects.parsers.DAE;

var espace:Viewport3D;
var rendu:BasicRenderEngine;
var scene:Scene3D;
var camera:Camera3D;

var objetDAE:DAE = new DAE();
```

```
objetDAE.load("objets3D/sofa/models/sofa.dae");

//-------------------- actions
initialisation();
function initialisation() {
   espace=new Viewport3D(stage.stageWidth,stage.stageHeight-110);
   addChild(espace);
   rendu = new BasicRenderEngine();
   scene = new Scene3D();
   camera = new Camera3D();
   //
   objetDAE.rotationX=0;
   objetDAE.rotationY=-90;
   objetDAE.rotationZ=0;
   objetDAE.x=0;
   objetDAE.y=0;
   objetDAE.z=0;
   camera.y=15;
   //
   activerAffichage3D();
}
function activerAffichage3D() {
   scene.addChild(objetDAE);
   TweenMax.to(camera, 5, {z:camera.z+900, delay:0, ease:Strong.easeInOut});
   addEventListener(Event.ENTER_FRAME, enBoucle);
}
function enBoucle(evt:Event) {
   objetDAE.yaw(-1);
   rendu.renderScene(scene, camera, espace);
}

//modifier un objet 3D
loupe_btn.addEventListener(MouseEvent.CLICK,deplacerObjetDAE);
function deplacerObjetDAE(evt:MouseEvent) {
   TweenMax.to(camera, 5, {zoom:camera.zoom+100, delay:0,
   ➥ ease:Strong.easeOut});
}
```

Dans la première partie de ce code, nous commençons par importer l'ensemble des sous-classes de la classe `PaperVision`. Nous ciblons, pour chacune d'entre elles, le dossier "org", comme point de référence de cet ensemble. Du fait que nous avons intégré le chemin de classes dans les préférences de l'application, nous spécifions ici le dossier org situé dans le répertoire "ClassesPersos" que nous avons préalablement créé.

Les classes importées sont utilisées pour les transitions `TweenMax` :

```
//-------------------- initialisation
import gs.TweenMax;
import gs.easing.*;
```

Puis, viennent celles de `PaperVision` :

- `import org.papervision3d.view.*;` : la classe `view` permet de gérer l'environnement 3D.

- `import org.papervision3d.cameras.*;` : la classe `cameras` permet de créer une caméra.
- `import org.papervision3d.scenes.*;` : la classe `scenes` permet de créer la scène que constitue l'espace 3D.
- `import org.papervision3d.render.*;` : la classe `render` calcule l'affichage 3D.
- `import org.papervision3d.objects.primitives.*;` : la classe `primitives`, non utilisée dans ce document, nous permettra de créer plus tard des formes primitives, telles que sphères, cylindres, cônes, plans, ou cubes.
- `import org.papervision3d.events.*;` : la classe `events` autorise l'interactivité avec les objets de la scène 3D.
- `import org.papervision3d.materials.*;` : la classe `materials` permet d'appliquer des textures à partir d'images à des formes 3D. Nous l'utilisons également plus loin dans ce chapitre.
- `import org.papervision3d.objects.parsers.DAE;` : enfin, la classe DAE autorise la gestion d'objets 3D importés au format `DAE`.

D'autres classes sont disponibles. Vous pouvez les identifier en parcourant le contenu du dossier `org`.

À la suite, nous définissons quelques variables qui caractérisent les types d'objets requis pour la construction de l'univers en trois dimensions :

```
var espace:Viewport3D;
var rendu:BasicRenderEngine;
var scene:Scene3D;
var camera:Camera3D;
```

`Viewport3D` définit la surface affichée à l'écran. Plus loin dans le code, cette variable reçoit en paramètre les dimensions de la zone d'affichage en pixels. L'objet `basicRenderEngine` est le moteur de rendu. L'objet `scene3D` constitue le volume à l'intérieur duquel nous plaçons tous les éléments en 3D. L'objet `Camera3D` enfin désigne l'œil du spectateur, la caméra.

Plus bas, nous créons un nouvel objet `DAE` qui charge le fichier 3D dans l'interface. Nous indiquons le chemin qui part du document Flash vers le fichier, à l'intérieur de la structure de dossier `KMZ` que nous avons décompressé :

```
var objetDAE:DAE = new DAE();
objetDAE.load("objets3D/sofa/models/sofa.dae");
```

La deuxième partie du code exécute l'interface 3D. Les actions sont rassemblées dans des fonctions afin de permettre une gestion de l'affichage contrôlée dans le temps.

```
//-------------------- actions
initialisation();
function initialisation() {
   espace=new Viewport3D(stage.stageWidth,stage.stageHeight-110);
   addChild(espace);
   rendu = new BasicRenderEngine();
```

```
        scene = new Scene3D();
        camera = new Camera3D();
        //
        objetDAE.rotationX=0;
        objetDAE.rotationY=-90;
        objetDAE.rotationZ=0;
        objetDAE.x=0;
        objetDAE.y=0;
        objetDAE.z=0;
        camera.y=15;
        //
        activerAffichage3D();
    }
```

La première instruction de la première fonction définit la zone d'affichage du rendu. Nous spécifions ici une largeur équivalente à la largeur de la scène du document Flash (`stage.stageWidth`) et une hauteur de 110 pixels de moins que la hauteur du document (`stage.stageHeight-110`). La hauteur est réduite afin de préserver la visibilité des contrôles de navigation que nous avons placés sur la scène (outil Loupe, entre autres). Gardez à l'esprit que plus votre surface de rendu est importante, plus les ressources de la machine client seront sollicitées.

La construction d'une interface 3D implique l'affichage de deux types d'objets. D'abord, nous affichons la zone de rendu avec ici l'objet espace dont nous venons de définir les dimensions. Plus loin, nous affichons les objets de la scène en 3D (DAE et primitives éventuelles). L'objet espace agit comme une fenêtre à travers laquelle on peut observer l'objet `scene` en 3D. L'objet `scene` ne définit en rien l'affichage, il n'est qu'un conteneur, pas un objet d'affichage.

Cette distinction est importante, car si la zone de rendu peut être gérée comme tout contenu depuis la liste d'affichage. La scène, elle, ne peut être invoquée directement, pour être redistribuée au-dessous d'un autre objet par exemple. Pour permettre de placer des symboles de la scène Flash au-dessus de la scène 3D, nous devons affecter le premier objet d'affichage, l'objet `espace` et non l'objet `scene`. Ainsi, si nous souhaitons que notre création 3D s'affiche à l'arrière-plan du bandeau gris du document Flash, par exemple, nous spécifions `addChildAt(espace,1)` au lieu de simplement `addChild(espace)`. Les objets 3D sont ajoutés à la scène avec également la méthode `addChild`, mais cela ne les inclut pas pour autant dans la liste d'affichage globale. De cette manière, l'objet `scene` induit une obstruction sémantique entre le contenu Flash et le contenu géré par `PaperVision`.


*Il est également possible de placer un symbole par-dessus un objet 3D en conservant toutefois l'objet espace au sommet de la liste d'affichage. Pour cela, il suffit d'importer le symbole à ajouter en tant que texture dans un objet primitif de type* `Plane`*, par exemple, puis de gérer la position de cette primitive sur l'axe des Z avec un positionnement frontal par rapport à la caméra. Reportez-vous à la section "Interactivité avec les touches du clavier" pour en savoir plus sur la construction des primitives.*

Une fois la zone d'affichage définie, nous l'ajoutons donc à la liste d'affichage classique avec `addChild(espace)`.

À la suite, nous matérialisons les objets pour lesquels nous avons créé les variables en amont, les variables :

```
rendu = new BasicRenderEngine();
scene = new Scene3D();
camera = new Camera3D();
```

Puis, nous modifions les propriétés des objets avant de les afficher, pour les caler dans l'environnement 3D :

```
objetDAE.rotationX=0;
objetDAE.rotationY=-90;
objetDAE.rotationZ=0;
objetDAE.x=0;
objetDAE.y=0;
objetDAE.z=0;
camera.y=15;
```

Nous ajustons les rotations de l'objet 3D afin qu'il apparaisse de face à la caméra (`rotationY=-90`). Puis, nous modifions légèrement la position verticale de la caméra pour recentrer également l'objet dans la scène (`camera.y=15`).

En dessous, nous poursuivons avec l'appel d'une fonction qui va activer l'affichage 3D des objets de la scène, et engager les animations des contenus alors prêts à être animés :

```
function activerAffichage3D() {
   scene.addChild(objetDAE);
   TweenMax.to(camera, 5, {z:camera.z+900, delay:0, ease:Strong.easeInOut});
   addEventListener(Event.ENTER_FRAME, enBoucle);
}
function enBoucle(evt:Event) {
   objetDAE.yaw(-1);
   rendu.renderScene(scene, camera, espace);
}
```

La deuxième instruction de cette fonction active une interpolation de type `TweenMax` qui anime la propriété `z` de l'objet `camera` de manière à créer un travelling jusqu'à atteindre l'objet DAE (`camera.z+=900`).

Plus bas, un écouteur, associé à un gestionnaire de type `ENTER_FRAME`, active le calcul du rendu de la scène 3D (`rendu.renderScene(scene, camera, espace)`). Les trois objets `scene`, `camera` et `espace` sont passés en paramètre.

Une dernière instruction ajoute un effet d'animation sur l'objet DAE. Cette instruction adopte une syntaxe propre à l'environnement `PaperVision` et équivaut à une rotation en Y d'un pas d'incrémentation de 1° à chaque image. La méthode employée est `yaw` et donne : `objetDAE.yaw(-1)`. Nous abordons d'autres méthodes de ce type dans la section suivante.

Le programme s'achève sur l'ajout d'un comportement sur le bouton Loupe. En cliquant dessus, nous lançons une nouvelle interpolation qui anime la caméra avec une autre propriété : `zoom`.

```
//modifier un objet 3D
loupe_btn.addEventListener(MouseEvent.CLICK,deplacerObjetDAE);
function deplacerObjetDAE(evt:MouseEvent) {
   TweenMax.to(camera, 5, {zoom:camera.zoom+100, delay:0,
   ➥ ease:Strong.easeOut});
}
```

Les propriétés `zoom` et `z` se différencie par un déplacement physique de la caméra pour `z`, tandis que `zoom` ne fait qu'agrandir l'image. Ainsi, la propriété `zoom` permet de se rapprocher de l'objet 3D sans jamais le traverser, alors que `z` passerait à travers.


La sous-classe `Camera3D` est parfois aussi appelée `FreeCamera3D` (dans d'anciennes versions de `PaperVision`). Selon la version du *package* que vous utilisez, vérifiez que le nom de votre classe `Camera` présente dans le code ActionScript de votre document correspond bien au nom du fichier `.as` disponible dans les sous-dossiers de la classe `PaperVision`.

Il peut arriver, selon le poids du fichier, que Flash annule le rendu à la publication. Le calcul du rendu est alors peut-être trop long pour Flash. Pour corriger ce problème, allez dans les paramètres de publication, et sous le champ mot de passe, allongez la durée limite d'exécution du script avant laquelle Flash annule le rendu.

**À retenir**

- Il est possible d'animer facilement la caméra dans un environnement 3D pour créer des travellings, des zooms et des trajectoires, grâce à la combinaison de propriétés 3D et d'interpolations de type `TweenMax`.
- Le contenu d'une scène 3D n'est pas accessible directement depuis la liste d'affichage, mais son enveloppe oui. Il est donc possible de placer les animations 3D entre différents objets dans la scène du document Flash.
- Les commandes ActionScript de `PaperVision` peuvent être rédigées directement depuis la fenêtre de scénario.

Le zoom se distingue d'un déplacement en Z, en cela qu'il ne traverse jamais l'objet, au contraire de Z.

## Programmer les mouvements d'objets 3D

Dans cette section, nous allons ajouter des animations qui permettent de manipuler directement les objets 3D de format DAE, importés dans Flash.

Exemples > ch10_3DPaperVision_2.fla

Le document "ch10_3DPaperVision_2.fla" est basé sur l'exemple précédent. Sur la scène principale, en plus de la bande grise et du bouton Loupe, figure un nouveau bouton nommé `sofa_btn` (voir Figure 10.28). Ce bouton est destiné à recevoir des instructions d'animation de l'objet 3D.

En publiant le document (voir Figure 10.29), lorsque le sofa est au premier plan, le bouton associé à l'objet le fait pivoter sur lui-même.

**Figure 10.28**
Aperçu du document publié.

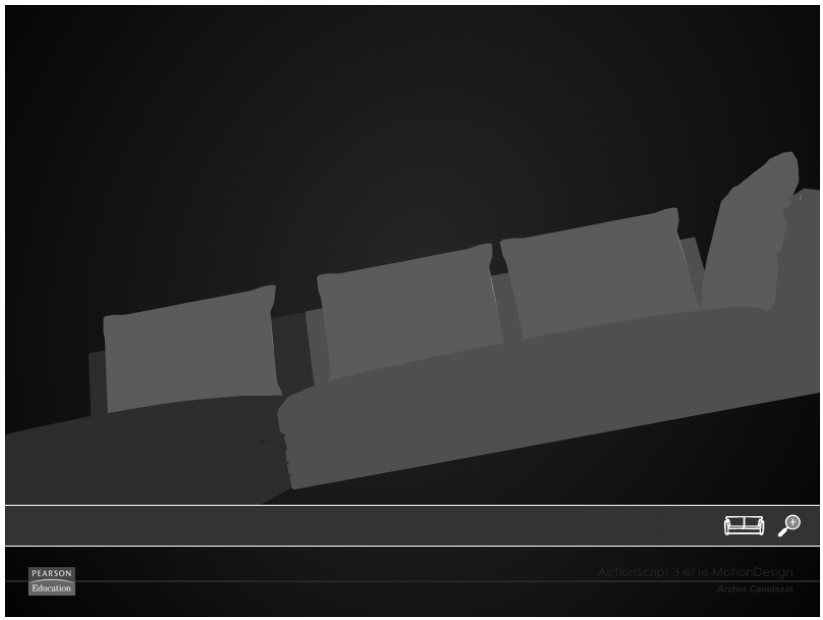



**Figure 10.29**
Scénario de la scène principale.


Le calque `actions` affiche le code suivant :

```
//-------------------- initialisation
import gs.TweenMax;
import gs.easing.*;

import org.papervision3d.view.*;
import org.papervision3d.cameras.*;
import org.papervision3d.scenes.*;
import org.papervision3d.render.*;
import org.papervision3d.objects.primitives.*;
import org.papervision3d.events.*;
import org.papervision3d.materials.*;
import org.papervision3d.objects.parsers.DAE;

var espace:Viewport3D;
var rendu:BasicRenderEngine;
var scene:Scene3D;
var camera:Camera3D;

var objetDAE:DAE = new DAE();
objetDAE.load("objets3D/sofa/models/sofa.dae");
```

```
//-------------------- actions
initialisation();
function initialisation() {
   espace=new Viewport3D(stage.stageWidth,stage.stageHeight-110);
   addChild(espace);
   rendu = new BasicRenderEngine();
   scene = new Scene3D();
   camera = new Camera3D();
   //
   objetDAE.rotationX=0;
   objetDAE.rotationY=-90;
   objetDAE.rotationZ=0;
   objetDAE.x=0;
   objetDAE.y=0;
   objetDAE.z=0;
   camera.y=15;
   camera.rotationY=-10;
   //
   activerAffichage3D();
}

function activerAffichage3D() {
   scene.addChild(objetDAE);
   TweenMax.to(camera, 5, {z:camera.z+900, delay:0, ease:Strong.easeInOut});
   addEventListener(Event.ENTER_FRAME, enBoucle);
}
//
function enBoucle(evt:Event) {
   objetDAE.yaw(-1);
   objetDAE.roll(-1);
   objetDAE.pitch(-1);
   rendu.renderScene(scene, camera, espace);
}

//modifier un objet 3D
loupe_btn.addEventListener(MouseEvent.CLICK,deplacerObjetDAE);
function deplacerObjetDAE(evt:MouseEvent) {
   TweenMax.to(camera, 5, {zoom:camera.zoom+100, delay:0, ease:Strong.easeOut});
}

//Animer le sofa
sofa_btn.addEventListener(MouseEvent.CLICK,animerCamera);
function animerCamera(evt:MouseEvent) {
   TweenMax.to(objetDAE, 5, {rotationY:objetDAE.rotationY+45, rotationX:
   ➥ objetDAE.rotationX+45, rotationZ:objetDAE.rotationZ+45, delay:0,
   ➥ ease:Strong.easeInOut});
}
```

Dans cet exemple, nous animons l'objet DAE en le ciblant par le nom d'objet qui lui a été attribué (objetDAE), dès la déclaration des différents objets en début de code.

L'animation d'un objet 3D se construit de la même manière que l'animation d'une caméra. Les transitions TweenMax permettent de repositionner l'objet dans l'espace, de le faire pivoter en rotation sur les axes X, Y et Z.

Nous utilisons aussi d'autres méthodes, dont la méthode yaw(), déjà évoquée à la section précédente :

```
objetDAE.yaw(-1);
```

La méthode yaw() exécute une rotation latérale sur Y d'une valeur qui correspond au chiffre renseigné en paramètre. L'objet tourne au sol sur lui-même comme un tourniquet, de degré en degré :

```
objetDAE.roll(-1);
```

La méthode roll() exécute une rotation verticale sur X d'une valeur qui correspond au chiffre renseigné en paramètre. L'objet tourne ici sur lui-même, mais l'axe de rotation est parallèle à l'écran :

```
objetDAE.pitch(-1);
```

La méthode pitch() exécute une rotation frontale en Z d'une valeur correspondant au chiffre renseigné en paramètre. L'objet tourne sur lui-même, comme les aiguilles d'une montre.

Ces trois méthodes peuvent être appliquées indépendamment les unes des autres et cumulées à des rotations X, Y et Z. Dans notre exemple, les trois méthodes donnent un effet de rotation hybride qui fait évoluer l'objet dans toutes les directions simultanément. Vous pouvez les désactiver une par une pour observer leur comportement individuel.

En fin de programme, un écouteur est attaché au bouton sofa_btn et effectue une interpolation de type TweenMax en modifiant les paramètres de rotation X, Y et Z de l'objet DAE.

Info

**Optimiser un site 3D.** Certains sites, bien que très spectaculaires, utilisent le même principe de navigation spatiale 3D avec des transitions de type TweenMax ou Caurina. Le site *System Bolaget Kampanj* (suédois) en est le parfait exemple. Il affiche au premier plan un village modélisé en 3D. À l'arrière-plan, sur la scène principale du document Flash, des panoramiques d'images PNG sont animés. Pour garantir la fluidité de l'effet 3D, les PNG du décor sont isolés dans des clips distincts et animés avec un cœfficient d'accélération (voir Chapitre 1). La superposition de la scène 3D et 2D donne l'illusion d'un ensemble entièrement 3D. En isolant une partie du décor sous la forme d'images fixes, plutôt que de tout réaliser en 3D, nous optimisons considérablement le poids d'un projet. Vous pouvez consulter le site référent à l'adresse suivante : **http://www.systembolagetkampanj.se/forskar-rapport/**.

**À retenir**

- Il est possible d'animer les propriétés 3D d'un objet DAE importé dans le document Flash, pour le faire évoluer, se déplacer ou pivoter, sur action de l'utilisateur, par exemple.
- Des méthodes d'animation (roll, yaw et pitch) permettent de simplifier les mouvements de ces objets dans l'espace 3D et peuvent être cumulées avec l'animation d'autres propriétés.