eBook Gratuit

# APPRENEZ ActionScript 3

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#actionscrip

t-3

# Table des matières

# À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: actionscript-3

It is an unofficial and free ActionScript 3 ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ActionScript 3.



Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

# Chapitre 1: Mise en route avec ActionScript 3

## Remarques

ActionScript 3 est le langage de programmation des environnements d'exécution Adobe Flash Player et Adobe AIR. Il s'agit d'un langage basé sur les objets ECMAScript utilisé principalement pour le développement d'applications natives sur les ordinateurs de bureau (Windows / Mac) et les appareils mobiles (iOS / Android).

Ressources pédagogiques Adobe: http://www.adobe.com/devnet/actionscript/learning.html

Histoire et plus de détails: https://en.wikipedia.org/wiki/ActionScript

Documentation en ligne sur les classes et la référence:
http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/package-detail.html

## Versions

# Il existe une version unique de Actionscript 3, nommée "ActionScript 3.0".

| Version Flash | Nom de code | Changements et améliorations | Date de sortie |
|---|---|---|---|
| Flash Player 9.x | Zaphod | Première version | 2006-06-23 |
| Flash Player 10.0 | Astro | introduit le type `Vector.<T>` , les filtres de shader Adobe Pixel Bender dans la classe `flash.filters.ShaderFilter` et son support matériel sur plusieurs processeurs. | 2008-10-15 |
| Flash Player 10.1 | Argo | introduit la classe `flash.events.TouchEvent` pour travailler avec des périphériques multitouch, ainsi que d'autres supports matériels pour périphériques mobiles, tels que l'accéléromètre. | 2010-06-10 |
| Flash Player 10.2 | Épicé | introduit la classe `flash.media.StageVideo` et le framework général pour travailler avec la lecture de la scène vidéo dans AS3. | 2011-02-08 |
| Flash Player 11 | Serrano | ajoute le support H.264 au streaming vidéo sur les objets `NetStream` dans les deux sens. Il ajoute également le support SSL / TLS pour la connexion Flash avec la classe | 2011-10-04 |

| Version Flash | Nom de code | Changements et améliorations | Date de sortie |
|---|---|---|---|
| | | `SecureSocket` . | |
| Flash Player 11.4 | Brannan | introduit la classe `flash.system.Worker` et la possibilité de déléguer le travail asynchrone à d'autres threads sur le client. | 2012-08-10 |
| Flash Player 11.8 | Harrison | Suppression du support matériel (compilation JIT) pour les filtres de nuance Adobe Pixel Bender, ce qui réduit considérablement les performances de toute exécution de filtre de shader PB. | 2013-05-09 |

# Examples

## Vue d'ensemble de l'installation

ActionScript 3 peut être utilisé en installant le kit Adobe AIR SDK ou Apache Flex SDK ou en tant que composant du produit Adobe Animate CC *(anciennement appelé Flash Professional)* .

Adobe Animate CC est une solution logicielle professionnelle permettant de créer des projets AS3 à l'aide d'outils visuels. Une fois installées, aucune autre étape n'est nécessaire pour créer des projets AS3.

Le SDK AIR et le SDK Flex peuvent être utilisés avec des outils de ligne de commande ou avec divers IDE tiers.

Outre Adobe Animate CC, il existe quatre autres IDE courants capables de fonctionner avec AS3. Ces IDE ont leurs propres instructions sur la façon de démarrer.

- Flash Builder (Par Adobe - basé sur Eclipse)
- IntelliJ IDEA (par Jetbrains)
- FlashDevelop
- FDT (plugin Eclipse)

## Bonjour le monde

Un exemple de classe de document qui affiche «Hello, World» sur la console de débogage une fois instancié.

```
import flash.display.Sprite;

public class Main extends Sprite {

    public function Main() {
        super();
```

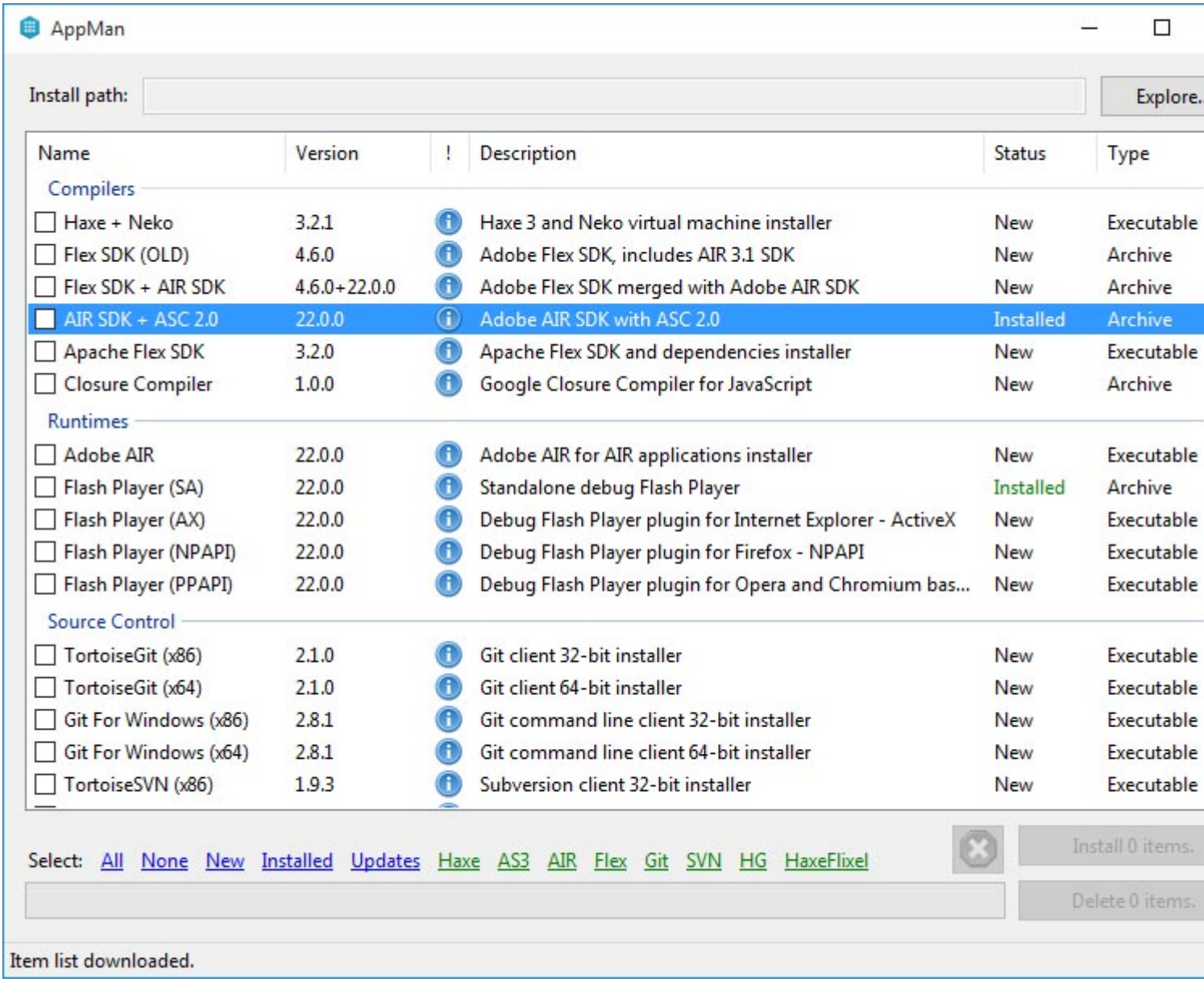
```
        trace("Hello, World");
    }

}
```

## Installation de développement Flash

FlashDevelop est un IDE open source multi-plateforme créé en 2005 pour les développeurs Flash. Sans frais, c'est un moyen très populaire pour commencer à développer avec AS3.

Pour installer FlashDevelop:

1. Téléchargez le fichier d'installation et lancez le programme d'installation
2. Une fois l'installation terminée, lancez FlashDevelop. Lors du premier lancement, la fenêtre `App Man` devrait apparaître pour vous demander de choisir le SDK et les outils à installer.




*Si AppMan ne s'ouvre pas automatiquement ou si vous souhaitez ajouter quelque chose ultérieurement, ouvrez-le en choisissant «Installer le logiciel» dans le menu «Outils».*

Vérifiez l'élément **AIR SDK + ACS 2.0** (dans la section «Compilateur») et l'élément **Flash Player (SA)** dans la section «Runtimes» (ainsi que tout ce que vous souhaitez installer). Cliquez sur le bouton d'installation.

3. Une fois le SDK installé, testons en créant un projet hello world. Commencez par créer un nouveau projet (à partir du menu *Projet* )

4. Choisissez le **projecteur AIR AS3** dans la liste et attribuez-lui un nom / un emplacement.

5. Dans le panneau du gestionnaire de projets (choisissez «Gestionnaire de projets» dans le menu d'affichage si ce n'est pas déjà visible), développez le dossier **src** et ouvrez le fichier `Main.as`

6. Dans le fichier `Main.as` , vous pouvez maintenant créer un premier programme comme Hello World

7. Exécutez votre projet en cliquant sur l'icône de lecture ou en appuyant sur `F5` ou `Ctrl+Enter` . Le projet compilera et une fois terminé, une fenêtre vide devrait apparaître (ceci est votre application). Dans la fenêtre de sortie de FlashDevelop, vous devriez voir les mots: **Hello World** .

Vous êtes maintenant prêt à développer des applications AS3 avec FlashDevelop!

## Installation d'Apache Flex

*de http://flex.apache.org/doc-getstarted.html*

1. Téléchargez le programme d'installation du SDK

2. Exécutez le programme d'installation du SDK. La première question qui vous sera posée est le répertoire d'installation.

    - sur un Mac, utilisez `/Applications/Adobe Flash Builder 4.7/sdks/4.14.0/`
    - sur un PC, utilisez `C:\Program Files(x86)\Adobe Flash Builder 4.7\sdks\4.14.0`

    Vous devrez créer les dossiers 4.14.0. Appuyez sur Suivant. Acceptez les licences SDK et installez-les.

Instructions spécifiques à l'IDE pour la configuration d'Apache Flex:

- Flash Builder
- IntelliJ IDEA
- FlashDevelop
- FDT

## Création de projets Flex ou Flash sur la ligne de commande à l'aide de mxmlc

Le compilateur Flex ( `mxmlc` ) est l'une des parties les plus importantes du SDK Flex. Vous pouvez modifier le code AS3 dans n'importe quel éditeur de texte que vous aimez. Créez un fichier de classe principal qui s'étend depuis `DisplayObject` .

Vous pouvez déclencher des builds sur la ligne de commande comme suit:

```
mxmlc -source-path="." -default-size [width in pixels] [height in pixels] -default-frame-rate
[fps] -o "outputPath.swf" "mainClass.as"
```

Si vous devez compiler un projet Flash (par opposition à Flex), vous pouvez ajouter une référence à la bibliothèque Flash comme suit (vous devrez installer l'IDE Adobe Animate):

```
mxmlc -source-path="." -library-path+="/Applications/Adobe Animate CC 2015.2/Adobe Animate CC
2015.2.app/Contents/Common/Configuration/ActionScript 3.0/libs" -static-link-runtime-shared-
libraries=true -default-size [width in pixels] [height in pixels] -default-frame-rate [fps] -o
"outputPath.swf" "mainClass.as"
```

Ou sous Windows:

```
mxmlc -source-path="." -library-path+="C:\Program Files\Adobe\Adobe Animate CC
2015.2\Common\Configuration\ActionScript 3.0\libs" -static-link-runtime-shared-libraries=true
-default-size [width in pixels] [height in pixels] -default-frame-rate [fps] -o
"outputPath.swf" "mainClass.as"
```

## Un exemple "Hello World" affiché

```
package {
    import flash.text.TextField;
    import flash.display.Sprite;

    public class TextHello extends Sprite {
        public function TextHello() {
            var tf:TextField = new TextField();
            tf.text = "Hello World!"
            tf.x = 50;
            tf.y = 40;
            addChild(tf);
        }
    }
}
```

Cette classe utilise la classe `TextField` pour afficher le texte.

Lire Mise en route avec ActionScript 3 en ligne: https://riptutorial.com/fr/actionscript-3/topic/1065/mise-en-route-avec-actionscript-3

# Chapitre 2: Chargement de fichiers externes

## Remarques

Dans certains cas, votre application ne peut pas manipuler le contenu des ressources chargées à partir d'une ressource externe (par exemple, transformer des images). Je ne me souviens pas avec certitude de ce qu'elles sont, mais je suis à peu près certain que cela concerne le chargement de contenu interdomaine.

## Examples

### Chargement d'images externes / SWF avec le chargeur

1. Créez un objet Loader:

```
var loader:Loader = new Loader();   //import
```

2. Ajouter des écouteurs sur le chargeur. Les standards sont complets et les erreurs io / de sécurité

```
loader.contentLoaderInfo.addEventListener(Event.COMPLETE, loadComplete); //when the
loader is done loading, call this function
loader.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR, loadIOError); //if the
file isn't found
loader.contentLoaderInfo.addEventListener(SecurityErrorEvent.SECURITY_ERROR,
loadSecurityError); //if the file isn't allowed to be loaded
```

3. Chargez le fichier désiré:

```
 loader.load(new URLRequest("image.png"));
```

4. Créez vos gestionnaires d'événements:

```
 function loadComplete(e:Event):void {
     //load complete
     //the loader is actually a display object itself, so you can just add it to the
display list
     addChild(loader)
     //or addChild(loader.content) to add the root content of what was loaded;
 }

 function loadIOError(e:IOErrorEvent):void {
     //the file failed to load,
 }

 function loadSecurityError(e:SecurityError):void {
     //the file wasn't allowed to load
 }
```

> Le chargement avec la classe Loader est asynchrone. Cela signifie qu'après avoir appelé `loader.load` l'application continuera à s'exécuter pendant le chargement du fichier. Le contenu de votre chargeur n'est disponible que lorsque le chargeur envoie l'événement `Event.COMPLETE` .

---

importations nécessaires:

```
import flash.display.Loader;
import flash.events.Event;
import flash.events.IOErrorEvent;
import flash.events.SecurityErrorEvent;
import flash.net.URLRequest;
```

## Chargement d'un fichier texte avec FileStream (moteur d'exécution AIR uniquement)

Un exemple simple sur la manière de lire un fichier texte UTF de manière synchrone.

```
import flash.filesystem.File;
import flash.filesystem.FileMode;
import flash.filesystem.FileStream;
```

---

```
//First, get a reference to the file you want to load
var myFile:File = File.documentsDirectory.resolvePath("lifestory.txt");

//Create a FileStream object
fileStream = new FileStream();

//open the file
fileStream.open(myFile, FileMode.READ);

//read the data and assign it to a local variable
var fileText:String = fileStream.readUTF();

//close the current filestream
fileStream.close();
```

Lire Chargement de fichiers externes en ligne: https://riptutorial.com/fr/actionscript-3/topic/1694/chargement-de-fichiers-externes

# Chapitre 3: Comprendre le "Erreur 1009: Impossible d'accéder à une propriété ou une méthode d'une référence d'objet null"

## Introduction

Une erreur 1009 est une erreur générale qui survient lorsque vous essayez de recevoir une valeur d'une variable ou d'une propriété dont la valeur est `null` . Les exemples fournis exposent divers cas où cette erreur survient, ainsi que des recommandations sur la manière d'atténuer cette erreur.

## Remarques

Le redoutable et souvent demandé "Erreur 1009: Impossible d'accéder à une propriété ou une méthode d'une référence d'objet NULL" est un signal que certaines des données apparaissent nulles, mais est essayé d'être utilisé comme un objet rempli. Il existe plusieurs types de problèmes qui peuvent provoquer ce comportement, et chacun doit être testé avec le code où l'erreur est apparue.

## Examples

### La scène n'est pas disponible

Parfois, les développeurs écrivent du code pour accéder à la `stage` , ou à la phase Flash, pour ajouter des écouteurs. Il peut fonctionner pour la première fois, puis soudainement ne plus fonctionner et produire l'erreur 1009. Le code en question peut même figurer sur la timeline, car c'est la première initiative à y ajouter du code, et de nombreux didacticiels existent encore. couche de code de ligne de temps pour placer le code.

```
public class Main extends MovieClip {
    public function Main() {
        stage.addEventListener(Event.ENTER_FRAME,update); // here
```

La raison pour laquelle ce code ne fonctionne pas est simple: un objet d'affichage est d'abord instancié, puis ajouté à la liste d'affichage, et alors qu'il est en dehors de la liste d'affichage, l' `stage` est nulle.

Pire si le code comme ça:

```
stage.addEventListener(Event.ENTER_FRAME,update); // here
```

est placé sur la timeline. Il peut même fonctionner pendant un certain temps, tandis que l'objet

`Main` est placé sur la scène via l'interface graphique. Ensuite, leur fichier SWF est chargé depuis un autre fichier SWF, et le code est brisé tout à coup. Cela se produit parce que les `Main` cadres de sont construits d'une manière différente lorsque le fichier SWF est chargé directement par le joueur et lorsque le chargement est traité de manière asynchrone. La solution consiste à utiliser `Event.ADDED_TO_STAGE` listener, à y placer tout le code qui adresse les étapes et à placer l'écouteur lui-même dans un fichier AS au lieu du scénario.

### Invalid typecast

```
function listener(e:Event):void {
    var m:MovieClip=e.target as MovieClip;
    m.x++;
}
```

Si un tel écouteur est associé à un objet qui n'est pas un descendant de `MovieClip` (par exemple, un `Sprite` ), le transtypage échouera et toute opération ultérieure avec son résultat lancera l'erreur 1009.

### Objet non confirmé

```
var a:Object;
trace(a); // null
trace(a.b); // Error 1009
```

Ici, une référence d'objet est déclarée, mais on ne lui attribue jamais de valeur, que ce soit avec `new` ou l'affectation d'une valeur non nulle. Demander ses propriétés ou sa méthode entraîne une erreur 1009.

### Expression multiniveau

```
x=anObject.aProperty.anotherProperty.getSomething().data;
```

Ici, tout objet avant le point peut être nul, et l'utilisation de méthodes qui renvoient des objets complexes ne fait qu'augmenter la complication du débogage de l'erreur nulle. Dans le pire des cas, si la méthode est sujette à des défaillances externes, par exemple, récupérer des données sur le réseau.

### Résultat de la fonction non traité

```
s=this.getChildByName("garbage");
if (s.parent==this) {...}
```

`getChildByName()` est l'une des nombreuses fonctions pouvant renvoyer null si une erreur s'est produite lors du traitement de son entrée. Par conséquent, si vous recevez un objet d'une fonction pouvant éventuellement renvoyer null, vérifiez d'abord la valeur null. Ici, une propriété est instantanément interrogée sans d'abord vérifier si `s` est nul, cela générera l'erreur 1009.

## Auditeur d'événement oublié

```
addEventListener(Event.ENTER_FRAME,moveChild);
function moveChild(e:Event):void {
    childMC.x++;
    if (childMC.x>1000) {
        gotoAndStop(2);
    }
}
```

Cet exemple déplacera le `childMC` (ajouté à `Main` au moment du design) mais lancera instantanément un 1009 dès que `gotoAndStop()` sera `childMC` , si cet `childMC` n'existe pas sur le frame 2. La raison principale est que chaque fois qu'une tête de lecture passe une image clé (un cadre qui n'hérite pas de l'ensemble d'objets précédent), soit en utilisant `gotoAndStop()` , `gotoAndPlay()` avec une image de séparation de l'image en cours par une image clé, soit en lecture normale si le fichier SWF est un animation, le contenu de l'image en cours est **détruit** et le nouveau contenu est créé en utilisant les données stockées à partir de l'interface graphique. Ainsi, si le nouveau cadre n'a pas d'enfant nommé `childMC` , la demande de propriété renverra null et 1009 sera renvoyé.

Le même principe s'applique si vous ajoutez deux écouteurs d'événement, mais n'en supprimez qu'un, ou ajoutez un écouteur à un objet, mais essayez de le supprimer d'un autre. L'appel `removeEventListener` ne vous avertira pas si un écouteur d'événement respectif n'a pas été associé à l'objet, lisez donc le code qui ajoute et supprime soigneusement les écouteurs d'événements.

Notez également que l'utilisation des objets `Timer` , appelant `setInterval()` et `setTimeout()` crée également des écouteurs d'événement, et ceux-ci doivent également être effacés correctement.

## Référence invalidée à un objet basé sur des images

Parfois, `gotoAndStop()` est appelé au milieu du code qui fait référence à certaines propriétés basées sur des images. Mais, **juste après la modification de l'image,** tous les liens vers les propriétés qui existaient sur l'image en cours sont invalidés, de sorte que tout traitement les impliquant doit être immédiatement arrêté.

Il existe deux scénarios généraux de ce type de traitement: Tout d'abord, une boucle ne se termine pas après l' `gotoAndStop()` à `gotoAndStop()` , comme ici:

```
for each (bullet in bullets) {
    if (player.hitTestObject(bullet)) gotoAndStop("gameOver");
}
```

Ici, une erreur 1009 signifie que le `player` MC a été détruit pendant le traitement de l'appel `gotoAndStop()` , mais que la boucle continue et renvoie le lien maintenant nul pour obtenir `hitTestObject()` . Si la condition `if (bullet.hitTestObject(player))` place, l'erreur serait # 2007 "Le paramètre hitTestObject ne doit pas être nul". La solution consiste à placer une déclaration de `return` juste après avoir appelé `gotoAndStop()` .

Le deuxième cas est constitué d'écouteurs multiples sur le même événement. Comme ça:

```
stage.addEventListener(Event.ENTER_FRAME,func1);
stage.addEventListener(Event.ENTER_FRAME,func2);
function func1(e:Event):void {
    if (condition()) {
        gotoAndStop(2);
    }
}
```

Ici, si `condition()` est vraie, le premier auditeur exécutera `gotoAndStop()` , mais le second écouteur sera toujours exécuté et si celui-ci référence des objets sur la trame, une erreur 1009 sera générée. La solution consiste à éviter plusieurs écouteurs sur un même événement, dans un seul objet, il est préférable d'avoir un écouteur qui gère toutes les situations sur cet événement et peut se terminer correctement si un changement de trame est nécessaire.

Lire Comprendre le "Erreur 1009: Impossible d'accéder à une propriété ou une méthode d'une référence d'objet null" en ligne: https://riptutorial.com/fr/actionscript-3/topic/2098/comprendre-le--erreur-1009--impossible-d-acceder-a-une-propriete-ou-une-methode-d-une-reference-d-objet-null-

# Chapitre 4: Conception d'applications réactive

## Examples

### Application sensible de base

```
package
{
    import flash.display.Sprite;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;
    import flash.events.Event;

    public class Main extends Sprite
    {
        //Document Class Main Constructor
        public function Main()
        {
            //Sometimes stage isn't available yet, so if not, wait for it before proceeding
            if (!stage) {
                addEventListener(Event.ADDED_TO_STAGE, stageReady);
            }else {
                stageReady();
            }
        }

        protected function stageReady(e:Event = null):void {
            //align the stage to the top left corner of the window/container
            stage.align = StageAlign.TOP_LEFT;
            //don't scale the content when the window resizes
            stage.scaleMode = StageScaleMode.NO_SCALE;

            //listen for when the window is resized
            stage.addEventListener(Event.RESIZE, stageResized);
        }

        protected function stageResized(e:Event):void {
            //use stage.stageWdith & stage.stageHeight to repostion and resize items
        }
    }
}
```

### Effectuer de longs processus et ne pas obtenir une application sans réponse

Il existe des situations où vous devez calculer quelque chose de vraiment important dans votre application Flash, sans interrompre l'expérience de l'utilisateur. Pour cela, vous devez concevoir votre processus long comme un processus en plusieurs étapes avec un état enregistré entre les itérations. Par exemple, vous devez effectuer une mise à jour en arrière-plan de nombreux objets internes, mais si vous souhaitez les mettre à jour tous en même temps avec un simple `for each (var o in objects) { o.update(); }`, Flash brièvement (ou pas aussi brièvement) ne répond plus à

l'utilisateur. Vous devez donc effectuer une ou plusieurs mises à jour par image.

```
private var processing:Boolean;      // are we in the middle of processing
private var lastIndex:int;           // where did we finish last time
var objects:Vector.<UpdatingObject>; // the total list of objects to update
function startProcess():Boolean {
    if (processing) return false; // already processing - please wait
    startProcessing=true;
    lastIndex=0;
    if (!hasEventListener(Event.ENTER_FRAME,iterate))
        addEventListener(Event.ENTER_FRAME,iterate); // enable iterating via listener
}
private function iterate(e:Event):void {
    if (!processing) return; // not processing - skip listener
    objects[lastIndex].update(); // perform a quantum of the big process
    lastIndex++; // advance in the big process
    if (lastIndex==objects.length) {
        processing=false; // finished, clear flag
    }
}
```

Le traitement avancé peut inclure l'utilisation de `getTimer()` pour vérifier le temps écoulé et autoriser une autre itération si le temps ne s'épuise pas, divisant `update()` en plusieurs fonctions si la mise à jour est trop longue liste d'objets à traiter, et beaucoup plus.

Lire Conception d'applications réactive en ligne: https://riptutorial.com/fr/actionscript-3/topic/1615/conception-d-applications-reactive

# Chapitre 5: Cycle de vie de la liste d'affichage

## Remarques

Les animations basées sur les images dans Flash et AIR implémentent le cycle de vie suivant:

- `Event.ENTER_FRAME` est envoyé
- Le code constructeur des objets d'affichage enfants est exécuté
- `Event.FRAME_CONSTRUCTED` est envoyé
- Les actions de cadre dans le symbole `MovieClip` sont exécutées
- Les actions de cadre dans les symboles `MovieClip` enfants sont exécutées
- `Event.EXIT_FRAME` est distribué
- `Event.RENDER` est envoyé

## Examples

### Ajouté et retiré du cycle de vie de la scène

```
package {
import flash.display.Sprite;
import flash.events.Event;

public class Viewport extends Sprite {

    /** Constructor */
    public function Viewport() {
        super();

        // Listen for added to stage event
        addEventListener(Event.ADDED_TO_STAGE, addedToStageHandler);
    }

    /** Added to stage handler */
    protected function addedToStageHandler(event:Event):void {
        // Remove added to stage event listener
        removeEventListener(Event.ADDED_TO_STAGE, addedToStageHandler);

        // Listen for removed from stage event
        addEventListener(Event.REMOVED_FROM_STAGE, removedFromStageHandler);
    }

    /** Removed from stage handler */
    protected function removedFromStageHandler(event:Event):void {
        // Remove removed from stage event listener
        removeEventListener(Event.REMOVED_FROM_STAGE, removedFromStageHandler);

        // Listen for added to stage event
        addEventListener(Event.ADDED_TO_STAGE, addedToStageHandler);
    }

    /** Dispose */
    public function dispose():void {
```

```
        // Remove added to stage event listener
        removeEventListener(Event.ADDED_TO_STAGE, addedToStageHandler);

        // Remove removed from stage event listener
        removeEventListener(Event.REMOVED_FROM_STAGE, removedFromStageHandler);
    }

}
}
```

Lire Cycle de vie de la liste d'affichage en ligne: https://riptutorial.com/fr/actionscript-3/topic/1877/cycle-de-vie-de-la-liste-d-affichage

# Chapitre 6: Dessin Bitmaps

## Examples

### Dessiner un objet d'affichage dans les données bitmap

Une fonction d'assistance pour créer une copie bitmap d'un objet. Cela peut être utilisé pour convertir des objets vectoriels, du texte ou des sprites imbriqués complexes en un bitmap aplati.

```
function makeBitmapCopy(displayObj:IBitmapDrawable, transparent:Boolean = false, bgColor:uint
= 0x00000000, smooth:Boolean = true):Bitmap {

    //create an empty bitmap data that matches the width and height of the object you wish to
draw
    var bmd:BitmapData = new BitmapData(displayObj.width, displayObj.height, transparent,
bgColor);

    //draw the object to the bitmap data
    bmd.draw(displayObj, null, null, null, null, smooth);

    //assign that bitmap data to a bitmap object
    var bmp:Bitmap = new Bitmap(bmd, "auto", smooth);

    return bmp;
}
```

Usage:

```
var txt:TextField = new TextField();
txt.text = "Hello There";

var bitmap:Bitmap = makeBitmapCopy(txt, true); //second param true to keep transparency
addChild(bitmap);
```

### Dessine un objet d'affichage avec les coordonnées du point d'enregistrement

```
    public function drawDisplayObjectUsingBounds(source:DisplayObject):BitmapData {
        var bitmapData:BitmapData;//declare a BitmapData
        var bounds:Rectangle = source.getBounds(source);//get the source object actual size
        //round bounds to integer pixel values (to aviod 1px stripes left off)
        bounds = new Rectangle(Math.floor(bounds.x), Math.floor(bounds.y),
Math.ceil(bounds.width), Math.ceil(bounds.height));

        //to avoid Invalid BitmapData error which occures if width or height is 0
        //(ArgumentError: Error #2015)
        if((bounds.width>0) && (bounds.height>0)){
            //create a BitmapData
            bitmapData = new BitmapData(bounds.width, bounds.height, true, 0x00000000);

            var matrix:Matrix = new Matrix();//create a transform matrix
            //translate if to fit the upper-left corner of the source
            matrix.translate(-bounds.x, -bounds.y);
            bitmapData.draw(source, matrix);//draw the source
```

```
            return bitmapData;//return the result (exit point)
        }
        //if no result is created - return an empty BitmapData
        return new BitmapData(1, 1, true, 0x00000000);
    }
```

Une note de côté: Pour que `getBounds()` renvoie des valeurs valides, l'objet doit avoir accès à une étape au moins une fois, sinon les valeurs sont fausses. Un code peut être ajouté pour s'assurer que la `source` transmise a une étape et, dans le cas contraire, il peut être ajouté à la phase puis supprimé à nouveau.

## Animation d'une feuille de sprite

Une feuille de sprite par définition est une image bitmap contenant une certaine animation. Les anciens jeux utilisent une feuille de sprite de type grille, c'est-à-dire que chaque image occupe une région égale et que les cadres sont alignés par les bords pour former un rectangle, avec probablement des espaces inoccupés. Plus tard, afin de minimiser la taille du bitmap, les feuilles de sprite commencent à être "empaquetées" en supprimant les espaces blancs autour du rectangle qui contient chaque image, mais chaque image reste un rectangle pour simplifier les opérations de copie.

Pour animer une feuille de sprite, deux techniques peuvent être utilisées. Tout d'abord, vous pouvez utiliser `BitmapData.copyPixels()` pour copier une certaine région de votre feuille d'image-objet sur une image `Bitmap` affichée, produisant ainsi un personnage animé. Cette approche est préférable si vous utilisez un seul `Bitmap` affiché qui héberge l'intégralité de l'image.

```
var spriteSheet:BitmapData;
var frames:Vector.<Rectangle>; // regions of spriteSheet that represent frames
function displayFrameAt(frame:int,buffer:BitmapData,position:Point):void {
    buffer.copyPixels(spriteSheet,frames[frame],position,null,null,true);
}
```

La seconde technique peut être utilisée si vous avez beaucoup de `Sprite` ou de `Bitmap` sur la liste d'affichage et qu'ils partagent la même feuille de sprite. Ici, le tampon cible n'est plus un objet unique, mais chaque objet possède son propre tampon. La stratégie appropriée consiste donc à manipuler la propriété `bitmapData` des bitmaps. Avant de le faire, cependant, la feuille de sprite doit être découpée dans des cadres individuels.

```
public class Stuff extends Bitmap {
    static var spriteSheet:Vector.<BitmapData>;
    function displayFrame(frame:int) {
        this.bitmapData=spriteSheet[frame];
    }
    // ...
}
```

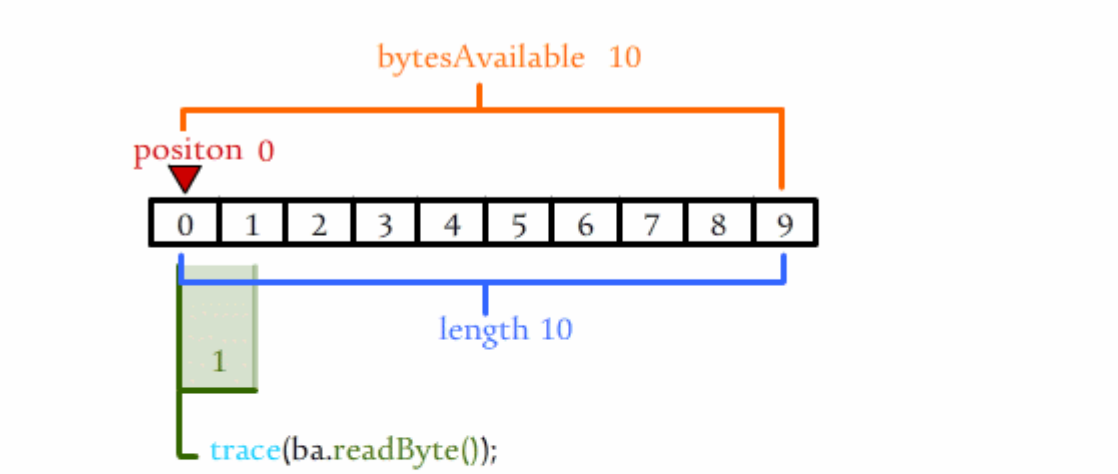
Lire Dessin Bitmaps en ligne: https://riptutorial.com/fr/actionscript-3/topic/2814/dessin-bitmaps

# Chapitre 7: Données binaires

## Examples

### Forme de lecture ByteArray à travers l'interface IDataInput.

L'animation ci-dessous montre ce qui se passe lorsque vous utilisez les méthodes d'interface IDataInput pour accéder au formulaire de données ByteArray et aux autres classes qui implémentent cette interface.




Lire Données binaires en ligne: https://riptutorial.com/fr/actionscript-3/topic/9464/donnees-binaires

# Chapitre 8: Envoi et réception de données à partir de serveurs

## Examples

### Faire une demande à partir de Flash

Les classes `URLRequest` et `URLLoader` fonctionnent ensemble pour faire des demandes à partir de Flash vers des ressources externes. Le `URLRequest` définit des informations sur la demande , par exemple le corps de la demande et le type de méthode de demande, et les `URLLoader` références ceci pour effectuer la demande réelle et fournir un moyen de notification quand une réponse est reçue à partir de la ressource.

Exemple:

```
var request:URLRequest = new URLRequest('http://stackoverflow.com');
var loader:URLLoader = new URLLoader();

loader.addEventListener(Event.COMPLETE, responseReceived);
loader.load(request);

function responseReceived(event:Event):void {
    trace(event.target.data); // or loader.data if you have reference to it in
                              // this scope.
}
```

### Ajout de variables à votre demande

La classe `URLVariables` vous permet de définir des données à envoyer avec une `URLRequest` .

Exemple:

```
var variables:URLVariables = new URLVariables();

variables.prop = "hello";
variables.anotherProp = 10;

var request:URLRequest = new URLRequest('http://someservice.com');
request.data = variables;
```

Vous pouvez envoyer la demande via un `URLLoader` ou ouvrir l'URL de la demande avec les variables associées à la chaîne de requête à l'aide de `navigateToURL` .

### Modifier la méthode HTTP (GET, POST, PUT, etc.)

La classe `URLRequestMethod` contient des constantes pour les différents types de requêtes que vous pouvez créer. Ces constantes doivent être allouées à la propriété de `method` `URLRequest` :

```
var request:URLRequest = new URLRequest('http://someservice.com');
request.method = URLRequestMethod.POST;
```

Notez que seuls `GET` et `POST` sont disponibles en dehors du moteur d'exécution AIR.

## Mes données de réponse sont toujours nulles, que signifie "asynchrone"?

Lorsque Flash effectue une demande de données depuis une source externe, cette opération est *asynchrone* . L'explication la plus simple de ce que cela signifie, c'est que les données sont chargées "en arrière-plan" et déclenche le gestionnaire d'événements que vous `Event.COMPLETE` à `Event.COMPLETE` lorsqu'il est reçu. Cela peut arriver à n'importe quel moment de la vie de votre application.

Vos données **ne** seront **PAS** disponibles immédiatement après avoir appelé `load()` sur votre `URLLoader` . Vous **devez** joindre un écouteur d'événement pour `Event.COMPLETE` et interagir avec la réponse.

```
var request:URLRequest = new URLRequest('http://someservice.com');
var loader:URLLoader = new URLLoader();

loader.addEventListener(Event.COMPLETE, responseReceived);
loader.load(request);

trace(loader.data); // Will be null.

function responseReceived(event:Event):void {
    trace(loader.data); // Will be populated with the server response.
}

trace(loader.data); // Will still be null.
```

Vous ne pouvez pas contourner cela avec de petites astuces comme l'utilisation de `setTimeout` ou similaire:

```
setTimeout(function() {
    trace(loader.data); // Will be null if the data hasn't finished loading
                        // after 1000ms (which you can't guarantee).
}, 1000);
```

## Requêtes interdomaines

Flash ne chargera pas de données d'un domaine autre que celui sur lequel votre application s'exécute, à moins que ce domaine ne dispose d'une stratégie de crossdomain XML dans la racine du domaine (par exemple, `http://somedomain.com/crossdomain.xml` ) ou dans un emplacement que vous peut cibler avec `Security.loadPolicyFile()` . Le fichier crossdomain.xml est l'endroit où vous pouvez spécifier des domaines capables de demander à votre serveur des données provenant d'une application Flash.

Exemple du crossdomain.xml le *plus permissif* :

```
<?xml version="1.0" ?>
```

```
<cross-domain-policy>
  <allow-access-from domain="*"/>
  <allow-http-request-headers-from domain="*" headers="*"/>
</cross-domain-policy>
```

> Notez que cet exemple **ne doit pas être utilisé dans des environnements de production** , utilisez une instance plus restrictive.

Un crossdomain.xml spécifique plus restrictif ressemblera à ceci par exemple:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.macromedia.com/xml/dtds/cross-domain-
policy.dtd">
<cross-domain-policy>
    <site-control permitted-cross-domain-policies="master-only" />

    <allow-access-from domain="*.domain.com" to-ports="80,843,8011" />
    <allow-access-from domain="123.123.123.123" to-ports="80,843,8011" />
</cross-domain-policy>
```

Ressources:

- La spécification du fichier de stratégie crossdomain .

Lire Envoi et réception de données à partir de serveurs en ligne: https://riptutorial.com/fr/actionscript-3/topic/1893/envoi-et-reception-de-donnees-a-partir-de-serveurs

# Chapitre 9: Génération de valeur aléatoire

## Examples

### Nombre aléatoire entre 0 et 1

```
Math.random();
```

produit un nombre aléatoire uniformément réparti entre 0 (inclus) et 1 (exclusif)

Exemple de sortie:

- 0.22282187035307288
- 0.3948539895936847
- 0.9987191134132445

### Nombre aléatoire entre les valeurs min et max

```
function randomMinMax(min:Number, max:Number):Number {
    return (min + (Math.random() * Math.abs(max - min)));
}
```

Cette fonction est appelée en passant une plage de valeurs minimales et maximales.

Exemple:

```
randomMinMax(1, 10);
```

Exemple de sorties:

- 1.661770915146917
- 2.5521070677787066
- 9.436270965728909

### Angle aléatoire, en degrés

```
function randomAngle():Number {
    return (Math.random() * 360);
}
```

Exemple de sorties:

- 31.554428357630968
- 230.4078639484942
- 312.7964010089636

## Valeur aléatoire d'un tableau

En supposant que nous avons un tableau `myArray` :

```
var value:* = myArray[int(Math.random() * myArray.length)];
```

Notez que nous utilisons `int` pour `Math.random()` le résultat de `Math.random()` en int, car les valeurs comme `2.4539543` ne seraient pas un index de tableau valide.

## Point aléatoire à l'intérieur d'un cercle

Définissez d'abord le rayon du cercle et son centre:

```
var radius:Number = 100;
var center:Point = new Point(35, 70);
```

Ensuite, générez un angle aléatoire en *radians à* partir du centre:

```
var angle:Number = Math.random() * Math.PI * 2;
```

Ensuite, générez un rayon effectif du point retourné, de sorte qu'il se trouve dans un `radius` donné. Un simple `Math.random()*radius` ne le fera pas, car avec cette distribution, les points produits se retrouveront dans le cercle intérieur à moitié rayon, mais le carré de ce cercle est le quart de l'original. Pour créer une distribution correcte, la fonction devrait être comme ceci:

```
var rad:Number=(Math.random()+Math.random())*radius; // yes, two separate calls to random
if (rad>radius) { rad=2*radius-rad; }
```

Cette fonction produit une valeur dont la fonction de probabilité augmente linéairement de 0 à zéro au maximum au `radius` . Cela se produit parce qu'une somme de valeurs aléatoires a une fonction de densité de probabilité égale à la convolution de toutes les fonctions de densité individuelles des valeurs aléatoires. Il s'agit de mathématiques étendues pour une personne de niveau moyen, mais une sorte de GIF est présentée pour dessiner un graphique de la fonction de convolution de deux fonctions de densité de distribution uniforme expliquées sous la forme de « signaux de boîte ». L'opérateur `if` plie la fonction résultante sur son maximum, ne laissant qu'un graphique en dents de scie.

Cette fonction est sélectionnée car le carré d'une bande de cercle située entre `radius=r` et `radius=r+dr` augmente linéairement avec l'augmentation de `r` et la très petite constante `dr` sorte que `dr*dr<<r` . Par conséquent, la quantité de points générés à proximité du centre est inférieure à la quantité de points générés au bord du cercle avec la même marge que le rayon de la zone centrale est inférieur au rayon de tout le cercle. Donc, globalement, les points sont répartis uniformément dans tout le cercle.

Maintenant, obtenez votre position au hasard:

```
var result:Point = new Point(
```

```
    center.x + Math.cos(angle) * rad,
    center.y + Math.sin(angle) * rad
);
```

Pour obtenir un point aléatoire sur le cercle (sur le bord du cercle d'un rayon donné), utilisez le `radius` au lieu de `rad` .

PS: L'exemple a fini par être surchargé par l'explication des maths.

### Angle aléatoire, en radians

```
function randomAngleRadians():Number
{
    return Math.random() * Math.PI * 2;
}
```

Exemple de sorties:

- 5.490068569213088
- 3.1984284719180205
- 4.581117863808207

### Détermination du succès d'une opération "pourcentage de chance"

Si vous devez effectuer un jet pour un `true` ou un `false` dans une situation "x% chance", utilisez:

```
function roll(chance:Number):Boolean {
    return Math.random() >= chance;
}
```

Utilisé comme:

```
var success:Boolean = roll(0.5); // True 50% of the time.
var again:Boolean = roll(0.25); // True 25% of the time.
```

### Créer une couleur aléatoire

Pour obtenir *une* couleur aléatoire:

```
function randomColor():uint
{
    return Math.random() * 0xFFFFFF;
}
```

Si vous avez besoin de plus de contrôle sur les canaux rouge, vert et bleu:

```
var r:uint = Math.random() * 0xFF;
var g:uint = Math.random() * 0xFF;
var b:uint = Math.random() * 0xFF;
```

```
var color:uint = r << 16 | g << 8 | b;
```

Vous pouvez spécifier ici votre propre plage pour `r` , `g` et `b` (cet exemple va de 0 à 255).

## Boucle aléatoire dans l'alphabet

```
var alphabet:Vector.<String> = new <String>[ "A", "B", "C", "D", "E", "F", "G",
                                             "H", "I", "J", "K", "L", "M", "N",
                                             "O", "P", "Q", "R", "S", "T", "U",
                                             "V", "W", "X", "Y", "Z" ];

while (alphabet.length > 0)
{
    var letter:String = alphabet.splice(int(Math.random() *
                                            alphabet.length), 1)[0];
    trace(letter);
}
```

Exemple de sortie:

V, M, F, E, D, U, S, L, X, K, Q, H, A, I, W, N, P, Y, J, C, T, O, R, Z

## Randomize Un tableau

```
var alphabet:Array = [ "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N",
"O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z" ];

for (var i:int=alphabet.length-1;i>0;i--) {
    var j:int=Math.floor(Math.random()*(i+1));
    var swap=alphabet[j];
    alphabet[j]=alphabet[i];
    alphabet[i]=swap;
}
trace(alphabet);
```

Exemple de sortie

A, K, T, F, V, X, Y, G UNE

Cette méthode est connue sous le nom de mélange de tableaux Fisher-Yates .

Lire Génération de valeur aléatoire en ligne: https://riptutorial.com/fr/actionscript-3/topic/1441/generation-de-valeur-aleatoire

# Chapitre 10: Les bases du développement de jeux

## Introduction

[! [entrer la description de l'image ici] [1]] [1] **bases** du développement de jeux. ---------- *Remarque* : cet ensemble de didacticiels / articles contient de nombreux concepts pouvant être fournis en tant que sujets séparés auparavant. nous devons les rafraîchir dans l'esprit et apprendre un peu à implémenter les parties les plus critiques d'un jeu vidéo via actionscript-3. [1]: https://i.stack.imgur.com/CUIsz.png

## Examples

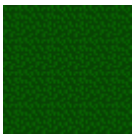
### personnage isométrique animant + mouvement

① les concepts utilisés dans cet exemple:

| Classe | Usage |
|---|---|
| `URLRequest` + `Loader` + `Event` | Chargement de la carte de l'atlas (sprite) à partir du chemin externe. |
| `BitmapData` + `Sprite` + `beginBitmapFill` + `Matrix` + `stageWidth & stageHeight` | dessiner des ressources chargées sur bitmapdata, utiliser des bitmaps en mosaïque, dessiner avec transformation. |
| `MovieClip` + `scrollRect` + `Bitmap` + `Rectangle` | création et cliping MovieClip en utilisant Bitmap comme scénario. |
| `KeyboardEvent` | détecter les entrées utilisateur |
| `Event.EXIT_FRAME` | implémenter la fonction de boucle de jeu |

---

② Ressources: **(pas de permission pour utiliser ces ressources à des fins commerciales)**



## ③ Code et commentaires:

> **Remarque:** FPS 15 Utilisé pour ce didacticiel, il est recommandé, mais si nécessaire, vous devez modifier une partie du code par vous-même.

dans un premier temps, nous devons télécharger nos ressources à partir des URL externes.

```
const src_grass_tile_url:String = "https://i.stack.imgur.com/sjJFS.png";
const src_character_atlas_url:String = "https://i.stack.imgur.com/B7ztZ.png";

var loader:Loader = new Loader();
loader.contentLoaderInfo.addEventListener(Event.COMPLETE, setGround);
loader.load(new URLRequest(src_grass_tile_url));
```

---

setGround sera appelé une fois que `src_grass_tile_url` sera chargé et prêt à être utilisé. en mettant en œuvre setGround pour obtenir des ressources et dessiner comme arrière-plan du jeu

```
function setGround(e:Event):void {
    /* drawing ground */
    /* loader is a displayObject, so we can simply draw it into the bitmap data*/
    /* create an instance of Bitmapdata with same width and height as our window*/
    /* (also set transparent to false because grass image, does not contains any transparent
pixel) */
    var grass_bmd:BitmapData = new BitmapData(loader.width, loader.height, false, 0x0);
    /* time to draw */
    grass_bmd.draw(loader); // drawing loader into the bitmapData
    /* now we have to draw a tiled version of grass_bmd inside a displayObject Sprite to
displaying
       BitmapData on stage */
    var grass_sprite:Sprite = new Sprite();
    // for drawing a bitmap inside sprite, we must use <beginBitmapFill> with graphic property
of the sprite
    // then draw a full size rectangle with that Fill-Data
    // there is a repeat mode argument with true default value so we dont set it true again.
    // use a matrix for scalling grass Image during draw to be more cute!
    var mx:Matrix = new Matrix();
    mx.scale(2, 2);
    grass_sprite.graphics.beginBitmapFill(grass_bmd, mx);
    grass_sprite.graphics.drawRect(0, 0, stage.stageWidth, stage.stageHeight);
    // now add sprite to displayobjectcontainer to be displayed
    stage.addChild(grass_sprite);

    // well done, ground is ready, now we must initialize our character
    // first, load its data, i just re-use my loader for loading new image, but with another
complete handler (setCharacter)
    // so remove existing handler, then add new one
    loader.contentLoaderInfo.removeEventListener(Event.COMPLETE, setGround);
    loader.contentLoaderInfo.addEventListener(Event.COMPLETE, setCharacter);
    loader.load(new URLRequest(src_character_atlas_url));
}
```

le code est très bien commenté, après avoir fait avec le sol, son temps pour mettre en œuvre le caractère. character contient également une ressource qui doit être chargée de la même manière. Donc, à la fin de `setGround` nous nous dirigeons vers `setCharacter` qui est un autre rappel complet.

```
function setCharacter(e:Event):void {
    // let assuming that what is really character!
    // a set of images inside a single image!
    // that images are frames of our character (also provides movement for different
directions)
    // first load this
    var character_bmd:BitmapData = new BitmapData(loader.width, loader.height, true, 0x0); //
note character is transparent
    character_bmd.draw(loader);
    // take a look at sprite sheet, how many frames you see?
    // 42 frames, so we can get width of a single frame
    const frame_width:uint = character_bmd.width / 42; // 41 pixels
    // as i show you above, to displaying a BitmapData, we have to draw it using a
DisplayObject,
    // another way is creating a Bitmap and setting its bitmapdata
    var character_bmp:Bitmap = new Bitmap(character_bmd);
    // but its not enough yet, a movieClip is necessary to cliping and animating this bitmap
(as a child of itself)
    var character_mc:MovieClip = new MovieClip();
    character_mc.addChild(character_bmp);
    character_bmp.name = "sprite_sheet"; // setting a name to character_bmp, for future
accessing
    character_mc.scrollRect = new Rectangle(0, 0, frame_width, character_bmd.height); //
cliping movieclip, to dusplaying only one frame
    character_mc.name = "character"; // setting a name to character_mc, for future accessing
    stage.addChild(character_mc); // adding it to stage.
    // well done, we have a character, but its static yet! 2 steps remaining. 1 controlling 2
animating
    // at first setting a control handler for moving character in 8 directions.
    stage.addEventListener(KeyboardEvent.KEY_DOWN, keyDown);
    stage.addEventListener(KeyboardEvent.KEY_UP, keyUp);
}
```

maintenant le personnage est prêt à contrôler. son affiché bien et prêt à contrôler. Donc, les événements clavier attachés et l'écoute des touches fléchées comme suit:

```
// we storing key stats inside <keys> Object
var keys:Object = {u:false, d:false, l:false, r:false};
function keyDown(e:KeyboardEvent):void {
    switch (e.keyCode) {
        case 38: //up
            keys.u = true;
            break;
        case 40: //down
            keys.d = true;
            break;
        case 37: //left
            keys.l = true;
            break;
        case 39: //right
            keys.r = true;
            break;
    }
}
```

```
function keyUp(e:KeyboardEvent):void {
    switch (e.keyCode) {
        case 38: //up
            keys.u = false;
            break;
        case 40: //down
            keys.d = false;
            break;
        case 37: //left
            keys.l = false;
            break;
        case 39: //right
            keys.r = false;
            break;
    }
}
```

---

`keys:Object` stocke la variable booléenne de 4 pour chaque touche fléchée, le processus de déplacement doit être effectué à l'intérieur de la fonction update (Loop) du jeu, nous devons donc lui transmettre des statistiques de clavier. permet d'implémenter la fonction de **boucle** .

```
// initialize game Loop function for updating game objects
addEventListener(Event.EXIT_FRAME, loop);

// speed of character movement
const speed:Number = 5;
// this function will be called on each frame, with same rate as your project fps
function loop(e:Event):void {
    if (keys.u) stage.getChildByName("character").y -= speed;
    else if (keys.d) stage.getChildByName("character").y += speed;
    if (keys.l) stage.getChildByName("character").x -= speed;
    else if (keys.r) stage.getChildByName("character").x += speed;
}
```

la vitesse est une constante d'assistance, définit la vitesse du caractère. Le code ci-dessus présente un mouvement simple à 8 directions avec cette priorité basse: `Up > Down Left > Right` . ainsi, si vous appuyez simultanément sur les flèches haut et bas, le caractère ne se déplace que vers le *haut* (pas de gel).

**bien joué!!! il ne reste qu'une étape, l'animation, la partie la plus importante de ce tutoriel**

Qu'est-ce que l'animation? un ensemble d'images clés contenant au moins une image
permet de créer notre objet keyframs, qui contient le nom des images clés
et aussi quelques données sur le début et la fin de l'image de cette image clé
**Notez que** dans les jeux isométriques, chaque image clé contient 8 directions (peut être réduite à 5 avec l'utilisation du retournement)

```
var keyframs:Object = {
    idle: {up:[0,0], up_right:[1,1], right:[2,2], down_right:[3,3], down:[4,4]}, // [2,2]
means start frame is 2 and end frame is 2
    run: {up:[5,10], up_right:[11,16], right:[17,22], down_right:[23,28], down:[29,34]}
};
```

nous devrions ignorer les images restantes, cet exemple ne fournit que l'animation inactive et

exécutée
Par exemple, la première image de l'animation inactive avec la direction droite est:
<keyframs.idle.right [0]>
permet maintenant la mise en œuvre de la fonction Animator

```
var current_frame:uint;
function animate(keyframe:Array):void {
    // how it works
    // just called with a keyframe with direction (each frame),
    // if keyframe is what is already playing, its just moved to next frame and got updated
(or begning frame for loop)
    // other wise, just moved to begining frame of new keyframe
    if (current_frame >= keyframe[0] && current_frame <= keyframe[1]) { // check if in bound
        current_frame++;
        if (current_frame > keyframe[1]) // play back if reached
            current_frame = keyframe[0];
    } else {
        current_frame = keyframe[0]; // start new keyframe from begining
    }
    // moving Bitmap inside character MovieClip
    var character:MovieClip = stage.getChildByName("character") as MovieClip;
    var sprite_sheet:Bitmap = character.getChildByName("sprite_sheet") as Bitmap;
    sprite_sheet.x = -1 * current_frame * character.width;
}
```

lire les commentaires de la fonction ci-dessus, cependant le travail principal de cette fonction déplace le `Bitmap` *sprite_sheet à l'* intérieur du *caractère* `MovieClip` .

Nous savons que chaque mise à jour doit être effectuée à l'intérieur de la fonction Loop. Nous allons donc appeler cette fonction à partir de Loop avec les images clés associées. c'est la fonction de boucle mise à jour:

```
// speed of character movement
const speed:Number = 8;
var last_keyStat:Object = {u:false, d:false, l:false, r:false}; // used to getting a backup of
previous keyboard stat for detecting correct idle direction
// this function will be called on each frame, with same rate as your project fps
function loop(e:Event):void {
    if (keys.u) stage.getChildByName("character").y -= speed;
    else if (keys.d) stage.getChildByName("character").y += speed;
    if (keys.l) stage.getChildByName("character").x -= speed;
    else if (keys.r) stage.getChildByName("character").x += speed;

    // animation detection
    if (keys.u && keys.l) { animate(keyframs.run.up_right); flip(true); }
    else if (keys.u && keys.r) { animate(keyframs.run.up_right); flip(false); }
    else if (keys.d && keys.l) { animate(keyframs.run.down_right); flip(true); }
    else if (keys.d && keys.r) { animate(keyframs.run.down_right); flip(false); }
    else if (keys.u) { animate(keyframs.run.up); flip(false); }
    else if (keys.d) { animate(keyframs.run.down); flip(false); }
    else if (keys.l) { animate(keyframs.run.right); flip(true); }
    else if (keys.r) { animate(keyframs.run.right); flip(false); }
    else {
        // if character dont move, so play idle animation
        // what is the best practice to detecting idle direction?
        // my suggestion is to sotring previous keyboard stats to determining which idle
direction is correct
```

```
        // do any better thing if possible (absolutely is possible)
        // i just simply copy it from above, and replaced (keys) with (last_keyStat) and (run)
with (idle)
        if (last_keyStat.u && last_keyStat.l) { animate(keyframs.idle.up_right); flip(true); }
        else if (last_keyStat.u && last_keyStat.r) { animate(keyframs.idle.up_right);
flip(false); }
        else if (last_keyStat.d && last_keyStat.l) { animate(keyframs.idle.down_right);
flip(true); }
        else if (last_keyStat.d && last_keyStat.r) { animate(keyframs.idle.down_right);
flip(false); }
        else if (last_keyStat.u) { animate(keyframs.idle.up); flip(false); }
        else if (last_keyStat.d) { animate(keyframs.idle.down); flip(false); }
        else if (last_keyStat.l) { animate(keyframs.idle.right); flip(true); }
        else if (last_keyStat.r) { animate(keyframs.idle.right); flip(false); }
    }
    // update last_keyStat backup
    last_keyStat.u = keys.u;
    last_keyStat.d = keys.d;
    last_keyStat.l = keys.l;
    last_keyStat.r = keys.r;
}
```

Lisez les commentaires, nous détectons simplement une véritable image-clé grâce aux statistiques du clavier. alors aussi faire la même chose pour détecter une animation inactive. pour les animations inactives, nous n'avons aucune entrée clé à utiliser pour détecter quel caractère de direction est activé, de sorte qu'une variable d'assistance simple pourrait être utile pour stocker l'état précédent du clavier (last_keyStat).

---

il y a aussi une nouvelle fonction `flip` qui est une autre fonction d'aide utilisée pour simuler les animations manquantes (à gauche + up_left + down_left) également cette fonction fait quelques corrections qui sont commentées ci-dessous:

```
// usage of flip function is because of Movieclip registration point, its a fix
// as the registration point of MovieClip is not placed in center, when flipping animation
(for non existing directions inside spritesheet)
// character location changes with an unwanted value equal its width, so we have to prevent
this and push it back or forward during flip
function flip(left:Boolean):void {
    var character:MovieClip = stage.getChildByName("character") as MovieClip;
    if (left) {
        if (character.scaleX != -1) {
            character.scaleX = -1;
            character.x += character.width; // comment this line to see what happen without
this fix
        }
    } else {
        if (character.scaleX != 1) {
            character.scaleX = 1;
            character.x -= character.width; // comment this line to see what happen without
this fix
        }
    }
}
```

**notre travail se termine ici. Un remerciement spécial pour l'éditeur qui rend ce tutoriel plus facile à contrôler. Aussi, voici une démonstration en direct de ce tutoriel et un lien externe**

**de code complet.**

## ④ Références externes:

- **code complet**
- **Démo en direct**

Lire Les bases du développement de jeux en ligne: https://riptutorial.com/fr/actionscript-3/topic/8237/les-bases-du-developpement-de-jeux

# Chapitre 11: Les types

## Examples

### Casting de type

Le casting de type se fait soit avec l'opérateur `as` :

```
var chair:Chair = furniture as Chair;
```

Ou en encapsulant la valeur dans `Type()` :

```
var chair:Chair = Chair(furniture);
```

Si la distribution échoue avec `as` , le résultat de cette distribution est `null` . Si la `TypeError` échoue en encapsulant dans `Type()` , une `TypeError` est `TypeError` .

### Le type de fonction

Les fonctions sont du type `Function` :

```
function example():void { }
trace(example is Function); // true
```

Ils peuvent être référencés par d'autres variables avec le type `Function` :

```
var ref:Function = example;
ref(); // ref.call(), ref.apply(), etc.
```

Et ils peuvent être passés en arguments pour les paramètres dont le type est `Function` :

```
function test(callback:Function):void {
    callback();
}

test(function() {
    trace('It works!');
}); // Output: It works!
```

### Le type de classe

Les références aux déclarations de classe sont typées `Class` :

```
var spriteClass:Class = Sprite;
```

Vous pouvez utiliser des variables typées `Class` pour instancier des instances de cette classe:

```
var sprite:Sprite = new spriteClass();
```

Cela peut être utile pour transmettre un argument de type `Class` à une fonction susceptible de créer une instance de la classe fournie:

```
function create(type:Class, x:int, y:int):* {
    var thing:* = new type();

    thing.x = x;
    thing.y = y;

    return thing;
}

var sprite:Sprite = create(Sprite, 100, 100);
```

## Types d'annotation

Vous pouvez indiquer au compilateur le type d'une valeur en l'annotant avec `:Type` :

```
var value:int = 10; // A property "value" of type "int".
```

Les paramètres de fonction et les types de retour peuvent également être annotés:

```
// This function accepts two ints and returns an int.
function sum(a:int, b:int):int {
    return a + b;
}
```

`TypeError` d'attribuer une valeur avec un type incompatible, vous obtiendrez une `TypeError` :

```
var sprite:Sprite = 10; // 10 is not a Sprite.
```

## Vérification des types

Vous pouvez utiliser l'opérateur `is` pour valider si une valeur est d'un certain type:

```
var sprite:Sprite = new Sprite();

trace(sprite is Sprite); // true
trace(sprite is DisplayObject); // true, Sprite inherits DisplayObject
trace(sprite is IBitmapDrawable); // true, DisplayObject implements IBitmapDrawable
trace(sprite is Number); // false
trace(sprite is Bitmap); // false, Bitmap inherits DisplayObject
                         // but is not inherited by Sprite.
```

Il existe également une `instanceof` opérateur (obsolète) qui fonctionne presque identique à `is` sauf qu'elle *renvoie `false` lors de la vérification des interfaces implémentées* et des types int / uint.

Le `as` opérateur peut aussi utilisé tout comme `is` l' opérateur. Cela est particulièrement utile si vous utilisez un IDE intelligent comme FlashDevelop qui vous donnera une liste de toutes les propriétés

possibles du type d'objet explicite. Exemple:

```
for (var i:int = 0; i < a.length; i++){
    var d:DisplayObject = a[i] as DisplayObject;
    if (!d) continue;
    d.//get hints here
    stage.addChild(d);
}
```

Pour obtenir le même effet `is` que vous écririez (sligthly moins pratique):

```
for (var i:int = 0; i < a.length; i++){
    if (a[i] is DisplayObject != true) continue;
    var d:DisplayObject = a[i] as DisplayObject;
    stage.addChild(d);
}
```

Gardez simplement à l'esprit que lors de la vérification des coditions avec l'opérateur `as` , la valeur donnée sera convertie en un type spécifié et le résultat de cette opération sera vérifié si elle n'est pas fausse.

```
if(false as Boolean) trace("This will not be executed");
if(false as Boolean != null) trace("But this will be");
```

Le tableau ci-dessous montre quelques valeurs et types de base avec le résultat des opérateurs de type. Les cellules vertes seront évaluées à true, rouge à false et greay provoquera des erreurs de compilation / runtime.

| | | Sprite | IBitmapDrawable | Object | Class | uint | int |
|---|---|---|---|---|---|---|---|
| new Sprite() | as | [object Sprite] | [object Sprite] | [object Sprite] | null | null | null |
| | is | true | true | true | false | false | false |
| | instanceof | true | false | true | false | false | false |
| | (<columnName>) | error | error | error | error | error | error |
| true | as | null | null | true | null | null | null |
| | is | false | false | true | false | false | false |
| | instanceof | false | false | true | false | false | false |
| | (<columnName>) | error | error | error | error | error | error |
| false | as | null | null | false | null | null | null |
| | is | false | false | true | false | false | false |
| | instanceof | false | false | true | false | false | false |
| | (<columnName>) | error | error | error | error | error | error |
| 0.3 | as | null | null | 0.3 | null | null | null |
| | is | false | false | true | false | false | false |
| | instanceof | false | false | true | false | false | false |
| | (<columnName>) | error | error | error | error | error | error |
| 1 | as | null | null | 1 | null | 1 | 1 |
| | is | false | false | true | false | true | true |
| | instanceof | false | false | true | false | false | false |
| | (<columnName>) | error | error | error | error | error | error |
| -1 | as | null | null | -1 | null | null | -1 |
| | is | false | false | true | false | false | true |
| | instanceof | false | false | true | false | false | false |
| | (<columnName>) | error | error | error | error | error | error |
| NaN | as | null | null | NaN | null | null | null |
| | is | false | false | true | false | false | false |
| | instanceof | false | false | true | false | false | false |
| | (<columnName>) | error | error | error | error | error | error |
| "string" | as | null | null | string | null | null | null |
| | is | false | false | true | false | false | false |
| | instanceof | false | false | true | false | false | false |
| | (<columnName>) | error | error | error | error | error | error |
| Boolean | as | null | null | [class Boolean] | [class Boolean] | null | null |
| | is | false | false | true | true | false | false |
| | instanceof | false | false | true | true | false | false |
| | (<columnName>) | true | true | true | true | true | true |
| String | as | null | null | [class String] | [class String] | null | null |
| | is | false | false | true | true | false | false |
| | instanceof | false | false | true | true | false | false |
| | (<columnName>) | [class Sprite] | [class IBitmapDrawable] | [class Object] | [class Class] | [class uint] | [class i |
| Number | as | null | null | [class Number] | [class Number] | null | null |
| | is | false | false | true | true | false | false |
| | instanceof | false | false | true | true | false | false |
| | (<columnName>) | NaN | NaN | NaN | NaN | NaN | NaN |
| int | as | null | null | [class int] | [class int] | null | null |
| | is | false | false | true | true | false | false |
| | instanceof | false | false | true | true | false | false |
| | (<columnName>) | 0 | 0 | 0 | 0 | 0 | 0 |
| uint | as | null | null | [class uint] | [class uint] | null | null |
| | is | false | false | true | true | false | false |
| | instanceof | false | false | true | true | false | false |
| | (<columnName>) | 0 | 0 | 0 | 0 | 0 | 0 |
| Class | as | null | null | [class Class] | [class Class] | null | null |
| | is | false | false | true | true | false | false |
| | instanceof | false | false | true | true | false | false |
| | (<columnName>) | [class Sprite] | [class IBitmapDrawable] | [class Object] | [class Class] | [class uint] | [class i |
| Object | as | null | null | [class Object] | [class Object] | null | null |
| | is | false | false | true | true | false | false |
| | instanceof | false | false | true | true | false | false |
| | (<columnName>) | [class Sprite] | [class IBitmapDrawable] | [class Object] | [class Class] | [class uint] | [class i |
| IBitmapDrawable | as | null | null | [class IBitmapDrawable] | [class IBitmapDrawable] | null | null |
| | is | false | false | true | true | false | false |
| | instanceof | false | false | true | true | false | false |
| | (<columnName>) | error | error | error | error | error | error |
| Sprite | as | null | null | [class Sprite] | [class Sprite] | null | null |
| | is | false | false | true | true | false | false |
| | instanceof | false | false | true | true | false | false |
| | (<columnName>) | error | error | error | error | error | error |



## Tableaux typés

[1] .

- Vous recevez le type de compilation `TypeError` s si vous essayez d'insérer des valeurs non-T dans la collection.
- Les IDE fournissent des informations utiles sur les indications de type pour les objets à l'intérieur d'une instance de `Vector.<T>` .

Exemples de création d'un `Vector.<T>` :

```
var strings:Vector.<String> = new Vector.<String>(); // or
var numbers:Vector.<Number> = new <Number>[];
```

---

[1] Les vecteurs n'apportent en réalité que des améliorations notables des performances par rapport aux tableaux lors de l'utilisation de types primitifs ( `String` , `int` , `uint` , `Number` , etc.).

Lire Les types en ligne: https://riptutorial.com/fr/actionscript-3/topic/2803/les-types

# Chapitre 12: Manipulation et filtrage de bitmap

## Introduction

Dans cette rubrique, vous pouvez en apprendre un peu plus sur la manipulation des données **bitmap** et du traitement visuel, travailler avec les pixels et commencer avec les filtres d'effets.

## Examples

### Effet de seuil (monochrome)

---

## Champs obligatoires:

1. comprendre les données Bitmap et Bitmap

---

**quel est le seuil**

> Ce réglage prend tous les pixels d'une image et les… pousse soit vers le blanc pur, soit vers le noir pur

**ce que nous avons à faire**

> Voici une **démonstration en direct** de cet exemple avec quelques modifications supplémentaires, comme l'utilisation d'une interface utilisateur pour modifier le niveau de seuil en cours d'exécution.


**seuil dans le script d'action 3** de la documentation officielle as3

> Teste les valeurs de pixels dans une image par rapport à un seuil spécifié et définit les pixels qui passent le test à de nouvelles valeurs de couleur. À l'aide de la méthode threshold (), vous pouvez isoler et remplacer les plages de couleurs d'une image et effectuer d'autres opérations logiques sur les pixels de l'image.
>
> **La logique de test de la méthode threshold () est la suivante:**
>
> 1. Si ((pixelValue & mask) opération (seuil et masque)), définissez le pixel sur couleur;
> 2. Sinon, si copySource == true, définissez le pixel sur la valeur de pixel correspondante de sourceBitmap.

Je viens de commenter le code suivant avec exactement les noms comme description citée.

```
import flash.display.BitmapData;
import flash.display.Bitmap;
import flash.geom.Rectangle;
import flash.geom.Point;

var bmd:BitmapData = new wildcat(); // instantied a bitmapdata from library a wildcat
var bmp:Bitmap = new Bitmap(bmd); // our display object to previewing bitmapdata on stage
addChild(bmp);
monochrome(bmd); // invoking threshold function

/**
    @param bmd, input bitmapData that should be monochromed
*/
function monochrome(bmd:BitmapData):void {
    var bmd_copy:BitmapData = bmd.clone(); // holding a pure copy of bitmapdata for
comparation steps
    // this is our "threshold" in description above, source pixels will be compared with this
value
    var level:uint = 0xFFAAAAAA; // #AARRGGBB. in this case i used RGB(170,170,170) with an
alpha of 1. its not median but standard
    // A rectangle that defines the area of the source image to use as input.
    var rect:Rectangle = new Rectangle(0,0,bmd.width,bmd.height);
    // The point within the destination image (the current BitmapData instance) that
corresponds to the upper-left corner of the source rectangle.
    var dest:Point = new Point();
    // thresholding will be done in two section
    // the last argument is "mask", which exists in both sides of comparation
    // first, modifying pixels which passed comparation and setting them all with "color"
white (0xFFFFFFFF)
    bmd.bitmapData.threshold(bmd_copy, rect, dest, ">", level, 0xFFFFFFFF, 0xFFFFFFFF);
    // then, remaining pixels and make them all with "color" black (0xFF000000)
    bmd.bitmapData.threshold(bmd_copy, rect, dest, "<=", level, 0xFF000000, 0xFFFFFFFF);
    // Note: as we have no alpha channel in our default BitmapData (pixelValue), we left it to
its full value, a white mask (0xffffffff)
}
```

Lire Manipulation et filtrage de bitmap en ligne: https://riptutorial.com/fr/actionscript-3/topic/8055/manipulation-et-filtrage-de-bitmap

# Chapitre 13: Motif Singleton

## Remarques

Le modèle singleton a pour but de permettre à une seule instance d'une classe d'exister à un moment donné.

Empêcher l'instanciation directe via le constructeur est généralement évité en le rendant privé. Cependant, cela n'est pas possible dans As3 et, par conséquent, d'autres moyens de contrôler le nombre d'instances doivent être utilisés.

## Examples

### Singleton organiser via une instance privée

Dans cette approche, le single est accessible via la méthode statique:

```
Singleton.getInstance();
```

Pour appliquer une seule instance du singleton, une variable statique privée conserve l'instance, tandis que toute tentative supplémentaire d'instancier une instance est appliquée dans le constructeur.

```
package {

public class Singleton {

    /** Singleton instance */
    private static var _instance: Singleton = new Singleton();

    /** Return singleton instance. */
    public static function getInstance():Singleton {
        return _instance;
    }

    /** Constructor as singleton enforcer. */
    public function Singleton() {
        if (_instance)
            throw new Error("Singleton is a singleton and can only be accessed through
Singleton.getInstance()");
    }

}
}
```

Lire Motif Singleton en ligne: https://riptutorial.com/fr/actionscript-3/topic/1437/motif-singleton

# Chapitre 14: Optimiser les performances

## Examples

### Graphiques vectoriels

Les graphiques vectoriels sont représentés par une multitude de données qui doivent être calculées par le processeur (points vectoriels, arcs, couleurs, etc.). Tout ce qui n'est pas des formes simples avec des points et des lignes droites minimes consomme de grandes quantités de ressources du processeur.

Il existe un indicateur "Cache as Bitmap" qui peut être activé. Cet indicateur stocke le résultat du dessin de l'objet DisplayObject vectoriel pour des retraits beaucoup plus rapides. Le problème est que si des transformations sont appliquées à l'objet, l'ensemble doit être redessiné et remis en cache. Cela peut être plus lent que de ne pas l'activer du tout si des transformations image par image sont appliquées (rotation, mise à l'échelle, etc.).

Généralement, le rendu graphique à l'aide de bitmaps est beaucoup plus performant que l'utilisation de graphiques vectoriels. Les bibliothèques telles que flixel en profitent pour rendre les sprites sur un "canvas" sans réduire le framerate.

### Texte

Le rendu du texte consomme beaucoup de CPU. Les polices sont rendues de la même manière que les graphiques vectoriels et contiennent de nombreux points vectoriels pour chaque caractère. La modification du texte image par image *va* dégrader les performances. L'indicateur "Cache as bitmap" est extrêmement utile s'il est utilisé correctement, ce qui signifie que vous devez éviter:

- Modifier le texte fréquemment.
- Transformation du champ de texte (rotation, mise à l'échelle).

Des techniques simples comme l'enrobage de mises à jour de texte dans une instruction `if` feront une différence majeure:

```
if (currentScore !== oldScore) {
    field.text = currentScore;
}
```

Le texte peut être rendu en utilisant le moteur de rendu anti-aliasé intégré à Flash ou en utilisant les "polices de périphérique". L'utilisation de "polices de périphérique" rend le rendu du texte beaucoup plus rapide, bien qu'il rende le texte irrégulier (aliasé). De plus, les polices de périphérique requièrent que la police soit pré-installée par votre utilisateur final, ou le texte risque de «disparaître» sur le PC de l'utilisateur, même s'il semble correct sur le vôtre.

```
field.embedFonts = false; // uses "device fonts"
```

## Vecteur et pour chaque vs tableaux et pour

En utilisant le type `Vector.<T>` et le `for each` boucle est plus performant qu'un tableau conventionnel et `for` boucle:

Bien:

```
var list:Vector.<Sprite> = new <Sprite>[];

for each(var sprite:Sprite in list) {
    sprite.x += 1;
}
```

Mal:

```
var list:Array = [];

for (var i:int = 0; i < list.length; i++) {
    var sprite:Sprite = list[i];

    sprite.x += 1;
}
```

## Suppression rapide d'éléments de tableau

Si vous n'avez pas besoin d'un tableau dans un ordre particulier, une petite astuce avec `pop()` vous permettra de gagner énormément en performance par rapport à `splice()` .

Lorsque vous `splice()` un tableau, l'index des éléments suivants de ce tableau doit être réduit de 1. Ce processus peut consommer une grande partie du temps si le tableau est volumineux et que l'objet que vous supprimez est plus proche du début de ce tableau. .

Si vous ne vous souciez pas de l'ordre des éléments du tableau, vous pouvez remplacer l'élément que vous souhaitez supprimer par un élément que vous `pop()` à la fin du tableau. De cette manière, les index de tous les autres éléments du tableau restent les mêmes et le processus ne se dégrade pas au fur et à mesure que la longueur de votre tableau augmente.

Exemple:

```
function slowRemove(list:Array, item:*):void {
    var index:int = list.indexOf(item);

    if (index >= 0) list.splice(index, 1);
}

function fastRemove(list:Array, item:*):void {
    var index:int = list.indexOf(item);

    if (index >= 0) {
        if (index === list.length - 1) list.pop();

        else {
            // Replace item to delete with last item.
```

```
            list[index] = list.pop();
        }
    }
}
```

## Vecteurs au lieu de tableaux

Flash Player 10 a introduit le type de liste générique Vector. <*> Plus rapide que le tableau. Cependant, ce n'est pas tout à fait vrai. Seuls les types de vecteur suivants sont plus rapides que les homologues Array, en raison de leur implémentation dans Flash Player.

- `Vector.<int>` - Vecteur d'entiers 32 bits
- `Vector.<uint>` - Vecteur d'entiers non signés 32 bits
- `Vector.<Double>` - Vecteur de flottants 64 bits

Dans tous les autres cas, l'utilisation d'un tableau sera plus performante que l'utilisation des vecteurs, pour toutes les opérations (création, manipulation, etc.). Cependant, si vous souhaitez "taper fortement" votre code, vous pouvez utiliser les vecteurs malgré le ralentissement. FlashDevelop a une syntaxe qui permet aux `/*ObjectType*/Array` déroulantes d'achèvement du code de fonctionner même pour les tableaux, en utilisant `/*ObjectType*/Array` .

```
var wheels:Vector.<Wheel> // strongly typed, but slow

var wheels:/*Wheel*/Array // weakly typed, but faster
```

## Réutilisation et regroupement des graphiques

La création et la configuration d'objets `Sprite` et `TextField` au moment de l'exécution peuvent être coûteux si vous en créez des centaines de milliers sur une seule image. Par conséquent, une astuce courante consiste à "regrouper" ces objets pour les réutiliser ultérieurement. Rappelez-vous que nous n'essayons pas seulement d'optimiser le temps de création ( `new Sprite()` ) mais aussi la configuration (paramétrage des propriétés par défaut).

Disons que nous construisions un composant de liste à l'aide de centaines d'objets TextField. Lorsque vous devez créer un nouvel objet, vérifiez si un objet existant peut être réutilisé.

```
var pool:Array = [];

if (pool.length > 0){

    // reuse an existing TextField
    var label = pool.pop();

}else{
    // create a new TextField
    label = new TextField();

    // initialize your TextField over here
    label.setDefaultTextFormat(...);
    label.multiline = false;
    label.selectable = false;
}
```

```
// add the TextField into the holder so it appears on-screen
// you will need to layout it and set its "text" and other stuff seperately
holder.addChild(label);
```

Plus tard, lorsque vous détruisez votre composant (ou le supprimez de l'écran), n'oubliez pas d'ajouter des étiquettes inutilisées dans le pool.

```
foreach (var label in allLabels){
    label.parent.removeChild(label); // remove from parent Sprite
    pool.push(label); // add to pool
}
```

Dans la plupart des cas, il est préférable de créer un pool par utilisation au lieu d'un pool global. Inconvénients de la création d'un pool global: vous devez réinitialiser l'objet à chaque fois pour le récupérer à partir du pool, pour annuler les paramètres définis par d'autres fonctions. Ceci est tout aussi coûteux et nie en grande partie le gain de performance lié à l'utilisation de la mise en commun en premier lieu.

Lire Optimiser les performances en ligne: https://riptutorial.com/fr/actionscript-3/topic/2215/optimiser-les-performances

# Chapitre 15: Programmation orientée objet

## Examples

### Constructeur "surchargé" par méthode statique

La surcharge du constructeur n'est pas disponible dans As3.

Afin de fournir une manière différente de récupérer une instance d'une classe, une méthode `public static` peut être fournie pour servir de "constructeur" alternatif.

`flash.geom.Point` , qui représente un objet ponctuel 2D, en est un exemple. Les coordonnées pour définir le point peuvent être

- **cartésien** dans le constructeur régulier

  ```
  public function Point(x:Number = 0, y:Number = 0)
  ```

  exemple d'utilisation:

  ```
  var point:Point = new Point(2, -.5);
  ```

- **polaire** dans une méthode statique

  ```
  public static function polar(len:Number, angle:Number):Point
  ```

  exemple d'utilisation:

  ```
  var point:Point = Point.polar(12, .7 * Math.PI);
  ```

  Comme il ne s'agit pas d'un constructeur réel, il n'y a pas de `new` mot clé.

### définir et obtenir des fonctions

Pour garantir l'encapsulation, les variables membres d'une classe doivent être `private` et ne doivent être accessibles au `public` par le biais de méthodes d'accès `get` / `set` publiques. Il est courant de préfixer des champs privés avec _

```
public class Person
{
    private var _name:String = "";

    public function get name():String{
        return _name;
        //or return some other value depending on the inner logic of the class
    }
```

```
    public function set name(value:String):void{
        //here you may check if the new value is valid
        //or maybe dispatch some update events or whatever else
        _name = value;
    }
```

Parfois, vous n'avez même pas besoin de créer un champ `private` pour une paire `get` / `set` . Par exemple, dans un contrôle tel qu'un groupe de radio personnalisé, vous devez savoir quel bouton radio est sélectionné. Cependant, en dehors de la classe, vous avez juste besoin d'un moyen d' `get` / de `set` uniquement la valeur sélectionnée:

```
public function get selectedValue():String {
    //just the data from the element
    return _selected ? _selected.data : null;
}
public function set selectedValue(value:String):void {
    //find the element with that data
    for (var i:int = 0; i < _elems.length; i++) {
        if (_elems[i].data == value) {
            _selected = _elems[i];//set it
            processRadio();//redraw
            return;
        }
    }
}
```

## Paquets

Les packages sont des lots de classes. Chaque classe doit être déclarée dans un package à l'aide de l'instruction `package` . L'instruction de `package` est suivie du nom de votre package ou suivie de rien dans le cas de l'ajout de classes au package de niveau supérieur. Les sous-packages sont créés à l'aide de la délimitation par points ( `.` ). L'instruction package est suivie d'un bloc qui contiendra *une définition de `class` unique* . Exemples:

```
package {
    // The top level package.
}

package world {
    // A package named world.
}

package world.monsters {
    // A package named monsters within a package named world.
}
```

Les packages doivent correspondre à la structure de fichier des classes par rapport à la racine source. En supposant que vous avez un dossier racine source nommé `src` , ce qui précède pourrait être correctement représenté dans le système de fichiers comme:

```
src
    TopLevelClass.as
```

```
    world
        ClassInWorldPackage.as
        AnotherClassInWorldPackage.as

        monsters
            Zombie.as
```

## Méthode à écraser

Lorsque vous `extend` une classe, vous pouvez `override` méthodes `override` par la classe héritée à l'aide du mot clé `override` :

```
public class Example {
    public function test():void {
        trace('It works!');
    }
}

public class AnotherExample extends Example {
    public override function test():void {
        trace('It still works!');
    }
}
```

Exemple:

```
var example:Example = new Example();
var another:AnotherExample = new AnotherExample();

example.test(); // Output: It works!
another.test(); // Output: It still works!
```

Vous pouvez utiliser le mot `super` clé `super` pour référencer la méthode d'origine de la classe héritée. Par exemple, nous pourrions changer le corps de `AnotherExample.test()` pour:

```
public override function test():void {
    super.test();
    trace('Extra content.');
}
```

Résultant en:

```
another.test(); // Output: It works!
                //         Extra content.
```

La substitution des constructeurs de classes est un peu différente. Le mot-clé `override` est omis et l'accès au constructeur hérité se fait simplement avec `super()` :

```
public class AnotherClass extends Example {
    public function AnotherClass() {
        super(); // Call the constructor in the inherited class.
    }
```

```
}
```

Vous pouvez également remplacer les méthodes `get` et `set` .

## getter et setter

Les getters et les setters sont des méthodes qui se comportent comme des propriétés. cela signifie qu'ils ont une structure de fonction mais lorsqu'ils sont utilisés, ils sont utilisés comme des propriétés:

**Structure des fonctions getter** :

ils doivent avoir le mot-clé `get` après la `function` et avant le nom de la fonction, sans argument, un type de retour spécifié et doit renvoyer une valeur:

```
public function get myValue():Type{
    //anything here
    return _desiredValue;
}
```

**Syntaxe** :

pour obtenir la valeur d'un getter, la syntaxe est identique à celle d'une propriété (no parens `()` est utilisé).

```
trace(myValue);
```

**Structure des fonctions de réglage** :

Ils doivent avoir `set` mot-clé après la `function` et avant le nom de la fonction, avec un argument et aucun retour de valeur.

```
public function set myValue(value:Type):void{
    //anything here
    _desiredProperty=value;
}
```

**Syntaxe** :

Pour définir la valeur d'un dispositif de réglage, la syntaxe est identique à la définition d'une valeur pour une propriété (en utilisant le signe égal `=` alors valeur).

```
myValue=desiredValue;
```

**définir un getter et un setter pour une valeur** :

> Remarque: si vous ne créez que getter ou uniquement setter avec un nom, cette propriété serait en lecture seule ou définie uniquement.

pour rendre une propriété à la fois lisible et configurable, il faut créer un getter et un setter avec:

1. même nom.
2. le même type (type de valeur de retour pour le getter et type de valeur d'entrée (argument) pour le setter,

> Remarque: les getters et les setters ne doivent pas avoir un nom identique à celui d'autres propriétés ou méthodes.

**Utilisation des getters et des setters:**

Utiliser des getters et des setters plutôt que des propriétés normales a de nombreux avantages:

1. **créer des propriétés en lecture seule ou uniquement en set:**
par exemple nombre d'enfants dans un objet d'affichage. ça ne peut pas être réglé.

2. **accès aux propriétés privées:**
un exemple:

```
private var _private:Type=new Type();
//note that function name "private" is not same as variable name "_private"
public function get private():Type{
    return _private;
}
```

3. **quand un changement est nécessaire après avoir défini une valeur:**
dans cet exemple, la modification de cette propriété doit être notifiée:

```
public static function set val:(input:Type):void{
    _desiredProperty=input;
    notifyValueChanged();
}
```

et bien d'autres usages

Lire Programmation orientée objet en ligne: https://riptutorial.com/fr/actionscript-3/topic/2042/programmation-orientee-objet

# Chapitre 16: Travailler avec des événements

## Remarques

Les événements sont des éléments de données qu'un programme peut créer, échanger et réagir. Le flux d'événements asynchrone est réparti sur la liste d'affichage par moteur Flash en réaction à des événements externes, tels que des mouvements de souris ou une autre image affichée. Tout autre flux d'événements et tout traitement d'événement est synchrone, donc si un morceau de code a généré un événement, toutes les réactions sont traitées avant l'exécution de la ligne de code suivante, même s'il y a plusieurs auditeurs d'un événement, tous aurait couru avant le prochain événement pourrait être traité.

Il existe plusieurs événements majeurs associés à la programmation Flash. `Event.ENTER_FRAME` est généré avant que Flash dessine une autre image, il signale à la liste d'affichage entière de se préparer à être dessinée et peut être utilisé comme une minuterie synchrone. `MouseEvent.CLICK` et ses frères et sœurs peuvent être utilisés pour recevoir des entrées de souris de l'utilisateur, et `TouchEvent.TOUCH_TAP` est un analogue pour les écrans tactiles. `KeyboardEvent.KEY_DOWN` et `KEY_UP` permettent de recevoir les entrées utilisateur depuis le clavier. Cependant, leur utilisation dans le département mobile est presque impossible en raison de l'absence de clavier physique sur les appareils. Enfin, `Event.ADDED_TO_STAGE` est distribué une fois qu'un objet d'affichage a accès à la scène et est inclus dans la liste d'affichage globale qui reçoit l'intégralité des événements pouvant monter ou descendre dans la liste d'affichage.

La plupart des événements de Flash sont spécifiques aux composants. Si vous concevez votre propre composant qui utilisera des événements Flash, utilisez une classe descendante `flash.events.Event` et ses propriétés `String` statiques pour créer le jeu d'événements de votre composant.

## Examples

### Événements personnalisés avec données d'événement

```
package
{
    import flash.events.Event;

    public class CustomEvent extends Event
    {
        public static const START:String = "START";
        public static const STOP:String  = "STOP";

        public var data:*;

        public function CustomEvent(type:String, data:*,
                                    bubbles:Boolean=false, cancelable:Boolean=false)
        {
            super(type, bubbles, cancelable);
```

```
            if (data)
                this.data = data;
        }
    }
}
```

Pour envoyer un événement personnalisé:

```
var dataObject:Object = {name: "Example Data"};

dispatchEvent(new CustomEvent(CustomEvent.START, dataObject))
```

Pour écouter les événements personnalisés:

```
addEventListener(CustomEvent.STOP, stopHandler);

function stopHandler(event:CustomEvent):void
{
    var dataObject:* = event.data;
}
```

## Gestion des événements hors liste d'affichage

```
package {
import flash.events.EventDispatcher;

public class AbstractDispatcher extends EventDispatcher {

    public function AbstractDispatcher(target:IEventDispatcher = null) {
        super(target);
    }

}
}
```

Pour envoyer un événement sur une instance:

```
var dispatcher:AbstractDispatcher = new AbstractDispatcher();
dispatcher.dispatchEvent(new Event(Event.CHANGE));
```

Pour écouter les événements sur une instance:

```
var dispatcher:AbstractDispatcher = new AbstractDispatcher();
dispatcher.addEventListener(Event.CHANGE, changeHandler);

function changeHandler(event:Event):void
{
}
```

## Gestion d'événement de base

Flash distribue des **Events** pour la plupart de ses objets. L'un des événements les plus

élémentaires est `ENTER_FRAME` , qui est envoyé (au débit du fichier SWF) sur chaque objet de la liste d'affichage.

```
import flash.display.Sprite;
import flash.events.Event;

var s:Sprite = new Sprite();
s.addEventListener(Event.ENTER_FRAME, onEnterFrame);

function onEnterFrame(e:Event)
{
    trace("I am called on every frame !");
}
```

Cette fonction sera appelée de manière asynchrone sur chaque image. Cela signifie que la fonction que vous attribuez en tant que gestionnaire d'événements `onEnterFrame` est traitée avant tout autre code ActionScript associé aux images concernées.

## Ajoutez vos propres événements

Vous pouvez créer vos propres événements et les distribuer en étendant la classe `Event` .

```
import flash.events.Event;

class MyEvent extends Event
{
    var data: String;

    static public var MY_EVENT_TYPE = "my_event_my_event_code";

    public function MyEvent(type: String, data: String)
    {
        this.data = data;
    }

    override public function clone():Event
    {
        return new MyEvent(type, data);
    }
}
```

Vous pouvez ensuite l'envoyer et l'écouter en utilisant un `EventDispatcher` . Notez que la plupart des objets flash sont des répartiteurs d'événements.

```
import flash.events.EventDispatcher;

var d = new EventDispatcher();
d.addEventListener(MyEvent.MY_EVENT_TYPE, onType);

function onType(e: MyEvent)
{
    trace("I have a string: "+e.data);
}

d.dispatchEvent(new MyEvent(MyEvent.MY_EVENT_TYPE, "Hello events!"));
```

Notez que la méthode de `clone` est requise si vous souhaitez redistribuer votre événement.

### Structure d'événement de souris simple

Grâce à l'utilisation de `event types` vous pouvez facilement réduire le gonflement du code qui se produit souvent lors de la définition des événements pour de nombreux objets sur la scène en filtrant les événements dans une fonction plutôt que de définir de nombreuses fonctions de gestion des événements.

Imaginez que nous avons 10 objets sur scène nommé `object1` , `object2` ... `object10`

Vous pouvez faire ce qui suit:

```
var i: int = 1;
while(getChildByName("object"+i) != null){
    var obj = getChildByName("object"+i)
    obj.addEventListener(MouseEvent.CLICK, ObjectMouseEventHandler);
    obj.addEventListener(MouseEvent.MOUSE_OVER, ObjectMouseEventHandler);
    obj.addEventListener(MouseEvent.MOUSE_OUT, ObjectMouseEventHandler);
    obj.alpha = 0.75;
    i++;
}

function ObjectMouseEventHandler(evt:Event)
{
    if(evt.type == "click")
    {
        trace(evt.currentTarget + " has been clicked");
    }
    else
    {
        evt.currentTarget.alpha = evt.type == "mouseOver" ? 1 : 0.75;
    }
}
```

**Les avantages de cette méthode incluent:**

1. Pas besoin de spécifier la quantité d'objets à appliquer aux événements.
2. Ne pas avoir besoin de savoir précisément quel objet a été interagi avec encore la fonctionnalité.
3. Appliquer facilement des événements en vrac.

Lire Travailler avec des événements en ligne: https://riptutorial.com/fr/actionscript-3/topic/1925/travailler-avec-des-evenements

# Chapitre 17: Travailler avec des minuteries

## Examples

### Exemple de compte à rebours

```
package {
import flash.events.TimerEvent;
import flash.utils.Timer;

public class CountdownTimer extends Timer {

    public var time:Number = 0;

    public function CountdownTimer(time:Number = Number.NEGATIVE_INFINITY, delay:Number =
1000) {
        super(delay, repeatCount);

        if (!isNaN(time))
            this.time = time;

        repeatCount = Math.ceil(time / delay);

        addEventListener(TimerEvent.TIMER, timerHandler);
        addEventListener(TimerEvent.TIMER_COMPLETE, timerCompleteHandler);
    }

    override public function start():void {
        super.start();
    }

    protected function timerHandler(event:TimerEvent):void {
        time -= delay;
    }

    protected function timerCompleteHandler(event:TimerEvent):void {
    }

    override public function stop():void {
        super.stop();
    }

    public function dispose():void {
        removeEventListener(TimerEvent.TIMER, timerHandler);
        removeEventListener(TimerEvent.TIMER_COMPLETE, timerCompleteHandler);
    }

}
}
```

Ce `CountdownTimer` étend le `Timer` et est utilisé exactement de la même façon, sauf que le temps est compté.

Exemple d'utilisation:

```
var timer:CountdownTimer = new CountdownTimer(5000);
timer.addEventListener(TimerEvent.TIMER, timerHandler);
timer.addEventListener(TimerEvent.TIMER_COMPLETE, completeHandler);
timer.start();

function timerHandler(event:TimerEvent):void {
    trace("Time remaining: " + event.target.time);
}

function completeHandler(event:TimerEvent):void {
    trace("Timer complete");
}
```

L'exemple ci-dessus produirait:

```
[trace] Time remaining: 4000
[trace] Time remaining: 3000
[trace] Time remaining: 2000
[trace] Time remaining: 1000
[trace] Time remaining: 0
[trace] Timer complete
```

## Intervalles et délais

```
import flash.utils.*;
var intervalId:uint=setInterval(schroedingerCat,1000);
// execute a function once per second and gather interval ID
trace("Cat's been closed in the box.");
function schroedingerCat():void {
    if (Math.random()<0.04) {
        clearInterval(intervalId); // stop repeating by ID
        trace("Cat's dead.");
        return;
    }
    trace("Cat's still alive...");
}

var bombId:uint;
function plantBomb(seconds:Number):uint {
    trace("The bomb has been planted, and will blow in "+seconds.toFixed(3)+" seconds!");
    var id:uint=setTimeout(boom,seconds*1000); // parameter is in milliseconds
    return id;
}
function defuseBomb(id:uint):void {
    clearTimeout(id);
    trace("Bomb with id",id,"defused!");
}
function boom():void {
    trace("BOOM!");
}
```

`setInterval()` est utilisé pour effectuer des tâches répétées de manière asynchrone en tant qu'intervalles spécifiés. L'objet `Timer` interne est utilisé, la valeur renvoyée de type `uint` est son ID interne, grâce à laquelle vous pouvez accéder et arrêter la répétition en appelant `clearInterval()` . `setTimeout()` et `clearTimeout()` fonctionnent de la même manière, mais l'appel à la fonction fournie n'est effectué qu'une seule fois. Vous pouvez fournir des arguments supplémentaires aux deux

fonctions définies, celles-ci seront transmises à la fonction dans l'ordre. Le nombre d'arguments et leur type ne sont pas vérifiés au moment de la compilation. Par conséquent, si vous fournissez une étrange combinaison d'arguments ou une fonction qui les requiert et n'en reçoit aucune, une erreur "Erreur # 1063: incompatibilité des arguments" est générée.

Vous pouvez effectuer les deux activités de `setInterval` et de `setTimeout` avec des objets `Timer` standard, en utilisant 0 ou 1 pour la propriété `repeatCount` , 0 pour les répétitions indéfinies et 1 pour les répétitions indéfinies.

## Exemple de minuterie aléatoire

```
package {
    import flash.events.TimerEvent;
    import flash.events.TimerEvent;
    import flash.utils.Timer;

    public class RandomTimer extends Timer {

        public var minimumDelay:Number;
        public var maximumDelay:Number;
        private var _count:uint = 0;
        private var _repeatCount:int = 0;

        public function RandomTimer(min:Number, max:Number, repeatCount:int = 0) {
            super(delay, repeatCount);

            minimumDelay = min;
            maximumDelay = max;
            _repeatCount = repeatCount;
        }

        override public function start():void {
            delay = nextDelay();
            addEventListener(TimerEvent.TIMER, timerHandler);
            super.start();
        }

        private function nextDelay():Number {
            return (minimumDelay + (Math.random() * (maximumDelay - minimumDelay)));
        }

        override public function stop():void {
            removeEventListener(TimerEvent.TIMER, timerHandler);
            super.stop();
        }

        protected function timerHandler(event:TimerEvent):void {
            _count++;
            if ((_repeatCount > 0) && (_count >= _repeatCount)) {
                stop();
                dispatchEvent(new TimerEvent(TimerEvent.TIMER_COMPLETE));
            }
            delay = nextDelay();
        }

        override public function reset():void {
            _count = 0;
            super.reset();
```

```
        }
    }
}
```

Ce `RandomTimer` étend `Timer` et est utilisé exactement de la même manière, sauf qu'il distribue à des intervalles aléatoires.

Exemple d'utilisation, répartition aléatoire entre 1 et 5 secondes:

```
var t:int = getTimer();

var timer:RandomTimer = new RandomTimer(1000, 5000);
timer.addEventListener(TimerEvent.TIMER, timerHandler);
timer.start();

function timerHandler(event:TimerEvent):void {
    trace("Time since last dispatch: " + (getTimer() - t));
    t = getTimer();
}
```

L'exemple ci-dessus produirait:

```
[trace] Time since last dispatch: 1374
[trace] Time since last dispatch: 2459
[trace] Time since last dispatch: 3582
[trace] Time since last dispatch: 1335
[trace] Time since last dispatch: 4249
```

Lire Travailler avec des minuteries en ligne: https://riptutorial.com/fr/actionscript-3/topic/2798/travailler-avec-des-minuteries

# Chapitre 18: Travailler avec des objets d'affichage

## Syntaxe

1. `addChild(child)` - ajoute un nouvel élément à l'arborescence enfant de cet objet en tant qu'élément le plus élevé.
2. `addChildAt(child, index)` - ajoute un nouvel élément à l'arborescence enfant de cet objet à une position spécifiée. L'élément le plus bas a un index de 0.
3. `getChildAt(index)` - Retourne un enfant avec un index donné.
4. `getChildIndex(child)` renvoie l'index d'un enfant *direct* de cet objet. Sinon, une exception est levée.
5. `removeChild(child)` - supprime l'enfant direct spécifié de l'arborescence enfant de cet objet. Lance une exception si le parent de l'enfant fourni n'est pas égal à `this` .
6. `removeChildAt(index)` - supprime un enfant sélectionné par index au lieu de référence. Lève une exception si l'arborescence enfant n'est pas aussi large.
7. `removeChildren(beginIndex:int = 0, endIndex:int = 0x7fffffff))` - ajouté dans Flash Player 11, supprime un sous-ensemble d'enfants par plage d'index ou tous les enfants s'ils sont appelés sans paramètre.
8. `setChildIndex(child,index)` - modifie l'index de l'enfant à la nouvelle valeur, en déplaçant tous les enfants entre les deux pour occuper le poste libéré.
9. `swapChildren(child1,child2)` - `swapChildren(child1,child2)` les positions des deux enfants dans la liste d'affichage, sans affecter les positions des autres enfants.
10. `swapChildrenAt(index1,index2)` - échange les enfants situés par leurs index.

## Remarques

La liste d'affichage est en réalité un arbre et est visualisée avec le premier algorithme de profondeur. Tout objet répertorié précédemment sera affiché plus tôt et pourrait être masqué par les objets répertoriés ultérieurement. Toutes les techniques pouvant être utilisées contre une arborescence peuvent être appliquées au travail avec la liste d'affichage.

## Examples

### Introduction à la liste d'affichage

Dans AS3, les ressources d'affichage ne sont visibles que lorsqu'elles sont ajoutées à la liste d'affichage.

Le moteur d'exécution AIR / Flash a une structure d'affichage hiérarchique (relation parent-enfant où les enfants peuvent avoir leurs propres enfants), la `stage` étant le parent de niveau supérieur.

Pour ajouter quelque chose à la liste d'affichage, utilisez `addChild` ou `addChildAt` . Voici un exemple

de base pour dessiner un cercle et l'ajouter à la liste d'affichage:

```
var myCircle:Shape = new Shape();

myCircle.graphics.beginFill(0xFF0000); //red
myCircle.graphics.drawCircle(25, 25, 50);
myCircle.graphics.endFill();

this.addChild(myCircle); //add the circle as a child of `this`
```

Pour voir l'objet dans l'exemple ci-dessus, `this` (le contexte du code) doit également figurer sur la liste d'affichage, ainsi que tous les parents éventuels. Dans AS3, la `stage` est la plus importante des parents.

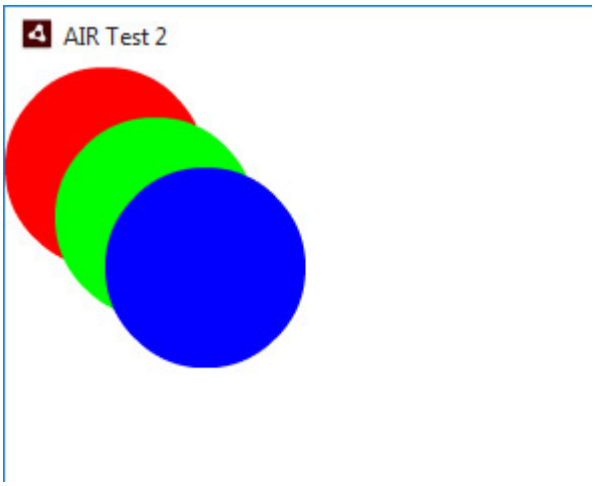
> Les objets d'affichage ne peuvent avoir qu'un seul parent. Donc, si un enfant a déjà un parent et que vous l'ajoutez à un autre objet, il sera supprimé de son parent précédent.

## Z-Order / Layering

Disons que vous avez répliqué le code de l'exemple précédent, vous avez donc 3 cercles:

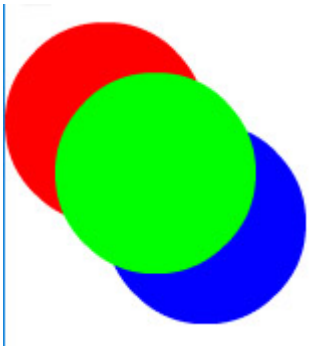
```
var redCircle:Shape = new Shape();
redCircle.graphics.beginFill(0xFF0000); //red
redCircle.graphics.drawCircle(50, 50, 50); //graphics.endFill is not required

var greenCircle:Shape = new Shape();
greenCircle.graphics.beginFill(0x00FF00); //green
greenCircle.graphics.drawCircle(75, 75, 50);

var blueCircle:Shape = new Shape();
blueCircle.graphics.beginFill(0x0000FF); //blue
blueCircle.graphics.drawCircle(100, 100, 50);

this.addChild(redCircle);
this.addChild(greenCircle);
this.addChild(blueCircle);
```

Comme la `addChild` ajoute l'enfant au-dessus de tout le même parent, vous obtiendrez ce résultat avec les éléments superposés dans le même ordre que vous utilisez addChild:



Si vous voulez un enfant en couches différent de ses frères, vous pouvez utiliser `addChildAt` . Avec `addChildAt` , vous passez un autre paramètre qui indique l'index (ordre-z) auquel l'enfant doit être. `0` étant la position / couche la plus basse.

```
this.addChild(redCircle);
this.addChild(greenCircle);
this.addChildAt(blueCircle,0); //This will add the blue circle at the bottom
```



Maintenant, le cercle bleu est sous ses frères et sœurs. Si, par la suite, vous souhaitez modifier l'index d'un enfant, vous pouvez utiliser la méthode `setChildIndex` (sur le parent de l'enfant).

```
this.setChildIndex(redCircle, this.numChildren - 1); //since z-index is 0 based, the top most
position is amount of children less 1.
```

Cela va réorganiser le cercle rouge pour qu'il soit au-dessus de tout. Le code ci-dessus produit exactement le même résultat que `this.addChild(redCircle)` .

## Suppression des objets d'affichage

Pour supprimer des objets, vous avez les méthodes inverse `removeChild` et `removeChildAt` ainsi que la méthode `removeChildren` .

```
removeChild(redCircle); //this will take redCircle off the display list

removeChildAt(0); //this will take the bottom most object off the display list

removeChildren(); //this will clear all children from the display list

removeChildren(1); //this would remove all children except the bottom most

removeChildren(1,3); //this would remove the children at indexes 1, 2 & 3
```

## Événements

Lorsqu'un enfant est ajouté à la liste d'affichage, certains événements sont déclenchés sur cet enfant.

- `Event.ADDED`
- `Event.ADDED_TO_STAGE`

Inversement, il y a aussi les événements de suppression:

- `Event.REMOVED`
- `Event.REMOVED_FROM_STAGE`

# Adobe Animate / Flash Professional

En ce qui concerne les calendriers FlashProfessional / Adobe Animate, l'ajout de quelque chose à la timeline gère automatiquement les nuances de la liste d'affichage. Ils ont ajouté et supprimé de la liste d'affichage automatiquement par la chronologie.

Cependant, il est bon de garder à l'esprit que:

> Si vous manipulez par code la parenté d'un objet d'affichage créé par le scénario (en utilisant addChild / setChildIndex), cet enfant ne sera plus supprimé automatiquement par le scénario et devra être supprimé via le code.

## Superposition

Il peut y avoir des situations où vous décidez qu'un ensemble d'objets d'affichage doit toujours être au-dessus d'un autre ensemble d'objets, par exemple des flèches au-dessus des têtes, des explosions sur quelque chose qui vient d'exploser, etc. et créez un ensemble de `Sprite` s, organisez-les de bas en haut, puis ajoutez simplement tous les objets de «dessus» à un calque au-dessus de celui utilisé pour les objets de l'ensemble «inférieur».

```
var monsters:Vector.<Monster>;
var bullets:Vector.<Bullet>; // desired: bullets strictly above monsters
var monsterLayer:Sprite=new Sprite();
var bulletLayer:Sprite=new Sprite();
addChild(monsterLayer);
addChild(bulletLayer);
```

Ensuite, chaque fois que vous ajoutez un `Monster` à la liste d'affichage, ajoutez-le à `monsterLayer` , et chaque fois que vous ajoutez une `Bullet` , ajoutez-la à `bulletLayer` pour obtenir l'effet souhaité.

## Supprimer tous les objets de la liste d'affichage

Si vous ciblez Flash Player 11+, la méthode removeChildren intégrée est la meilleure façon de supprimer tous les enfants:

```
removeChildren(); //a start and end index can be passed
```

Pour les applications héritées, la même chose peut être accomplie avec une boucle:

```
while (numChildren > 0) {
    removeChildAt(0);
}
```

## Passer des images au changement de contenu manuel

Au début, un développeur Flash utilise des cadres, tels qu'ils sont disponibles en natif dans Flash Player, pour héberger différents écrans de leur application (le plus souvent, c'est un jeu). En fin de compte, ils pourraient tomber sur un problème que quelque chose ne va pas exactement parce qu'ils ont utilisé des cadres, et ignoré les difficultés qui en découlent, et chercher des moyens de conserver leur structure de trame mais aussi d'éliminer les complications. La solution consiste à utiliser des classes descendantes `Sprite` ou des images exportées en tant que `MovieClip` avec une seule image (à celles conçues dans Adobe Flash CS), et à changer manuellement le contenu avec `addChild()` et `removeChild()` .

La classe du gestionnaire doit avoir toutes ses classes de cadres enfants prêtes et, chaque fois qu'une transition est appelée, une fonction similaire à celle-ci peut être utilisée:

```
var frames:Vector.<DisplayObject>; // this holds instances to ALL children
var currentFrame_alt:int; // current frame. Can't use the property
function changeFrame(frame:int):void {
    removeChild(frames[currentFrame_alt]);
    addChild(frames[frame]);
    currentFrame_alt=frame;
}
```

Tous les enfants peuvent à la fois envoyer et écouter des événements avec `Event.ADDED_TO_STAGE` utilisé comme point d'entrée pour tout ce qui se produit après `gotoAndStop()` qui cible cette image, et toutes les transitions sortantes peuvent être codées comme des chaînes écoutées dans `Main` classe `Main` . qui effectue ensuite la transition.

```
frames[0].addEventListener("startGame",startGame); // assuming frame 0 is a "Play" button
function startGame(e:Event):void {
    changeFrame(1); // switch to frame 1 - will display frames[1]
}
```

Bien sûr, le jeu de chaînes doit être prédéfini. Par exemple, l'intro screen peut avoir deux boutons pour démarrer le jeu, par exemple "Start game" et "Start muted", et les boutons doivent envoyer différents événements qui seront ensuite traités différemment dans la classe de gestionnaire.

Ce modèle peut aller aussi loin que nécessaire. Si une image du projet contient un MovieClip avec plusieurs images, cette méthode peut également être découplée en images-objets.

Lire Travailler avec des objets d'affichage en ligne: https://riptutorial.com/fr/actionscript-3/topic/1628/travailler-avec-des-objets-d-affichage

# Chapitre 19: Travailler avec des valeurs numériques

## Examples

### Si un nombre est une valeur paire

```
function isEven(n:Number):Boolean {
    return ((n & 1) == 0);
}
```

Exemples:

```
isEven(1); // false
isEven(2); // true

isEven(1.1); // false
isEven(1.2); // false
isEven(2.1); // true
isEven(2.2); // true
```

### Si un nombre est une valeur impaire

```
function isOdd(n:Number):Boolean {
    return ((n & 1) == 1);
}
```

Exemples:

```
isOdd(1); // true
isOdd(2); // false

isOdd(1.1); // true
isOdd(1.2); // true
isOdd(2.1); // false
isOdd(2.2); // false
```

### Arrondi au X le plus proche

Pour arrondir une valeur au multiple le plus proche de x:

```
function roundTo(value:Number, to:Number):Number {
    return Math.round(value / to) * to;
}
```

Exemple:

```
roundTo(8, 5); // 10
roundTo(17, 3); // 18
```

## Erreurs d'arrondi des nombres à virgule flottante

```
/**
 * @param n Number to be rounded.
 * @param precision Decimal places.
 * @return Rounded Number
 */
function roundDecimal(n:Number, precision:Number):Number {
    var factor:int = Math.pow(10, precision);
    return (Math.round(n * factor) / factor);
}
```

Exemples:

```
trace(0.9 - 1); // -0.09999999999999998

trace(roundDecimal(0.9 - 1, 1));     // -0.1
trace(roundDecimal(0.9 - 1, 2));     // -0.1

trace(roundDecimal(0.9 - 1.123, 1)); // -0.2
trace(roundDecimal(0.9 - 1.123, 2)); // -0.22
trace(roundDecimal(0.9 - 1.123, 3)); // -0.223
```

## Affichage des nombres avec précision requise

```
var a:Number=0.123456789;
trace(a);                    // 0.123456789
trace(a.toPrecision(4));     // 0.1235
trace(a.toFixed(4));         // 0.1235
trace(a.toExponential(4));   // 1.2345e-1
trace(a.toString(16));       // 0 - works for integer part only
var b:Number=12345678.9876543; // a bigger number to display rounding
trace(b);                    // 12345678.9876543
trace(b.toPrecision(4));     // 1.235e+7
trace(b.toFixed(4));         // 12345678.9877
trace(b.toExponential(4));   // 1.2345e+7
trace(b.toString(16));       // bc614e
b=1.0e+16;
trace(b.toString(36));       // 2qgpckvng1s
```

Lire Travailler avec des valeurs numériques en ligne: https://riptutorial.com/fr/actionscript-3/topic/1899/travailler-avec-des-valeurs-numeriques

# Chapitre 20: Travailler avec la chronologie

## Examples

### Référencement du scénario principal ou de la classe de document depuis d'autres MovieClips

Dans le scénario de tout objet `DisplayObject` attaché en tant que descendant de l'arborescence, vous pouvez utiliser la propriété `root` . Cette propriété pointe vers le scénario principal dans le cas d'aucune classe de document personnalisée ou de la classe de document si vous en définissez une.

Étant donné que `root` est typé `DisplayObject` , le compilateur ne vous permettra pas d'accéder aux méthodes ou propriétés personnalisées définies sur le scénario principal ou dans votre classe de document en tant que:

```
root.myCustomProperty = 10;
root.myCustomMethod();
```

Pour contourner ce problème, vous pouvez transtyper `root` dans votre classe de document dans le cas où vous avez une classe de document:

```
(root as MyDocumentClass).myCustomMethod();
```

Ou `MovieClip` dans le cas d'une classe sans document:

```
(root as MovieClip).myCustomMethod();
```

La raison pour laquelle la diffusion vers `MovieClip` fonctionne ici est que `MovieClip` est `dynamic` . Cela signifie que le compilateur autorise la déclaration des propriétés d'exécution et de la méthode, évitant les erreurs de compilation lors de la tentative d'accès aux propriétés ou aux méthodes non explicitement définies sur `MovieClip` . L'inconvénient de ceci est que vous perdez toute la sécurité du type à la compilation. Vous feriez bien mieux de déclarer une classe de document et de la publier.

Lire Travailler avec la chronologie en ligne: https://riptutorial.com/fr/actionscript-3/topic/2459/travailler-avec-la-chronologie

# Chapitre 21: Travailler avec la date et l'heure

## Examples

### Ante meridiem (AM) ou Post meridiem (PM) pour une horloge de 12 heures

```
function meridiem(d:Date):String {
    return (d.hours > 11) ? "pm" : "am";
}
```

### Nombre de jours dans le mois spécifié

```
/**
 * @param year   Full year as int (ex: 2000).
 * @param month  Month as int, zero-based (ex: 0=January, 11=December).
 */
function daysInMonth(year:int, month:int):int {
    return (new Date(year, ++month, 0)).date;
}
```

### Si l'année spécifiée est l'année bissextile

```
function isLeapYear(year:int):Boolean {
    return daysInMonth(year, 1) == 29;
}
```

### Si l'heure d'été est actuellement observée

```
function isDaylightSavings(d:Date):Boolean {
    var months:uint = 12;
    var offset:uint = d.timezoneOffset;
    var offsetCheck:Number;

    while (months--) {
        offsetCheck = (new Date(d.getFullYear(), months, 1)).timezoneOffset;

        if (offsetCheck != offset)
            return (offsetCheck > offset);
    }

    return false;
}
```

### Date de début du jour à minuit

```
function today(date:Date = null):Date {
    if (date == null)
        date = new Date();
```

```
    return new Date(date.fullYear, date.month, date.date, 0, 0, 0, 0);
}
```

Lire Travailler avec la date et l'heure en ligne: https://riptutorial.com/fr/actionscript-3/topic/1926/travailler-avec-la-date-et-l-heure

# Chapitre 22: Travailler avec la géométrie

## Examples

### Obtenir l'angle entre deux points

Utiliser les mathématiques de la vanille:

```
var from:Point = new Point(100, 50);
var to:Point = new Point(80, 95);

var angle:Number = Math.atan2(to.y - from.y, to.x - from.x);
```

En utilisant un nouveau vecteur représentant la différence entre les deux premiers:

```
var difference:Point = to.subtract(from);

var angle:Number = Math.atan2(difference.y, difference.x);
```

> Note: `atan2()` renvoie les radians, pas les degrés.

### Obtenir la distance entre deux points

Utiliser les mathématiques de la vanille:

```
var from:Point = new Point(300, 10);
var to:Point = new Point(75, 40);

var a:Number = to.x - from.x;
var b:Number = to.y - from.y;

var distance:Number = Math.sqrt(a * a + b * b);
```

En utilisant la fonctionnalité intégrée de `Point` :

```
var distance:Number = to.subtract(from).length; // or
var distance:Number = Point.distance(to, from);
```

### Conversion de radians en degrés

```
var degrees:Number = radians * 180 / Math.PI;
```

### Conversion de degrés en radians

```
var radians:Number = degrees / 180 * Math.PI;
```

## La valeur d'un cercle en degrés et radians

- Un cercle entier est de `360` degrés ou `Math.PI * 2` radians.
- La moitié de ces valeurs est `Math.PI 180` degrés ou `Math.PI` radians.
- Un quart est alors de `90` degrés ou `Math.PI / 2` radians.

Pour obtenir un segment en pourcentage d'un cercle entier en radians:

```
function getSegment(percent:Number):Number {
    return Math.PI * 2 * percent;
}

var tenth:Number = getSegment(0.1); // One tenth of a circle in radians.
```

## Déplacement d'un point selon un angle

En supposant que vous avez l'angle que vous souhaitez déplacer et un objet avec des valeurs `x` et `y` vous souhaitez déplacer:

```
var position:Point = new Point(10, 10);
var angle:Number = 1.25;
```

Vous pouvez vous déplacer sur l'axe des `x` avec `Math.cos` :

```
position.x += Math.cos(angle);
```

Et l'axe `y` avec `Math.sin` :

```
position.y += Math.sin(angle);
```

Vous pouvez bien sûr multiplier le résultat de `Math.cos` et `Math.sin` par la distance à parcourir:

```
var distance:int = 20;

position.x += Math.cos(angle) * distance;
position.y += Math.sin(angle) * distance;
```

> Remarque: L'angle d'entrée doit être en radians.

## Déterminer si un point est à l'intérieur d'un rectangle

Vous pouvez tester si un point se trouve dans un rectangle à l'aide de `Rectangle.containsPoint()` :

```
var point:Point = new Point(5, 5);
var rectangle:Rectangle = new Rectangle(0, 0, 10, 10);

var contains:Boolean = rectangle.containsPoint(point); // true
```

Lire Travailler avec la géométrie en ligne: https://riptutorial.com/fr/actionscript-

3/topic/3201/travailler-avec-la-geometrie

# Chapitre 23: Travailler avec la vidéo

## Examples

### Charger et lire un fichier vidéo externe

---

**référence** : `NetConnection` , `NetStream` , `Video`

**sujets connexes** : Travailler avec le son

---

Exemple de base de lecture d'un fichier vidéo externe (FLV, MP4, F4V). Le code lit également les fichiers audio M4A.

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var ns:NetStream = new NetStream(nc);

var myVideo:Video = new Video();
addChild(myVideo);

myVideo.attachNetStream(ns);

ns.play("http://www.yourwebsite.com/somefile.mp4");
```

Notez que le code a utilisé un `nc.connect.null` ? En effet, dans ce cas, il n'est pas nécessaire de créer une connexion pair à pair (par exemple: comme prévu dans une application de chat vidéo), car nous lisons un fichier stocké.

En définissant un `nc.connect.null` il est nécessaire de fournir un lien vers un fichier qui se trouve sur un serveur Web ou local (même emplacement / dossier) vers le fichier SWF en cours d'exécution.

- Pour un fichier **Web** , utilisez: `ns.play("http://www.yourwebsite.com/somefile.mp4");`
- Pour un fichier **local** , utilisez: `ns.play("somefile.mp4");`

### Avec NetStatusEvent

```
package {
    import flash.events.NetStatusEvent;
    import flash.net.NetStream;
    import flash.net.NetConnection;
    import flash.events.Event;
    import flash.media.Video;
    import flash.display.Sprite;
    public class VideoWithNetStatus extends Sprite {
```

```
        private var video:Video = new Video();
        private var nc:NetConnection;
        private var ns:NetStream;

        public function VideoWithNetStatus() {
            nc = new NetConnection();
            nc.addEventListener(NetStatusEvent.NET_STATUS, onStatus);
            nc.connect(null);//or media server url
        }

         private function onStatus(e:NetStatusEvent):void{
             switch(e.info.code){
                 case 'NetConnection.Connect.Success':
                     connectStream();
                 break;
                 default:
                     trace(e.info.code);//to see any unhadled events
             }
         }
         private function connectStream():void{
             ns = new NetStream(nc);
             ns.addEventListener(NetStatusEvent.NET_STATUS, onStatus);
             addChild(video);
             video.attachNetStream(ns);
             ns.play('url/to/video.flv');
         }
    }
}
```

Lire Travailler avec la vidéo en ligne: https://riptutorial.com/fr/actionscript-3/topic/2406/travailler-avec-la-video

# Chapitre 24: Travailler avec le son

## Syntaxe

- Sound.play (startTime: Number = 0, boucles: int = 0, sndTransform: flash.media: SoundTransform = null): SoundChannel // Lit un son chargé, renvoie un SoundChannel

## Examples

### Arrêtez de jouer un son

```
import flash.net.URLRequest;
import flash.media.Sound;
import flash.media.SoundChannel;
import flash.events.Event;

var snd:Sound; = new Sound();
var sndChannel:SoundChannel
var sndTimer:Timer;

snd.addEventListener(Event.COMPLETE, soundLoaded);
snd.load(new URLRequest("soundFile.mp3")); //load after adding the complete event

function soundLoaded(e:Event):void
{
    sndChannel = snd.play();

    //Create a timer to wait 1 second
    sndTimer = new Timer(1000, 1);
    sndTimer.addEventListener(TimerEvent.TIMER, stopSound, false, 0, true);
    sndTimer.start();
}

function stopSound(e:Event = null):void {
    sndChannel.stop(); //Stop the sound
}
```

### Boucle infinie un son

```
import flash.net.URLRequest;
import flash.media.Sound;
import flash.events.Event;

var req:URLRequest = new URLRequest("filename.mp3");
var snd:Sound = new Sound(req);

snd.addEventListener(Event.COMPLETE, function(e: Event)
{
    snd.play(0, int.MAX_VALUE); // There is no way to put "infinite"
}
```

Vous n'avez pas non plus besoin d'attendre que le son soit chargé avant d'appeler la fonction

`play()` . Donc, cela fera le même travail:

```
snd = new Sound(new URLRequest("filename.mp3"));
snd.play(0, int.MAX_VALUE);
```

Et si vous voulez vraiment créer un son inifinite en boucle pour une raison quelconque ( `int.MAX_VALUE` va boucler le son des boucles pendant environ 68 ans, sans compter la pause `int.MAX_VALUE` un mp3 ...), vous pouvez écrire quelque chose comme ceci:

```
var st:SoundChannel = snd.play();
st.addEventListener(Event.SOUND_COMPLETE, repeat);
function repeat(e:Event) {
    st.removeEventListener(Event.SOUND_COMPLETE, repeat);
    (st = snd.play()).addEventListener(Event.SOUND_COMPLETE, repeat);
}
```

`play()` renvoie une nouvelle instance de l'objet `SoundChannel` chaque appel. Nous l'assignons à variable et écoutons son événement SOUND_COMPLETE. Dans le rappel d'événement, l'écouteur est supprimé de l'objet `SoundChannel` actuel et un autre est créé pour le nouvel objet `SoundChannel` .

## Charger et lire un son externe

```
import flash.net.URLRequest;
import flash.media.Sound;
import flash.events.Event;

var req:URLRequest = new URLRequest("click.mp3");
var snd:Sound = new Sound(req);

snd.addEventListener(Event.COMPLETE, function(e: Event)
{
    snd.play();
}
```

Lire Travailler avec le son en ligne: https://riptutorial.com/fr/actionscript-3/topic/2043/travailler-avec-le-son

# Chapitre 25: Utilisation de la classe proxy

## Introduction

Tout d'abord, je dois dire. Il **y a** une raison pour laquelle ce truc, malgré son utilité apparente, n'est pas suffisamment mis en évidence sur Internet.

Vous **ne pouvez pas l'** utiliser pour regarder une propriété aléatoire d'une classe / instance aléatoire. Vous ne pouvez utiliser cette technique qu'en sous- **classant la** classe **Proxy** .

Certaines opérations n'attendent pas d'exceptions, vous devez donc comprendre complètement ce que vous faites et pourquoi vous le faites, et votre code **doit** être absolument propre et sans erreur.

## Examples

### la mise en oeuvre

L'autre chose à propos de la classe Proxy, et pourquoi elle n'est pas si populaire, est qu'il est plutôt difficile d'imaginer un problème qui nécessite une classe dynamique avec un accès contrôlable à ses propriétés et méthodes dynamiques. Chaque fois que j'ai essayé d'utiliser Proxy, j'ai fini par recourir à autre chose, plus simple et plus contrôlable.

Cependant, ne nous décourageons pas. J'aime l'idée d'adresser les derniers éléments du tableau par les indices [-1], [-2], etc. dans **Python** . Ce n'est peut-être pas un gros exploit, mais ça fait du bien d'utiliser ça plutôt que de longs et maladroits **someArray [someArray.length - 1]** . Voyons ce que nous pouvons faire à ce sujet.

```
package
{
    import flash.utils.Proxy;
    import flash.utils.flash_proxy;

    /**
     * Pyaray the Tentacled Whisperer of Impossible Secrets.
     */

    dynamic public class PyArray extends Proxy
    {
        private var data:Array;

        public function PyArray(...args:Array)
        {
            if (args.length == 0)
            {
                data = new Array;
            }
            else if ((args.length == 1) && (args[0] is Array))
            {
                data = args[0];
```

```
        }
        else
        {
            data = args;
        }
    }

    // This is a getter proxy to all the available Array
    // elements and properties and, sometimes, methods.
    override flash_proxy function getProperty(name:*):*
    {
        var anIndex:int = name;

        // Handle the int indices of the Array elements.
        if (anIndex == name)
        {
            // Handle the -1, -2, etc indexing.
            if (anIndex < 0) anIndex += data.length;

            if (anIndex >= data.length) return null;
            if (anIndex < 0) return null;

            return data[anIndex];
        }

        // Handle the existing public Array properties.
        if (data.hasOwnProperty(name)) return data[name];

        // Handle the Array methods addressed via ["member"] access.
        try
        {
            if (data[name] is Function) return data[name];
            else throw new Error;
        }
        catch (fail:Error)
        {
            trace("[PyArray] is unable to resolve property \"" + name + "\".");
        }

        return null;
    }

    // This will set either elements, or settable properties.
    override flash_proxy function setProperty(name:*, value:*):void
    {
        var anIndex:int = name;

        // Handle the int indices of the Array elements.
        if (anIndex == name)
        {
            // Handle the -1, -2, etc indexing.
            if (anIndex < 0) anIndex += data.length;

            // In case the element index is out of range,
            // the PyArray will extend its data Array.
            // if (anIndex >= data.length) return;
            if (anIndex < 0) return;

            data[anIndex] = value;

            return;
```

```
        }

        // Handle the existing (or dynamic) public Array properties.
        try
        {
            data[name] = value;
        }
        catch (fail:Error)
        {
            trace("[PyArray] is unable to set property \"" + name + "\".");
        }

        return;
    }

    // This allows to delete PyArray elements with "delete" operator.
    override flash_proxy function deleteProperty(name:*):Boolean
    {
        var anIndex:int = name;

        // Handle the int indices of the Array elements.
        if (anIndex == name)
        {
            // Handle the -1, -2, etc indexing.
            if (anIndex < 0) anIndex += data.length;

            if (anIndex >= data.length) return false;
            if (anIndex < 0) return false;

            data.splice(anIndex, 1);

            return true;
        }

        // Handle the dynamic public Array properties.
        try
        {
            delete data[name];
            return true;
        }
        catch (fail:Error)
        {
            trace("[PyArray] is unable to delete the \"" + name + "\" property.");
        }

        return false;
    }

    // This proxies any attempt to call a method on PyArray directly to data Array, thus
    // all Array methods (including "toString" method called through trace) are available.
    override flash_proxy function callProperty(name:*, ...rest):*
    {
        try
        {
            return (data[name] as Function).apply(data, rest);
        }
        catch (fail:Error)
        {
            trace("[PyArray] is unable to resolve method \"" + name + "\".");
            return null;
        }
```

```
        }

        // This allows PyArray to handle for..in and for..each..in loops.
        // The initial call starts with zero, so we need to do this +1 -1 magic
        // in order for enumeration to work correctly. I'm not happy with this either.
        override flash_proxy function nextNameIndex(index:int):int
        {
            if (index >= data.length) return 0;
            else return index + 1;
        }

        // This method handles the for..in loop.
        override flash_proxy function nextName(index:int):String
        {
            return (index - 1).toString();
        }

        // This method handles the for..each..in loop.
        override flash_proxy function nextValue(index:int):*
        {
            return data[index - 1];
        }
    }
}
```

## Usage

```
package
{
    import flash.display.Sprite;

    /**
     * Daemonette of Slaanesh.
     *
     * It is minor female demon, vaguely human-like, but with crab-like pincers instead of
hands.
     * She wears a rather indecent skimpy leather bikini, moves quickly and casts deadly
spells!
     */

    public class Slaanesh extends Sprite
    {
        public function Slaanesh()
        {
            // Lets initialize the PyArray.
            var PA:PyArray = new PyArray(1,2,3,4,5,4,3,2,1,"Foo");

            // Basic check: get the last element.
            trace(PA[-1]);
            // output:
            // Foo

            // This will map to the 0-based third element.
            trace(PA[2.0]);
            // output:
            // 3

            // This should not get us anywhere.
            trace(PA[2.1]);
```

```
// output:
// [PyArray] is unable to resolve property "2.1".
// null

// This should return the length of the data Array.
trace(PA["length"]);
// output:
// 10

// This should return the length of the data Array.
// This will not compile unless PyArray class is marked "dynamic".
trace(PA.length);
// output:
// 10

// This will map to indexOf method of data Array via getProperty method.
trace(PA["indexOf"]);
// output:
// function Function() {}

// This will map to indexOf method of data Array via getProperty method.
// This will not compile unless PyArray class is marked "dynamic".
trace(PA.indexOf);
// output:
// function Function() {}

// This is a try to access a non-existent property.
// This will not compile unless PyArray class is marked "dynamic".
trace(PA.P124);
// output:
// [PyArray] is unable to resolve property "P124".
// null

// This is a try to call a non-existent method via callProperty method.
// This will not compile unless PyArray class is marked "dynamic".
trace(PA.P124());
// output:
// [PyArray] is unable to resolve method "P124".
// null

// Basic check: calling a proxied method via callProperty method.
trace(PA.indexOf(5));
// output:
// 4

// An attempt to replace an Array method with a random value.
// This will not compile unless PyArray class is marked "dynamic".
PA.indexOf = 123;
// output:
// [PyArray] is unable to set property "indexOf".

// An attempt to assign a random value to a random property.
// It will succees because Array, as a dynamic class, allows so.
// This will not compile unless PyArray class is marked "dynamic".
PA.indexOfz = 123;
trace(PA.indexOfz);
// output:
// 123

// An attempt to assign an Array element via negative indexing.
// This trace works fine because toString method is also proxied.
```

```
            PA[-3] = "Hello";
            trace(PA);
            // output:
            // 1,2,3,4,5,4,3,Hello,1,Foo

            // An attempt to delete Array elements via normal and negative indexing.
            // This operation is mapped via deleteProperty method.
            delete PA[-4]; // deletes "3" before "Hello"
            delete PA[0];  // deletes "1" at the start.
            trace(PA);
            // output:
            // 2,3,4,5,4,Hello,1,Foo

            // An attempt to delete a non-dynamic method reference.
            // There's no error output, AS3 must be handling this internally.
            delete PA.indexOf;
            trace(PA.indexOf);
            // output:
            // function Function() {}

            // An attempt to set an element out of index range.
            PA[10] = "123abc";
            trace(PA);
            // output:
            // 2,3,4,5,4,Hello,1,Foo,,,123abc

            var aText:String;

            // This is a test of for..in loop, Array elements are
            // enumerated via nextName and nextNameIndex methods.
            aText = ""

            for (var aKey:String in PA)
                aText += aKey + ":" + PA[aKey] + " ";

            trace(aText);
            // output:
            // 0:2 1:3 2:4 3:5 4:4 5:Hello 6:1 7:Foo 8:undefined 9:undefined 10:123abc

            // This is a test of for..each..in loop, Array elements are
            // enumerated via nextValue and nextNameIndex methods.
            aText = "";

            for each (var aValue:* in PA)
                aText += aValue + " ";

            trace(aText);
            // output:
            // 2 3 4 5 4 Hello 1 Foo undefined undefined 123abc
        }
    }
}
```

Lire Utilisation de la classe proxy en ligne: https://riptutorial.com/fr/actionscript-3/topic/10631/utilisation-de-la-classe-proxy

# Crédits

| S. No | Chapitres | Contributeurs |
|---|---|---|
| 1 | Mise en route avec ActionScript 3 | BadFeelingAboutThis, Community, Jason Sturges, joshtynjala, Kit Grose, null, Programmer Dancuk, Vesper |
| 2 | Chargement de fichiers externes | BadFeelingAboutThis, Marty |
| 3 | Comprendre le "Erreur 1009: Impossible d'accéder à une propriété ou une méthode d'une référence d'objet null" | Vesper, www0z0k |
| 4 | Conception d'applications réactive | BadFeelingAboutThis, null, Vesper |
| 5 | Cycle de vie de la liste d'affichage | Jason Sturges, mnoronha |
| 6 | Dessin Bitmaps | BadFeelingAboutThis, Marty, Vesper, www0z0k |
| 7 | Données binaires | Paweł Audionysos |
| 8 | Envoi et réception de données à partir de serveurs | Marty, null, xims |
| 9 | Génération de valeur aléatoire | alebianco, BadFeelingAboutThis, HITMAN, Jason Sturges, Marty, mnoronha, null, Vesper, xims |
| 10 | Les bases du développement de jeux | payam_sbr |
| 11 | Les types | BadFeelingAboutThis, joshtynjala, Marty, Paweł Audionysos |
| 12 | Manipulation et filtrage de bitmap | payam_sbr, VC.One |
| 13 | Motif Singleton | commovere, Jason Sturges, mnoronha, null |

| | | |
|---|---|---|
| 14 | Optimiser les performances | Community, Marty |
| 15 | Programmation orientée objet | HITMAN, Marty, null, www0z0k |
| 16 | Travailler avec des événements | blue112, Jason Sturges, mnoronha, null, VC.One, Vesper, Zze |
| 17 | Travailler avec des minuteries | Jason Sturges, mnoronha, Vesper, www0z0k |
| 18 | Travailler avec des objets d'affichage | BadFeelingAboutThis, Jason Sturges, Vesper |
| 19 | Travailler avec des valeurs numériques | Jason Sturges, Jonny Henly, Marty, Vesper |
| 20 | Travailler avec la chronologie | Marty |
| 21 | Travailler avec la date et l'heure | Jason Sturges, mnoronha |
| 22 | Travailler avec la géométrie | Marty, mnoronha, www0z0k |
| 23 | Travailler avec la vidéo | VC.One, www0z0k |
| 24 | Travailler avec le son | BadFeelingAboutThis, blue112, Paweł Audionysos, VC.One |
| 25 | Utilisation de la classe proxy | Organis |