

La nouvelle architecture du framework ActionScript 3

par Olivier Bugalotto ([Mes articles](#))

Date de publication : 26/02/2007

Article décrivant l'architecture du framework ActionScript 3

I - L'architecture

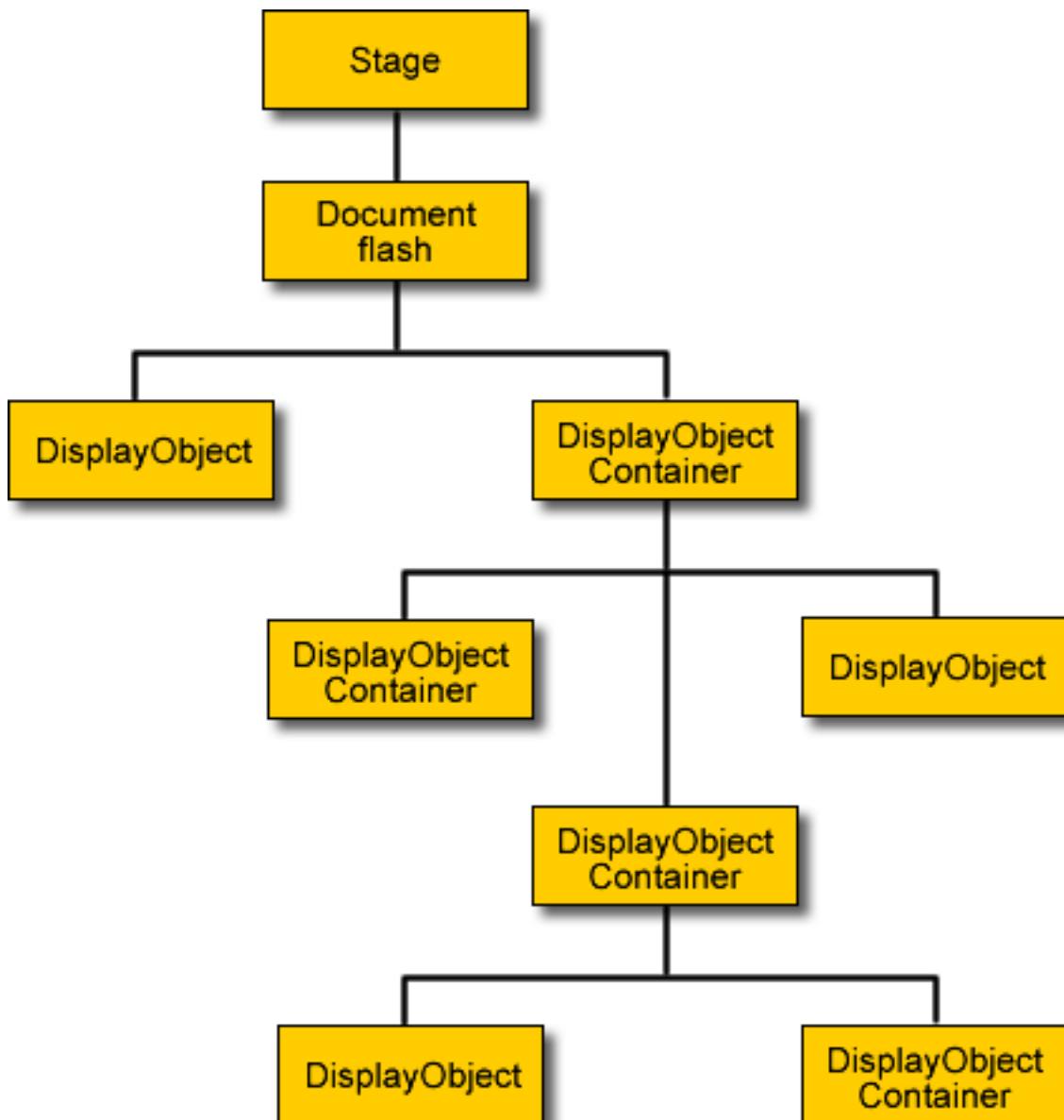
II - Exemple

III - Téléchargement

I - L'architecture

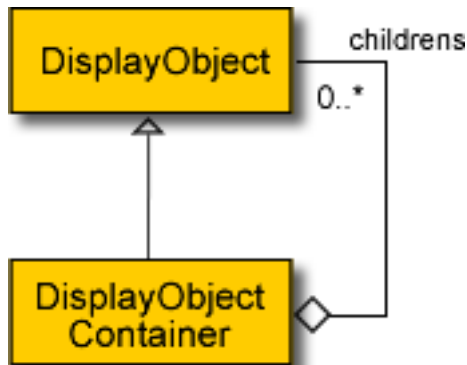
Dans la nouvelle architecture, une application flash possède une liste d'affichage ("display list"). Cette liste contient tous les éléments visibles. Ces éléments se repartissent en 2 types :

- **DisplayObject** : élément visible qui se caractérise par sa position, ses dimensions, etc.
- **DisplayObjectContainer** : zone rectangulaire qui peut contenir aussi bien des DisplayObject que des DisplayObjectConteneur.



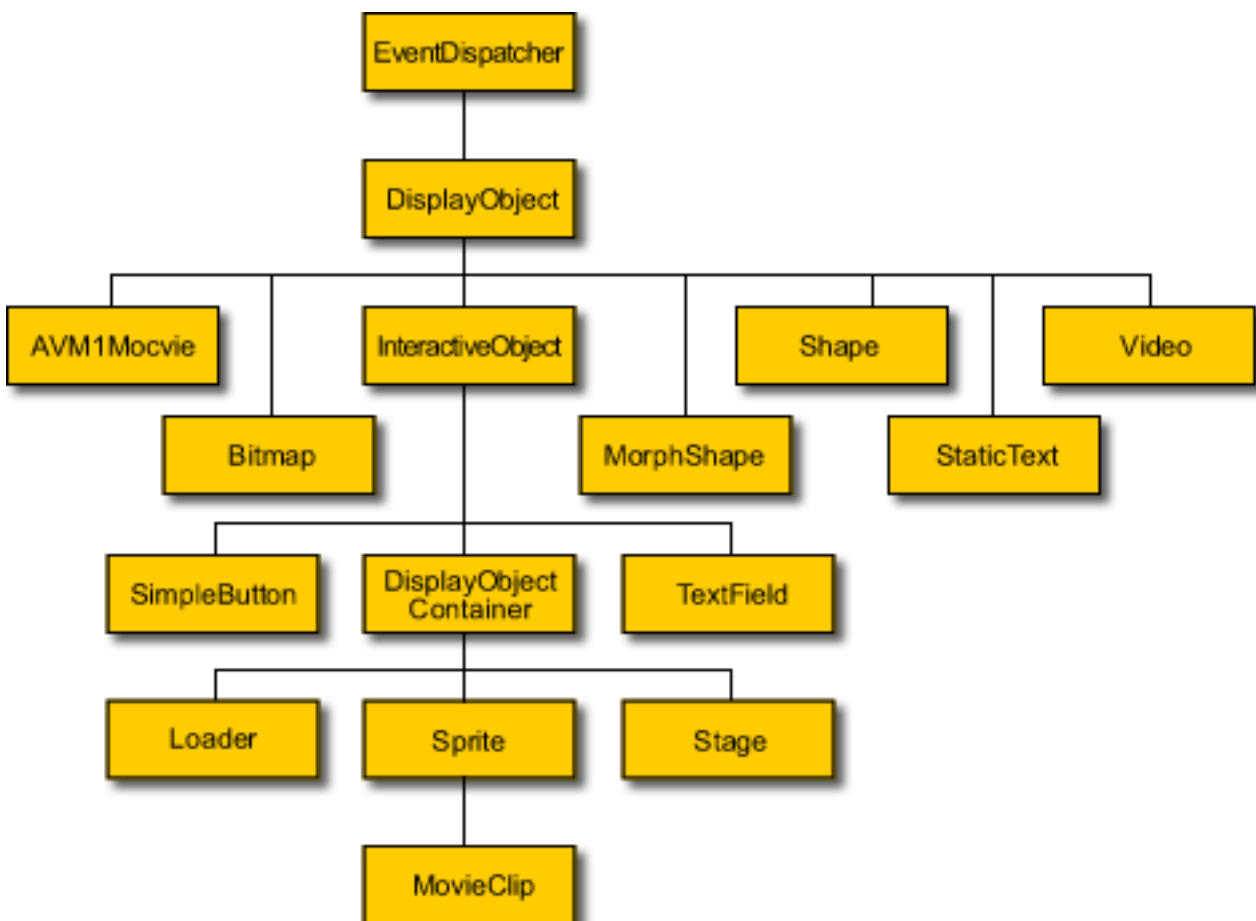
En regardant cette liste d'affichage, nous pourrions la comparer à un arbre dans lequel les feuilles sont les DisplayObjects et les nœuds sont les DisplayObjectContainers.

D'ailleurs cette structure est basée sur le **design pattern Composite** :



A la base de cette liste d'affichage, nous trouvons le type Stage qui est le conteneur principal. Nous pouvons le comparer au `_root` de cette application. L'accès a ce conteneur principal se fait à l'aide de la propriété `stage` de tous les `DisplayObjects`.

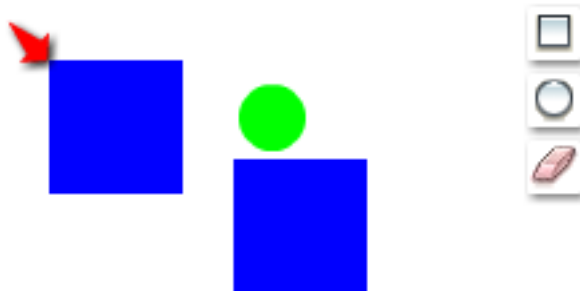
Mais où sont dans tout ça, les `MovieClip`, les `TextField`, etc. Voici un schéma qui nous les situe :



Nous remarquerons au passage que tout est `EventDispatcher`, ce qui signifie que tous les éléments de ce schéma peuvent émettre des évènements.

II - Exemple

Nous allons voir à travers une application, les comportements que nous proposent `DisplayObject` et `DisplayObjectContainer`. Dans cette application, 3 outils en haut à droite nous permettent d'ajouter et retirer des formes rectangulaires et ovales. Nous pouvons sélectionner une forme, cette sélection est indiquée par une flèche rouge, et lui ajouter ainsi d'autres formes grâce au comportement des `DisplayObjectContainer` sachant que chaque forme peut être déplacée :



J'ai d'abord créé une classe `Application` qui hérite de la classe `Sprite` que je passe en tant que document flash (voir tutorial : [La classe document dans Flash 9](#)). Lorsque la fonction constructeur est appelée, je crée 3 outils, je laisse la classe `Tool` pour un prochain tutorial, voici la fonction constructeur commentée :

```
public function Application() {
    // Creation d'une surface de travail
    // C'est dans celui-ci que nous allons ajouter nos formes
    screen = new Square(250,250,0xFFFFFFFF);
    screen.name = "screen";

    // Comme je le disais tout est EventDispatcher
    // Ici nous ecoutons les evenements de pression et relachement
    // de la souris
    screen.addEventListener(MouseEvent.CLICK, selectShape);
    screen.addEventListener(MouseEvent.CLICK, stopDragShape);

    // Definit par default la forme active
    activeShape = screen;

    // Crétion des 3 outils pour l'ajout des formes rectangulaire, ovales et la suppression
    toolAddRect = new Tool(new RectangleIcon());
    toolAddRect.commandType = "rect";
    toolAddRect.addEventListener(MouseEvent.CLICK, addShape);
    toolAddRect.move(225,10);

    toolAddOval = new Tool(new OvalIcon());
    toolAddOval.commandType = "oval";
    toolAddOval.addEventListener(MouseEvent.CLICK, addShape);
    toolAddOval.move(225,35);

    toolEraser = new Tool(new EraserIcon());
    toolEraser.commandType = "erase";
    toolEraser.addEventListener(MouseEvent.CLICK, eraseShape);
    toolEraser.move(225,60);
    //

    // Instancie un élément graphique présent dans la bibliothèque
}
```

```
ptr = new Pointeur();
ptr.name = "pointeur";
    // Nous faisons en sort de désactive l'utilisation de la souris sur cet élément
ptr.mouseEnabled = false;

    // Ajout des elements à notre liste d'affichage
addChild(screen);
addChild(toolAddRect);
addChild(toolAddOval);
addChild(toolEraser);
}
```

Examinons les gestionnaires d'évènements :

```
private function selectShape(e:MouseEvent):void {
    // Nous verifions si l'element actif contient une fleche rouge
    // si c'est la cas nous la retirons à l'aide de la méthode removeChild
    // comportement de la classe DisplayObjectContainer
    // Cette méthode nous demande la référence à la flêche rouge
if(activeShape.contains(ptr)) {
    activeShape.removeChild(ptr);
}

    // Si la cible de l'évènement est screen
    // Nous l'attribuons en tant que forme active
if(e.target == screen) {
    activeShape = screen;
} else {
    // Dans le cas contraire
    // nous attribuons la cible de l'évènement
    // comme forme active
    activeShape = e.target;
    // Nous lui ajoutons une flêche pour indiquer
    // quelle est la forme active
    activeShape.addChild(ptr);
    // Et nous pouvons commencer à la deplacer
    activeShape.startDrag();
}
}

private function stopDragShape(e:MouseEvent):void {
    // Nous arretons le déplacement de la forme active
    // Lorsque le bouton gauche de votre souris est relachée
    activeShape.stopDrag();
}
}
```

Regardons la méthode executée lorsque nous supprimons une forme :

```
private function eraseShape(e:ToolEvent):void { // ToolEvent est un evenement personnalise
    // Nous vérifions que nous ne sommes pas sur le screen
    // afin d'éviter de le supprimer
if(activeShape != screen) {
    // Nous recuperons le parent de la forme active
    // a l'aide de la propriete parent de DisplayObjectContainer
    var container:DisplayObjectContainer = activeShape.parent;
    // Le parent nous permet ainsi de supprimer la forme active à l'aide de la
    // méthode removeChild qui nous demande la référence de cette dernière
    container.removeChild(activeShape);
}
}
```

III - Téléchargement

Nous venons de voir à travers cet exemple, la nouvelle architecture de AS3, pour ceux qui sont intéressés par le code, vous le trouverez ici : **[Architecture.zip](#)** (**[Miroir](#)**)

