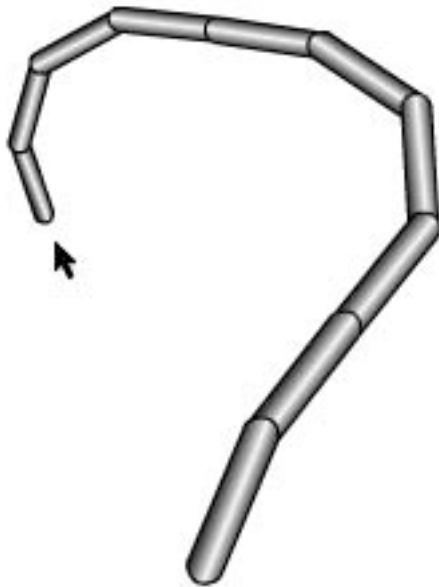


ARM

Tutoriel de Barbara Sansone www.headroom.ws/slice0
à partir du fichier 031222.fla de Keith Peters www.bit-101.com
Traduit de l'italien par Chantal Emonet <http://e.chantal.free.fr>

Pour suivre ce tutorial, ouvrir le fichier 031222.fla contenu dans le dossier start et écrire tout le code ou bien le lire en ouvrant le fichier 031222.fla contenu dans le dossier end.



Tout le code est regroupé en un script inclus dans l'image 1 de l'unique calque existant du scénario.

1. Avant tout il faut décider de combien de morceaux le bras sera composé. Cette valeur peut être changée à n'importe quel moment, c'est pour cela qu'il convient de la conserver à l'intérieur d'une variable déclarée au début du script:

```
num = 10;
```

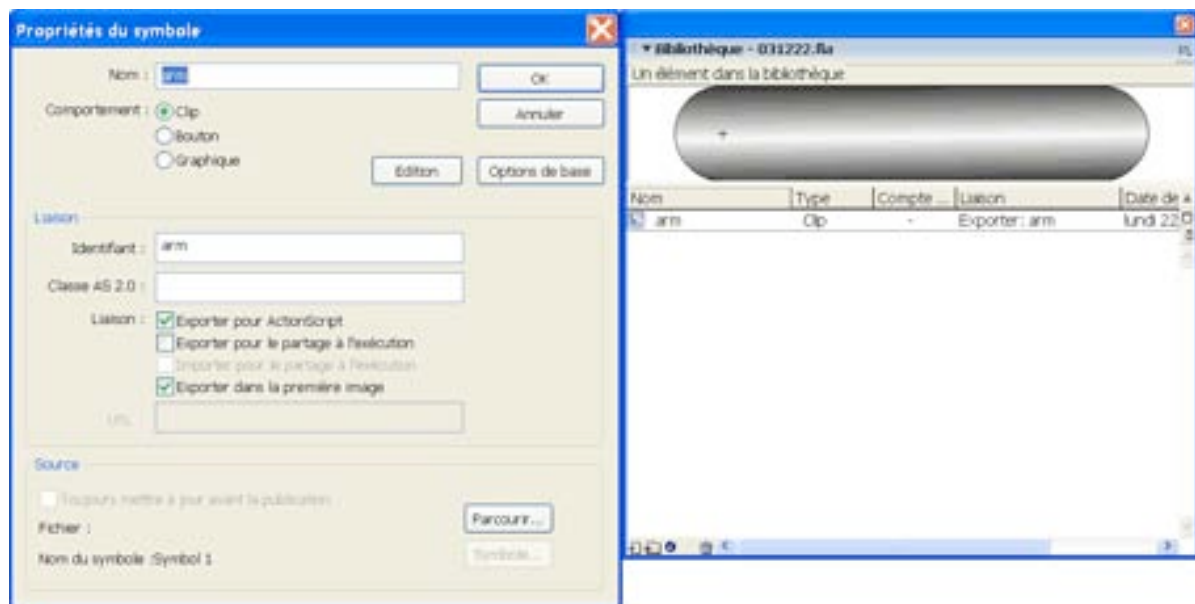
2. Les morceaux ne seront pas tous égaux en dimension. Par rapport au symbole original, il faut décider un facteur d'échelle qui sera incrémenté pendant qu'Action Script dessinera les occurrences. Cette valeur aussi, pour accepter facilement des variations successives, doit être conservée dans une variable:

```
scale = 30;
```

3. Les actions nécessaires pour dessiner les occurrences et attribuer leurs mouvements, avec quelques astuces, sont les mêmes pour tous les morceaux. Il est donc nécessaire d'utiliser une boucle qui utilise la valeur de num:

```
for(var i=0;i<num;i++){
```

4. La première action de la boucle consiste à créer l'occurrence d'un symbole. Le clip d'animation arm a été opportunément équipé d'un identifiant nécessaire pour l'exportation Action Script, comme on peut le remarquer par la propriété liaison, repérable au menu des options de la bibliothèque.



5. Les occurrences seront nommées progressivement « a0 », « a1 », et ainsi de suite, ajoutant au caractère « a » la valeur courante de i:

```
arm = attachMovie("arm", "a"+i, i);
```

6. L'occurrence créée dans la boucle s'appellera simplement arm. La valeur de cette variable sera, à chaque boucle, le nom de l'occurrence à peine créée. Maintenant on attribue à l'occurrence des facteurs d'échelle x et y égaux à la valeur d'échelle.

```
arm._xscale = arm._yscale = scale;
```

7. Dans la variable `w` à l'intérieur de l'objet, on conserve une valeur égale à un pourcentage déterminé de la longueur de l'objet (ici de 80%, mais elle peut être modifiée pour vérifier l'effet). Cela permettra de régler la distance entre les différents morceaux du bras:

```
arm.w = arm._width * .8;
```

8. Le facteur `scale` doit s'incrémenter, pour faire que le morceau créé à la boucle suivante soit plus grand. Cette valeur peut être modifiée pour créer des objets d'aspect divers:

```
scale +=3;
```

9. Toutes les actions qui règlent la position et la rotation de l'occurrence dont on parle peuvent être recueillies dans une fonction qui sera reportée successivement dans le script. Ici, il sera suffisant de l'appeler:

```
arm.onEnterFrame = move;
```

10. Les morceaux devront se conditionner mutuellement : le premier, en effet suivra la souris, mais sa position et sa rotation influencera celle du morceau successif. Il est donc utile de créer la variable `parent` (à ne pas confondre avec `_parent`), qui résidera dans l'occurrence qui aura le nom avec le numéro inférieur et aura la valeur de l'occurrence en cours. Plus simplement, l'occurrence « `a0` » contiendra la variable `parent` qui aura comme valeur « `a1` ». De cette manière, durant les actions relatives à « `a0` », la variable `parent` peut être utilisée pour faire référence au morceau successif du bras, « `a1` »:

```
_root["a"+(i-1)].parent = arm;
```

11. Refermer le cycle:

```
}
```

12. A la fin de la boucle, la dernière valeur de `arm` était « `a9` ». En utilisant encore cette variable (si l'on modifiait le nombre des morceaux, le nom de l'occurrence serait différent), les coordonnées du plus grand morceau se placent au centre de la scène:

```
arm._x = 270;  
arm._y = 380;
```

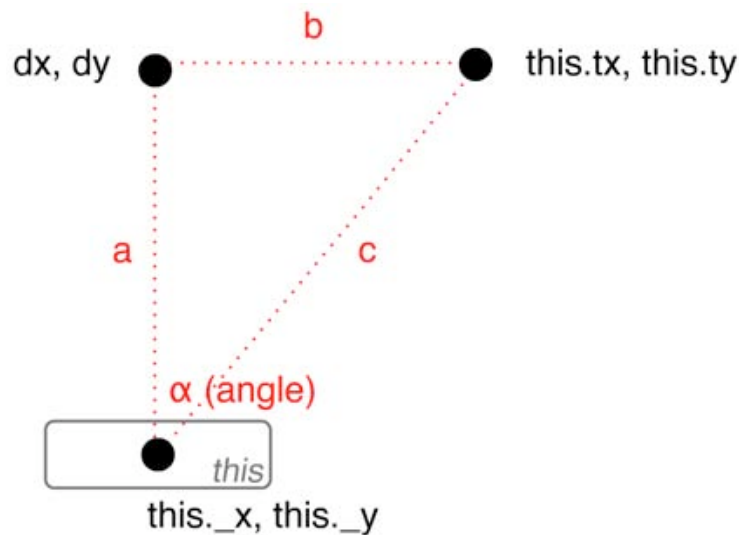
13. Maintenant on peut écrire la fonction `move`:

```
function move() {
```

14. Maintenant il s'agit de déterminer le point d'attraction. Cela devra être défini par les coordonnées x et y , qui seront conservées respectivement dans les variables tx et ty . Le point d'attraction doit être clairement différent pour chaque morceau de bras. Le plus petit, « $a0$ », sera attiré par les coordonnées de la souris. Selon l'angle que cette occurrence assumera, le point d'attraction du morceau successif, « $a1$ », sera déterminé et ainsi de suite.

Le point d'attraction de « $a0$ » sera déterminé à la fin du script dans un handler `onEnterFrame`, qui permettra d'enregistrer continuellement les coordonnées de la position du pointeur. Le calcul du point d'attraction du morceau parent permettra d'utiliser la même variable avec une nouvelle valeur à la boucle suivante.

Avant tout il faut trouver l'angle d'inclinaison que `arm` doit assumer pour se diriger vers le point d'attraction. Pour cela, il faut imaginer que le point d'attraction et le point d'alignement de l'occurrence dessinent un triangle rectangle imaginaire.



15. Les deux autres côtés de ce triangle peuvent être calculés en soustrayant les coordonnées respectives horizontales et verticales du point d'attraction et du point d'alignement de l'objet:

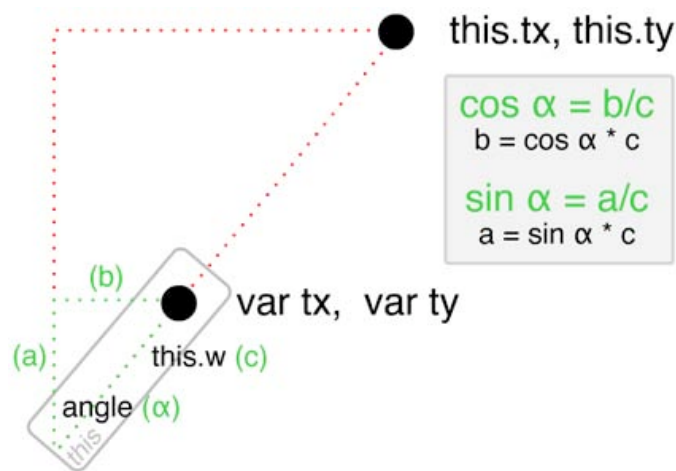
```
var dx = this.tx - this._x;  
var dy = this.ty - this._y;
```

16. A ce moment là, parce que les valeurs de dx et dy sont connues, il est possible d'obtenir l'angle $_$ en calculant l'arc tangente:

```
var angle = Math.atan2(dy, dx);
```

17. Maintenant on suppose que l'objet a été retourné. Il faut que le point d'attraction, pour le morceau successif, soit placé à 80% de sa longueur, raison pour laquelle a été créée la variable `w` précédemment. Maintenant, on connaît un angle et l'hypoténuse mais pas les deux autres côtés, nécessaires pour calculer la nouvelle valeur de `tx` et `ty`. Ces deux côtés peuvent être calculés en multipliant respectivement le cosinus et le sinus de l'angle par l'hypoténuse (`this.w`). La valeur obtenue est donc soustraite aux valeurs précédentes de `this.tx` et `this.ty` pour obtenir les coordonnées du point:

```
var tx = this.tx - Math.cos(angle)*this.w;  
var ty = this.ty - Math.sin(angle)*this.w;
```



18. Maintenant on peut attribuer les valeurs obtenues aux variables `tx` et `ty` résidentes dans l'objet parent, c'est-à-dire du morceau successif dans la chaîne (celui avec le numéro plus grand). De cette manière, à la boucle suivante, quand l'objet parent sera devenu `this`, elles serviront ultérieurement pour calculer les prochaines valeurs:

```
this.parent.tx = tx;  
this.parent.ty = ty;
```

19. La méthode `atan2` de l'objet `Math` restitue une valeur en radians, pendant que la propriété `_rotation` d'un objet est mesurée en grades. Donc, comme l'angle et la valeur de la propriété `_rotation` devront être traités ensemble, il convient d'uniformiser les unités de mesure, en portant angle en grades:

```
var targetRotation = angle*180/Math.PI;
```

20. Il faut soustraire à `targetRotation` l'angle actuel de rotation de l'objet pour obtenir l'angle de variation de la rotation que doit subir l'objet:

```
var diff = targetRotation-this._rotation;
```

21. Si la valeur obtenue est supérieure à 180, il faut lui soustraire 360 (on utilise while plutôt que if pour que la procédure se fasse tant que la condition est vraie), alors que si la valeur obtenue est plus petite de -180, il faut l'additionner à 360. De cette manière, on évite des fonctionnements bizarres quand, dans le passage successif, on attribuera une vitesse:

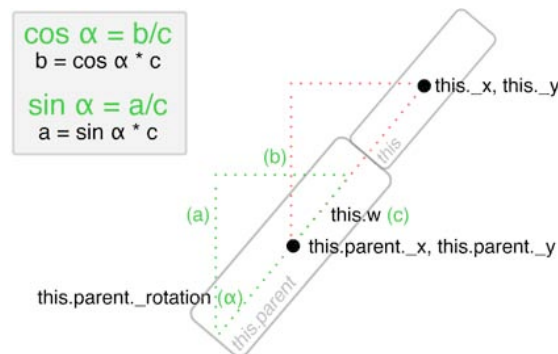
```
while (diff>180) {  
    diff -= 360;  
}  
while (diff<-180) {  
    diff += 360;  
}
```

22. Pour adoucir le mouvement, mieux vaut ajouter un facteur de frottement. Il s'agit d'une très petite valeur, avec laquelle on multipliera le facteur de rotation (ici on propose 0,1, égal à 10% de l'angle, mais il est possible d'essayer de modifier cette valeur pour obtenir des effets diverses.) Quand l'angle diminue, sa vitesse diminue aussi, donnant un caractère plus organique à l'objet. Pour les valeurs inférieures à 2, cette variation serait tellement petite qu'elle ne serait pas visible, c'est pour cela qu'il est possible de ne pas la calculer en utilisant une structure conditionnelle:

```
if (Math.abs(diff)>2) {  
    this.vr = diff*.1;  
    this._rotation += this.vr;  
}
```

23. Enfin, il faut calculer la position de l'objet, à partir des coordonnées du parent. Avec la même formule que précédemment, on récupère les deux plus petits côtés du triangle rectangle grâce à la longueur w du parent et sa rotation. Donc on ajoutera ces valeurs aux coordonnées x et y du parent, pour obtenir les nouvelles coordonnées x et y de l'objet this:

```
this._x = this.parent._x + Math.cos(this.parent._rotation*Math.PI/180) *  
this.parent.w;  
this._y = this.parent._y + Math.sin(this.parent._rotation*Math.PI/180) *  
this.parent.w;  
}
```



24. Les coordonnées tx et ty du point d'attraction relatif au premier morceau du bras, "a0", doivent coïncider avec les coordonnées du pointeur. Comme cela a été dit précédemment, le contrôle doit se faire continuellement et est donc inséré dans un handler onEnterFrame:

```
onEnterFrame = function () {  
    a0.tx = _xmouse;  
    a0.ty = _ymouse;  
}
```