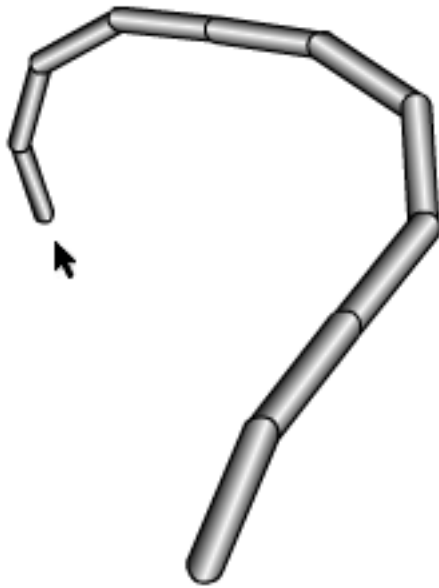


ARM

Tutorial von Barbara Sansone www.headroom.ws/slice0
Die 031222 fla Datei von Keith Peters betreffend, www.bit-101.com

Um diesem Tutorial zu folgen, bitte das in der Start-Datei 031222 fla File öffnen.



Der gesamte Code befindet sich in einem Script, das an den Frame 1 von der einzigen Ebene der Timeline angehängt ist.

1. Zunächst muss man entscheiden, aus wie viele Teilen der Arm bestehen wird. Dieser Wert ist jederzeit zu ändern: aus diesem Grund ist es besser, dass eine Variable den Wert erhält, die am Anfang der Script-Datei deklariert wird:

```
num = 10;
```

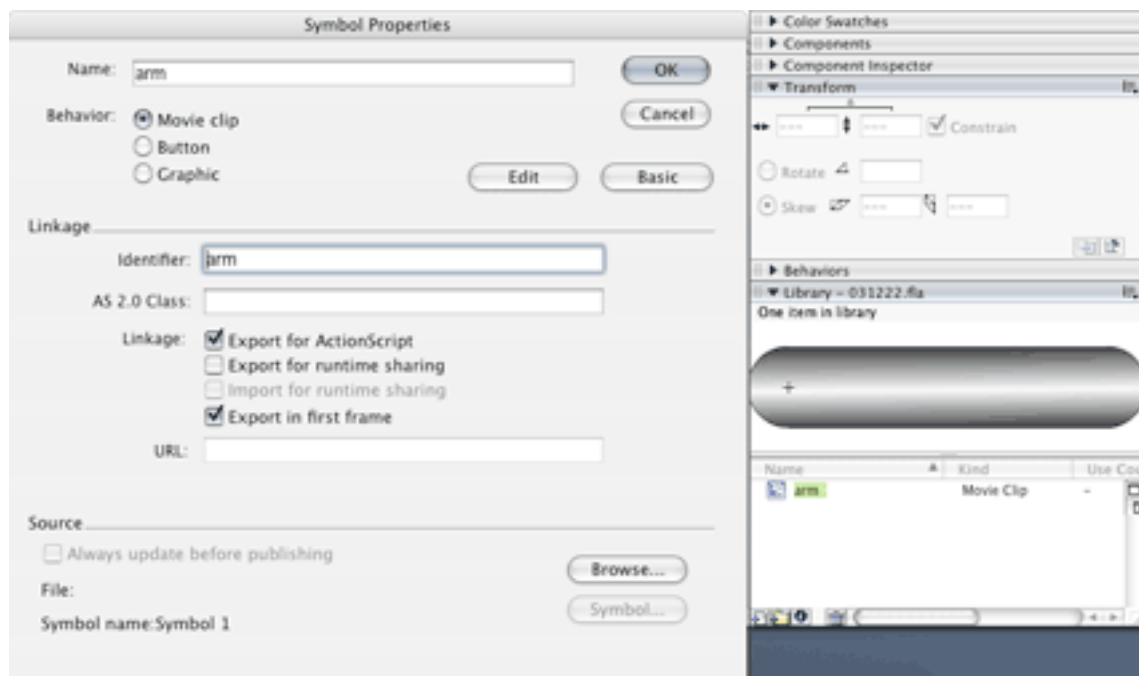
2. Jeder Teil wird eigene Dimensionen haben. Im Bezug auf das originale Symbol muss man einen Skalierungsfaktor feststellen, der steigert, indem ActionScript die Instanzen zeichnet. Auch dieser Wert muss in einer Variable erhalten werden, um ihn später leichter modifizieren zu können:

```
scale = 30;
```

3. Die erforderlichen Aktionen, um Instanzen zu zeichnen und ihnen bestimmte Bewegungen zu zuweisen, sind die gleichen für alle Teile. Man muss dann einen Zyklus benutzen, der den folgenden *num* Wert gebraucht:

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

4. Die erste Aktion des Zyklus besteht im Generieren eine Symbolinstanz. Die *arm* MovieClip wurde mit einer ID ausgestattet, die erforderlich für den Export von ActionScript ist, so wie die Linkage Eigenschaft, die unter den Menüoptionen der Bibliothek zu finden ist, zeigt.



5. Die Instanzen werden von "a0" ab progressiv genannt: außerdem wird der String "a" den aktuellen Wert von *i* hinzugefügt:

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

6. Die im Zyklus geschaffene Instanz wird einfach *arm* genannt. Diesem Variablenwert entspricht, in jedem Zyklus, der Name der zur letzten geschaffenen Instanz. Die Instanz bekommt *x* und *y* Skalierungsfaktoren, die den folgenden *scale* wert entsprechen:

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

7. Die *w* Variable, innerhalb des Objektes, wird ein Wert behalten, der einem bestimmten Prozent der Objektlänge entspricht (hier ist 80 Prozent, aber man kann den Wert ändern, um die verschiedenen Effekte zu prüfen). Dadurch kann man den Abstand unter den Armenteilen regulieren:

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

8. Der *scale* Faktor muss steigern, so dass der im folgenden Zyklus geschaffene Teil größer ist. Man kann diesen Wert modifizieren, um Objekte mit verschiedenen Aussehen zu schaffen:

```
scale +=3;
```

9. Alle Aktionen, welche die Stelle und die Drehung der Instanz regulieren, können in eine Funktion gesammelt werden. Diese Funktion wird später in der Script-Datei erscheinen. Hier reicht, sie zu nennen, wie es folgt:

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

10. Die Teile müssen sich voneinander abhängig machen: Der erste Teil wird der Maus folgen, aber seine Stelle und Drehung werden jene von dem folgenden Teil beeinflussen. Es ist dann nützlich die *parent* Variable zu schaffen (Achtung: nicht mit *_parent* verwechseln!): diese wird in der mit der niedrigsten Nummer genannten Instanz liegen, die den Wert der aktuellen Instanz hat. Klarer gesagt: die "a0" Instanz wird die *parent* Variable erhalten, und ihr Wert wird "a1" sein. So, während der "a0" betreffenden Aktionen kann man die *parent* Variable benutzen, um sich auf den folgenden Teil, d.h. "a1", zu beziehen:

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

11. Zyklus schliessen:

```
}
```

12. Am Ende des Zyklus war "a9" der letzte Wert von *arm*. Diese Variable wird wieder benutzt (mit einer anderen Teilanzahl wäre der Instanzname anders), um die Koordinaten des größeren Teils in der Mitte des Stage einzustellen:

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

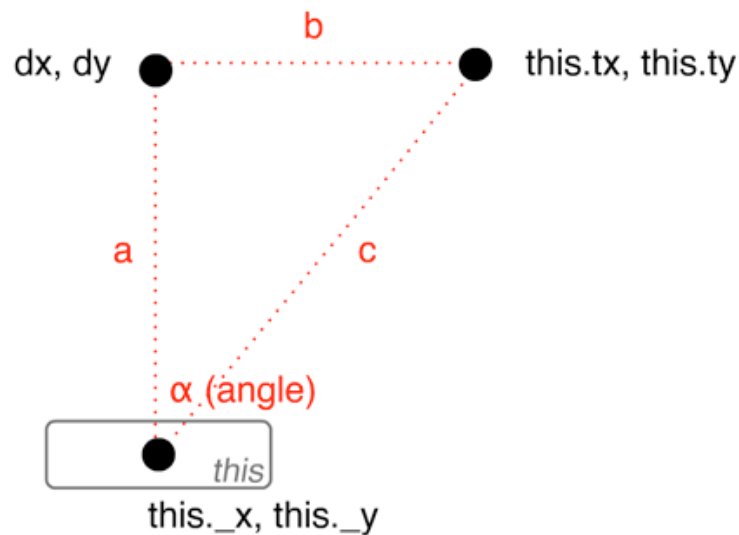
13. Jetzt kann man die *move* Funktion schreiben:

```
function move() {
```

14. Jetzt muss man den Anziehungspunkt festsetzen. Er wird von seinen x und y Koordinaten bestimmt, die in der tx, beziehungsweise ty Variable erhalten sind. Selbstverständlich muss der Anziehungspunkt anders für jeden Teil des Armes sein. Der kleinste Teil, d.h. "a0", wird von den Mauskoordinaten angezogen. Gemäß dem Winkel dieser Instanz muss der Anziehungspunkt von den folgenden Teilen, "a1" usw., festgesetzt werden.

Der Anziehungspunkt von "a0" wird am Ende der Script-Datei in einem Handler onEnterFrame bestimmt: durch den kann man die Stellungskoordinaten vom Zeiger ständig aufzeichnen. Den Anziehungspunkt vom parent Teil zu berechnen, erlaubt, dieselbe Variable mit einem neuen Wert im folgenden Zyklus zu benutzen.

Zunächst braucht man den Wert vom Inklinationswinkel zu finden, den *arm* einstellen muss, um sich zum Anziehungspunkt anzuwenden. Dazu muss man sich vorstellen, dass der Anziehungspunkt und der Aufzeichnungspunkt der Instanz ein imaginäres rechtwinkeliges Dreieck zeichnen.



15. Die Katheten von diesem Rechteck können gerechnet werden, wenn man die horizontalen, beziehungsweise die vertikalen Koordinaten vom Anziehungs- und Aufzeichnungspunkt des Objektes abzieht.

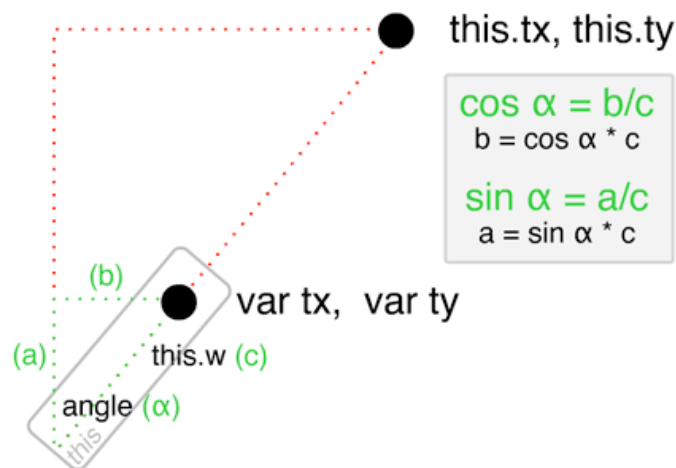
```
var dx = this.tx - this._x;  
var dy = this.ty - this._y;
```

16. Da die Werte von dx und dy bekannt sind, ist es jetzt möglich den α Winkel zu erhalten, indem man den Arkustangenswert kalkuliert:

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

17. Nehmen wir jetzt an, dass der Objekt gedreht wurde. Man möchte, dass der Anziehungspunkt für den folgenden Teil auf 80% seiner Länge liegt: aus diesem Grund wurde die `w` Variable im früheren Schritt geschaffen. Also, der Winkel und die Hypotenuse sind bekannt, aber die zwei Katheten, die nötig sind, den neuen Wert von `tx` und `ty` zu kalkulieren, kennt man nicht. Man kann die Katheten berechnen, indem man Kosinus und Sinus des Winkels mit der Hypotenuse multipliziert (`this.w`). Man muss dann den erhaltenen Wert von den vorigen `this.tx` und `this.ty` Werten abziehen, um die folgenden Punktkoordinaten zu erhalten:

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



18. Es ist jetzt möglich, die erhaltene Werte den `tx` und `ty` Variablen zuzuteilen, die in dem parent Objekt, d.h. im folgenden Teil der Kette (jenem mit der höchsten Nummer), liegen. Auf diese Weise, wenn der parent Objekt im nächsten Zyklus in `this` umgewandelt wird, werden die erhaltenen Werte benutzt, um die nächsten Werten zu kalkulieren:

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

19. Die `atan2` Methode vom `Math` Objekt gibt einen Radiantenwert zurück, während die Eigenschaft `_rotation` von einem Objekt normalerweise Grad gemessen ist. Also, da man `angle` und den Wert von `Eigenschaft_rotation` zusammen behandeln wird, ist es besser ihre Maßeinheiten zu vereinheitlichen. Dann wird `angle` Grad gemessen:

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

20. Von `targetRotation` muss man den aktuellen Drehungswinkel vom Objekt abziehen, um den Abänderungswinkel der Drehung zu erhalten, die der Objekt erfährt:

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

21. Wenn der erhaltene Wert größer als 180 ist, muss man von ihm 360 abziehen (while wird statt if benutzt, so dass man weitergehen kann, bis die Bedingung wahr ist). Wenn der erhaltene Wert kleiner als -180 ist, muss man ihn zu 360 summieren. Auf diese Weise wird man seltsamen Abläufen zuvorkommen, wenn, in dem folgenden Schritt, ein Geschwindigkeitswert zugeteilt wird:

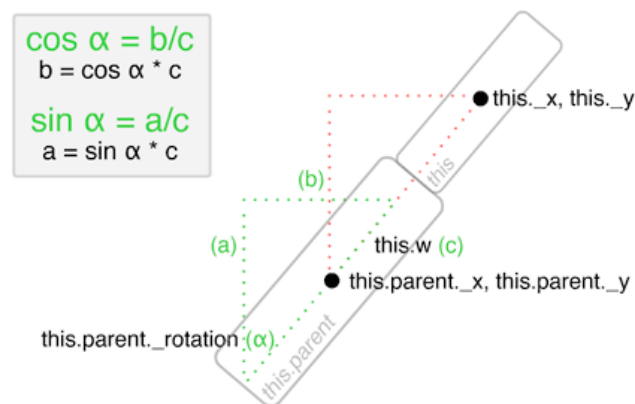
```
while (diff>180) {  
    diff -= 360;  
}  
while (diff<-180) {  
    diff += 360;  
}
```

22. Um die Bewegung weicher zu machen, ist es besser, einen Reibungsfaktor hinzuzurechnen. Das ist ein sehr kleiner Wert, mit dem der Drehungsfaktor multipliziert wird (hier ist 0.1 vorgeschlagen, d.h. das 10% vom Winkel, aber man kann versuchen, diesen Wert zu modifizieren, um verschiedene Ergebnisse zu erhalten). Indem der Winkelwert kleiner wird, wird auch seine Geschwindigkeit verringert: das macht den Objekt geschlossener. Für die Werte unter 2, wäre diese Änderung so klein, dass sie auch nicht wahrnehmbar ist: deswegen kann man sie auch nicht kalkulieren, indem man eine konditionale Struktur benutzt:

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

23. Man muss die Objektstellung auf den Koordinaten von parent berechnen. Durch dieselbe Formel, die vorher benutzt wurde, wird man die Katheten vom rechtwinkligen Dreieck auf Grund der w Länge vom parent und ihrer Drehung erhalten. Dann wird man diese Werte den x und y Koordinaten vom parent, um die neuen x und y Koordinaten vom this Objekt zu erhalten:

```
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. Die tx und ty Koordinaten vom Anziehungspunkt dem ersten Armteil entsprechend, d.h. "a0", müssen mit den Zeigerkoordinaten übereinstimmen. Wie schon gesagt, muss man die Zeigerkoordinaten ständig kontrollieren und deswegen muss man ein handler onEnterFrame eingeben:

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