

Animating Dynamic MovieClips in AS3 の日本語訳

本ドキュメントは、KIRUPA.COM のサイトにある“Animating Dynamic MovieClips in AS3 “をヒム・カンパニー 永井勝則が自主的に日本語に訳したものです。

<http://www.himco.jp>

knagai@himco.jp

(2007/10/10)

本ドキュメントの原文は

http://www.kirupa.com/developer/flashcs3/animating_dynamic_movieclips_AS3_pg1.htm

です。

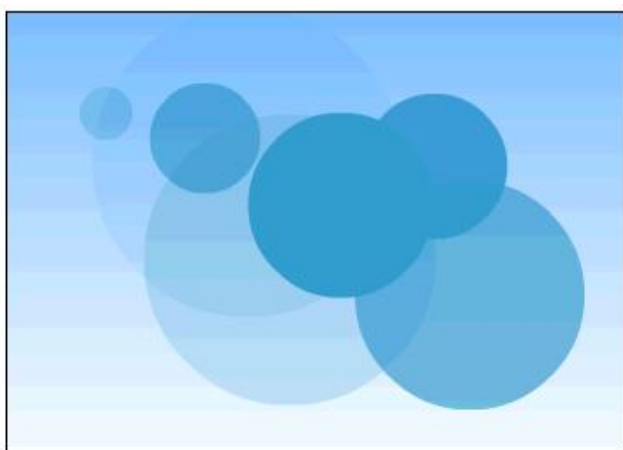
KIRUPA.COM

AS 3.0 を使ったダイナミックなムービークリップのアニメーション

page 1

Flash ではタイムラインを使ったアニメーションの作成はいたって簡単です。少しプログラミングの知識があればステージにあるムービークリップをアニメーションさせることもできます。しかしタイムラインも使わずステージに存在しないムービークリップをアニメーションさせるには工夫が要ります。これらのムービークリップはただライブラリの中に存在するだけです。このチュートリアルではコードを使ってこれらをアニメーションさせる方法を取り上げます。

たとえば以下の青い矩形をクリックすると、クリックした場所で円がフェードインしフェードアウトします。



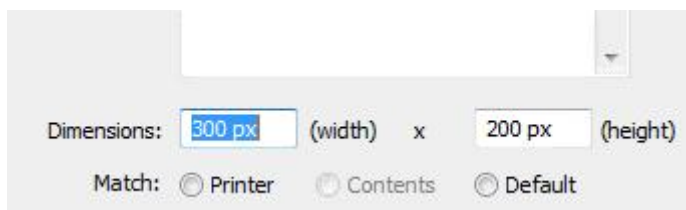
円はマウスでクリックされたときのみステージに追加されアニメーションします。このチュートリアルでは、ライブラリに保持した描画コンテンツをアニメーションさせる方法を学びます。

スタート

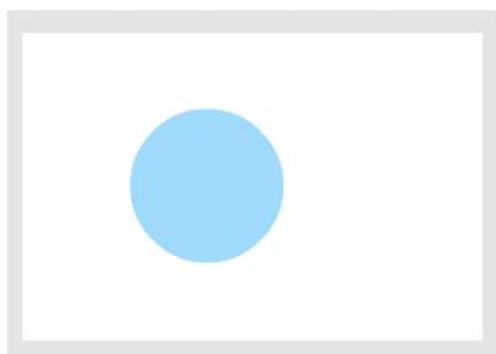
まずダイナミックにロードするムービークリップを作成します。この記事では単に単色の青い円のムービークリップを作成し、ライブラリでそのクラス名を **BlueCircle** にします。その方法についてよく分からない場合は、以下の説明が助けになります。

ライブラリに作成した円を保持し **BlueCircle** というクラス名のつけ方をお分かりの方は、コードを記述する次のページに進んでください。

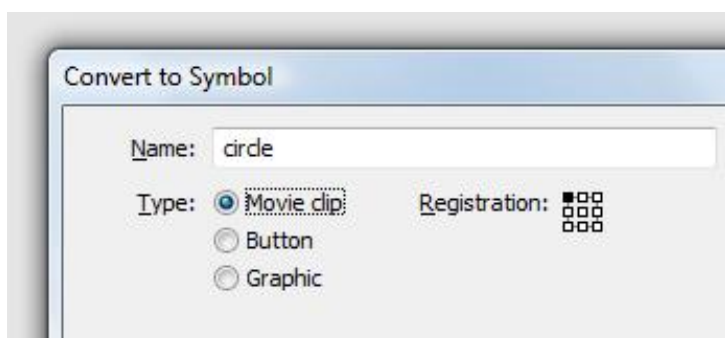
1. まず Flash CS3 で新規 FLA ファイルを作成します (AS 3.0)。プロパティインスペクタで[サイズ]の横にあるボタンをクリックし、FLA ドキュメントの幅と高さをそれぞれ 300 ピクセルと 200 ピクセルに設定します。



2. ステージの幅と高さが希望通りに設定できたので、円を描画します。[楕円]ツールで青い単色塗りの円を描きます。

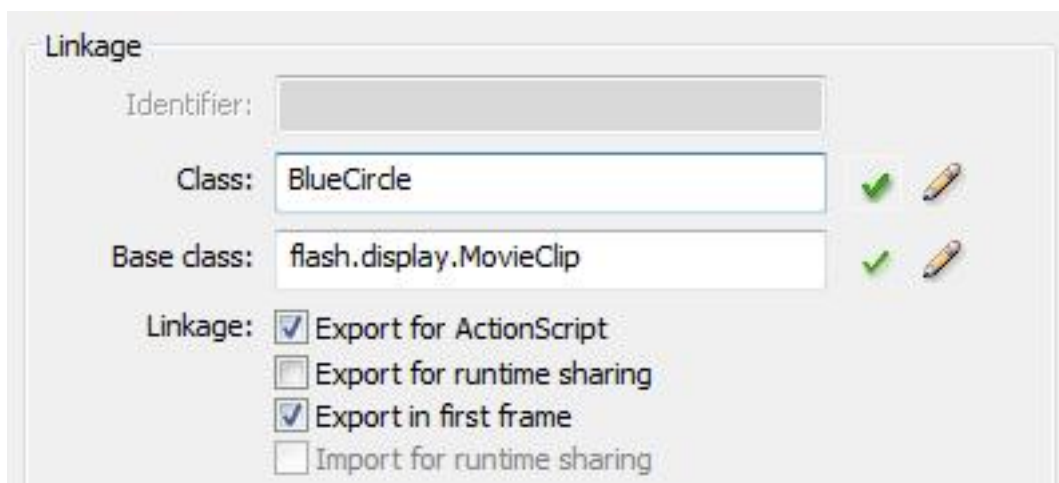


3. 円を選択し F8 を押して[シンボルに変換]ダイアログボックスを表示させます。[名前]に **circle** を入力し、[タイプ]に[ムービークリップ]オプションが選択されていることを確認します。まだ[OK]はクリックしません。もう少し変更を加えます。(下図ではムービークリップの基準点は左上で作成していますが、クリックした点から円が均等に大きくなっていくので、基準点は真ん中の方が作成する方が見栄えはよくなります)



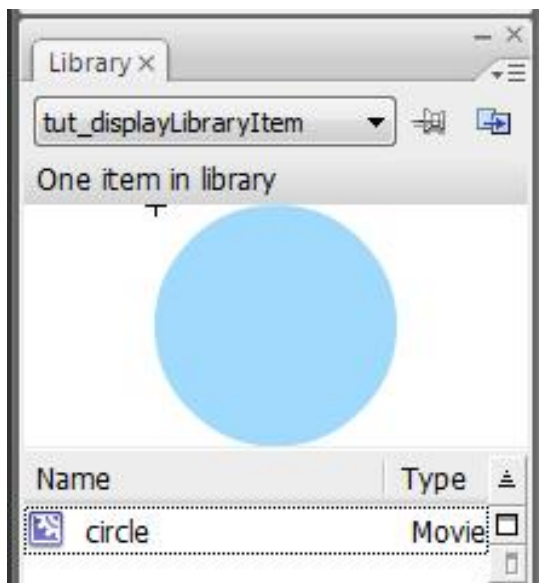
4. [シンボルに変換]ダイアログボックスで[リンケージ]領域を探します。[リンケージ]領域が表示されていない場合は、[詳細]ボタンをクリックします。[リンケージ]領域で[ActionScript に書き出し]チェックボックスを選

択します。その2つ上の[クラス]フィールドで表示されている文字(circle)を **BlueCircle** に変更します。



[基本クラス]フィールドには自動的に値が入ります。もし入っていない場合は、flash.display.MovieClip と入力します。

5. [OK]ボタンをクリックしてダイアログボックスを閉じます。[OK]をクリックすると、ライブラリに新しく作成したシンボルが表示されます。



6. ここでは作成した円のムービークリップはライブラリに保持され、その同じクリップのコピーがステージ上にあります。ステージにある円のムービークリップを選択し Delete キーを押してそれを消去します。これでステージ上に

は何もなくなりました。

これでステージは空で、ライブラリにはクラス名 `BlueCircle` という円のムービークリップが表示されています。ここまでは実際には何もやっていません。その変更は次のページでコードを記述するときに行います。

page 2

前のページでは動的にロードしアニメーションさせるムービークリップを作成しました。このページではこの両方の動作を行わせることのできるコードをお見せします。

コードの追加

タイムラインのキーフレームを右クリックし[アクション]を選択します。アクションパネルに以下のコードを記述します。

```
function Main() {
    //ステージへのマウスイベントの追加
    stage.addEventListener(MouseEvent.CLICK,AddCircle);
}
Main();

function AddCircle(e:MouseEvent):void {
    //ステージへの円の追加
    var newCircle:BlueCircle=new BlueCircle();
    this.addChild(newCircle);

    //円の x と y 位置の設定
    newCircle.x=mouseX;
    newCircle.y=mouseY;

    //円のスケールとアルファの設定
    newCircle.scaleX=0;
    newCircle.scaleY=0;

    newCircle.alpha=0;

    //ENTER_FRAME イベントリスナーの追加
```

```

newCircle.addEventListener(Event.ENTER_FRAME,ZoomCircle);
}

function ZoomCircle(e:Event):void{
    //クリックされた円の取得
    var circleMC:MovieClip=MovieClip(e.target);

    //スケールの増加
    circleMC.scaleX+=.05;
    circleMC.scaleY+=.05;

    //一定のサイズに達したら円をフェードアウトさせる
    if(circleMC.scaleX<2){
        circleMC.alpha+=.03;
    }else{
        circleMC.alpha-=.03;

        //円が(ほとんど)見えなくなったら enterFrame イベントを停止する
        if(circleMC.alpha<.1){
            circleMC.removeEventListener(Event.ENTER_FRAME,ZoomCircle);
        }
    }
}
}

```

ムービーをプレビューします。ステージのどこかをクリックすると、1ページめで見たとように円がフェードインしフェードアウトします。

コードの細部

アニメーションが動くようになったので、ダイナミックなムービークリップをアニメーションさせるときにもっとよいアイデアが浮かぶように、コードを詳しく見ていきましょう。

Main 関数から見ていきます。

```

function Main() {
    //ステージへのマウスイベントの追加

```

```
stage.addEventListener(MouseEvent.CLICK,AddCircle);  
}  
Main();
```

Main 関数はムービーが最初にスタートしたら呼び出される関数です。この関数の最も重要な(そして唯一の)コードは、マウスのクリックを監視するイベントリスナーを追加している行です。

```
stage.addEventListener(MouseEvent.CLICK,AddCircle);
```

イベントを監視するためにはまず、リスナーとして動作するターゲットを指定する必要があります。ここではムービーの stage をリスナーとして指定しています。有効なリスナーにするにはまた、何を監視すべきかを知る必要があります。このコードでは MouseEvent.CLICK イベントを監視します。

監視すべきものが分かったらおそらく何かを行いたくなるでしょう。最も簡単な方法は、その“何かを行う”コードを持った関数を作成することです。ここでは AddCircle 関数を呼び出して聞きつけたイベントを処理させています。

要点を繰り返すと、stage にマウスクリックのイベントを監視するように伝え、そのクリック(MouseEvent.CLICK)を聞きつけたら、AddCircle 関数を呼び出すのです。

イベントリスナーに関する説明は、AddCircle 関数を呼び出すところまでできました。では AddCircle 関数を見てみましょう。

```
function AddCircle(e:MouseEvent):void {  
    .  
    .  
    .  
}
```

イベントハンドラによって呼び出される関数は、ほかの関数の呼び出しと違って、ある特徴を満たす必要があります。まず Event にもとづく1つ(1つのみ)の引数をとらなくてはなりません。2つめに、値を返すことはできません。したがって戻り型は必ず void になります。

AddCircle 関数はこの基準を2つとも満たしています。MouseEvent 型の引数を1つとり(MouseEvent は Event クラスにもとづいています)、何も返しません。したがって戻り型は void に設定しています。

では AddCircle 関数全体を見てみましょう。

```
function AddCircle(e:MouseEvent):void {  
    //ステージへの円の追加  
    var newCircle:BlueCircle=new BlueCircle();  
    this.addChild(newCircle);  
  
    //円の x と y 位置の設定  
    newCircle.x=mouseX;  
    newCircle.y=mouseY;  
  
    //円のスケールとアルファの設定  
    newCircle.scaleX=0;  
    newCircle.scaleY=0;  
  
    newCircle.alpha=0;  
  
    //ENTER_FRAME イベントリスナーの追加  
    newCircle.addEventListener(Event.ENTER_FRAME,ZoomCircle);  
}
```

最初の部分では、BlueCircle クラスを追加しステージにそれを表示しています。これに関してはすべて“[AS 3.0 でのライブラリコンテンツの表示](#)”チュートリアルで取り上げているので、ここでは説明しません。ただ新しい円は newCircle で参照し、また scaleX と scaleY、alpha の初期値を 0 に設定していることに注意してください。

このチュートリアルでもっと面白い行は、newCircle ムービークリップに ENTER_FRAME イベントリスナーを割り当てているところです。

```
//ENTER_FRAME イベントリスナーの追加  
newCircle.addEventListener(Event.ENTER_FRAME,ZoomCircle);
```

イベントリスナーを追加するこの形式はもうおなじみのはずですが。AS 2 では onEnterFrame/enterFrame であったのが今は Event.ENTER_FRAME なのです。このイベントリスナーは、フレームレートで可能な速さで ZoomCircle 関数を繰り返し呼び出します。

次のページでは、ZoomCircle 関数のコードから見ていきましょう。

前のページでは、コードの説明から始めました。その中ではイベントハンドラの追加方法を学びました。このページでは、残りのコードを説明しこのチュートリアルを終えます。

```
function ZoomCircle(e:Event):void{
    //クリックされた円の取得
    var circleMC:MovieClip=MovieClip(e.target);

    //スケールの増加
    circleMC.scaleX+=.05;
    circleMC.scaleY+=.05;

    //一定のサイズに達したら円をフェードアウトさせる
    if(circleMC.scaleX<2){
        circleMC.alpha+=.03;
    }else{
        circleMC.alpha-=.03;

        //円が(ほとんど)見えなくなったら enterFrame イベントを停止する
        if(circleMC.alpha<.1){
            circleMC.removeEventListener(Event.ENTER_FRAME,ZoomCircle);
        }
    }
}
```

ZoomCircle 関数は、イベントハンドラが呼び出す関数の標準的な決まりに合っています。この関数は Event 型の引数を1つとり、戻り型は void です。関数内のコードでは円を拡大させる処理を行っています。複雑なことは何も行っていません。

ここではあっても意味がないので、前に追加したイベントハンドラを削除することでアニメーションを止めています。

```
//円が(ほとんど)見えなくなったら enterFrame イベントを停止する
if(circleMC.alpha<.1){
    circleMC.removeEventListener(Event.ENTER_FRAME,ZoomCircle);
}
```

```
}
```

イベントハンドラを削除するには、`removeEventListener` 関数を使用します。`removeEventListener` に渡す引数は、リスナーをオブジェクトに追加したときに渡した引数に一致してはいけません。言い換えると、呼び出されるイベントと関数は同じものでなくてはならないのです。

どのオブジェクトがイベントのソースであるか知ることは非常に重要です。この場合でいうと、どのムービークリップが `ZoomCircle` 関数を発射するイベントリスナーを保持しているのかを知る必要があります。それによってそのイベントリスナーを削除できます。

イベントを引き起こすオブジェクトの判定は、イベントオブジェクトの `target` メソッドによって決めることができます。

```
var circleMC:MovieClip=MovieClip(e.target);
```

Flash ヘルプより引用:

Event.target プロパティ

target:Object [read-only]

言語バージョン : ActionScript 3.0

Player のバージョン : Flash Player 9

イベントターゲットです。このプロパティにはターゲットノードが含まれます。たとえば、ユーザーが [OK] ボタンをクリックした場合、ターゲットノードはそのボタンを含む表示リストノードです。

実装

```
public function get target():Object
```

`ZoomCircle` 関数のこのコードでは、Event オブジェクトの `target` プロパティを調べることで、どのムービークリップがそのイベントを呼び出したのかを判定しています。

`e.target` は Object 型のオブジェクトを返します。ここで関心があるのはムービークリップなので、返されたオブジェクトを `MovieClip(object)` を使ってムービークリップにキャストしています。この `object` は型変換する対象です。ここでは `e.target` が返すオブジェクトを `MovieClip` に型変換しています。

まとめ

このチュートリアルでは幅広い事柄を取り上げ、イベントハンドラを使ってダイナミックなムービークリップをアニメーションさせる方法を学びました。その上、イベントハンドラの追加と削除といった方法の詳細を知り、イベントのターゲットを判定することは今後非常に役に立ちます。

ENTER_FRAME イベントを使うと滑らかなアニメーションを作成することができます。従来のループと異なり、enter frame アクションは内部的なメッセージ待ち行列システムを氾濫させません。これは、アニメーションがマウスクリックに反応するなど、ほかのことを行うことができるということです。これに対しループはその動作中アプリケーションを一時的にフリーズさせます。

オブジェクトに ENTER_FRAME イベントを割り当てるには、関数を呼び出す addEventListener を使用する必要があります。この手順では、AS 2 で学んだダイナミックにアニメーションさせる方法のプログラミングを少しやり直す必要がありますが、長い目で見れば、その新しい方法は、そのほかのプログラミング言語で見られるシンタックスやスタイルと一致することに気づかれるでしょう。