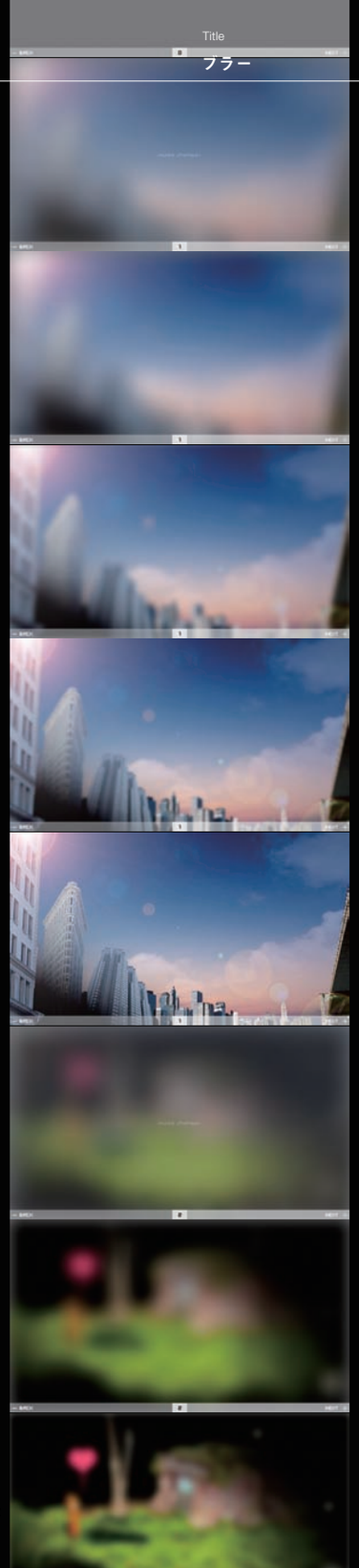


ブラー

Creator

タナカミノル（pickles）

スクリプトでブラー効果

Flash 8 からスクリプトで制御できるようになったブラー。
モーションブラーとクロスフェードをスクリプトで実現する。

Title
ブラー

Sample URL
モーションブラー：http://www.pickles.tv/dotfla/03_1/
クロスフェード：http://www.pickles.tv/dotfla/03_2/

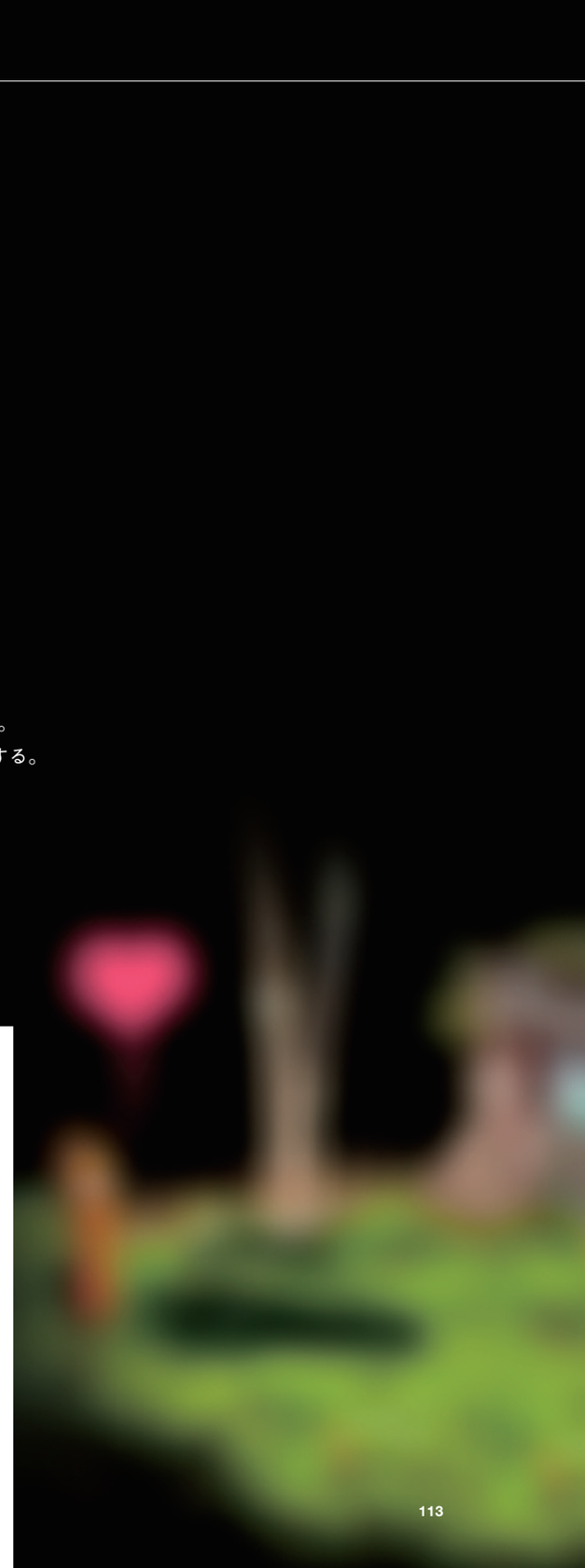
Archive
blur.zip

File
11

スクリプト
ActionScript 1.0

対応プレーヤー
Flash Player 8以上

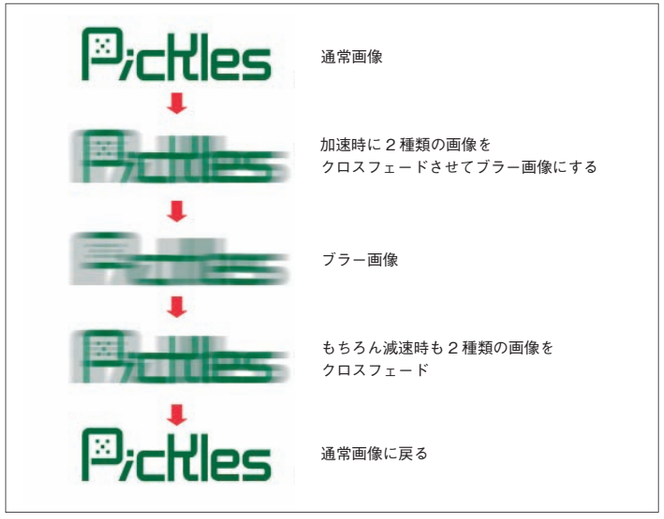
制作アプリケーション
Flash 8



発案
Flash 8 のブラー効果を実験

「ブラー」、日本語でいうとぼかしです。このブラーという表現は、アニメーションを付ける際に非常によく使います。使わなくても問題はないのですが、使ったほうがクオリティが高いので、仕事では強要されます…。そんな感じでイヤイヤやっていた表現だったので、Flash 8 が出た際に一番注目したのはブラーの効果でした。なぜイヤイヤやっていたかというと、Flash MX 2004 まではブラー画像を別に作成して、元の画像とクロスフェードさせて表現させなければならなかったからです。

従来のブラー表現（X 方向でのモーションブラーの場合）



これだと普通にアニメーションを付ける作業の 2 倍の労力がかかります。それが、Flash 8 のブラーを使えば、この作業がいらなくなる！　ということで、「こりゃエライ工数が減らせるぞ」となりました。まだ仕事では、Flash 8 を使用できる案件は少ないのですが、1 年後くらいには Flash 8 を使用する案件もスタンダードになると思います。そこで、いまのうちに流用できるスクリプトを作っておけば、ラクができるなと思い、ブラーのサンプル作りをすることにしました（「人類の進歩とは、なまけようとする弱い気持ちの上に成り立っているんだなぁ」と実感…）。

ブラーを使う表現として、代表的なものが 2 種類あります。1 つは「モーションブラー」、オブジェクトが移動するときにビュッと動く表現です。もう 1 つは「クロスフェード」、通常は α 値のみで表現するのですが、ブラーを併用するとより豊かな表現になります。将来のラクのために、この 2 種類を練習で作ってみることにしました。

スクリプト
モーションブラー

○ ブラーの組み込み

まずはモーションブラーのサンプル作りをしてみました。画面をクリックしたら、その場所にオブジェクトが移動するというよくあるものです。とりあえず、さらっとスクリプトで組んでみました。

しかし、斜めに動かすとどうしても XY 両方向にブラーがかかってしまって、移動中はタダのピンぼけになってしまいます（右図「ブラーが XY 両方向にかかる」参照）。それをどう解決するか考えました。

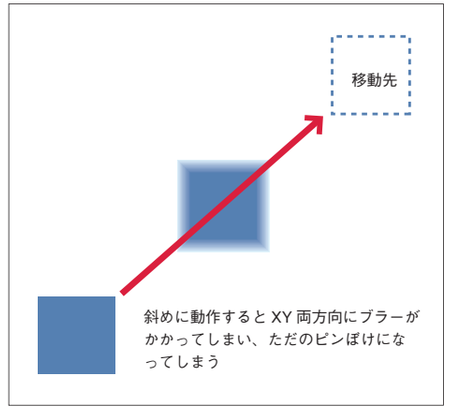
理屈としては、移動角度を求めて、その角度の分だけオブジェクトを回転して水平方向にブラーをかけ、そしてそのオブジェクトを BitMapData で画像にしてから元の角度に戻してあげればできるはず、とチャレンジしました。しかし技術がなく、なかなか実装ができません…。

そこで、この方法はあきらめて別な方法を考えました。結局「水平垂直にしかブラーがきかないように、動作を水平垂直のみにすればいいな」と思い、それで作ることにしました。

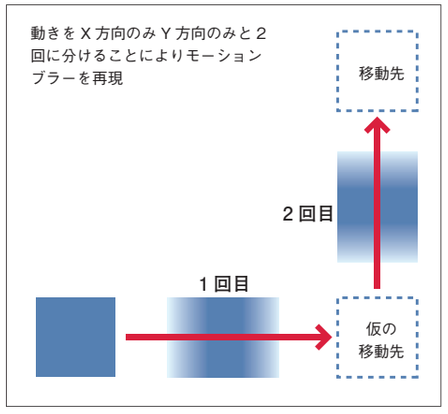
要は動作を 2 回に分ける方法です。いきなり移動先に行ってしまう斜め動作はせずに、最初は X 方向だけ動いて、X 方向が仮の移動先に近付くと Y 方向に動作が切り替わるという安直な解決法です。

流用できるスクリプト作りをしようという目標は「今まで見たことない動作だからオッケー」といういいわけによってなくなりました…。

ブラーが XY 両方向にかかる



斜めの動き



○ 水平垂直の動作をスクリプト化

オブジェクトの動作は 2 回に分けられるので、1 回目の動作が終了したら 2 回目の動作に自動で切り替われば直角に動くと考えました。

まずは、ブラーが使えるようにします。

SOURCE ブラーの組み込み

```
blur = new flash.filters.BlurFilter();// ブラーフィルターをブラーオブジェクトに設定
blur.quality = 2;// 品質を決める
blur.blurX = 0;//X 方向のブラー適用度
blur.blurY = 0;//y 方向のブラー適用度
```

動作は２回なので、クリックされた場所を２回目の移動先として設定できるようにします。

SOURCE ２回目の移動先を入れる変数を設定

```
var xpos2 = 0;//2 回目の移動先
var ypos2 = 0;
```

クリック時の場所と現在のオブジェクトを比較して、距離が大きいほうが１回目の「移動場所」と「移動方向」になります。

最後に、２回目の移動場所を設定して関数 move() にわたします。

SOURCE クリックされた場所を２回目の移動先とする

```
bg.onPress = function() {
   var mc_name = "rogo";// 移動させるオブジェクト名を代入
var xpos = 0;
var ypos = 0;
var houkou;
// クリックされた場所と現在のオブジェクトの差を求める
var x_kyori = this._xmouse-_root[mc_name]._x;
var y_kyori = this._ymouse-_root[mc_name]._y;
// それぞれを正の値にする
if (x_kyori<0) {
     x_kyori *= -1;
}
if (y_kyori<0) {
     y_kyori *= -1;
}
// それぞれを比較して 1 回目の移動場所と方向を決める
if (x_kyori>y_kyori) {
     xpos = this._xmouse;// クリックされた x 位置
     ypos = _root[mc_name]._y;// オブジェクトの y 位置
     houkou = "x";// 移動方向
} else {
     xpos = _root[mc_name]._x;
     ypos = this._ymouse;
     houkou = "y";
}
// 二回目の移動場所を代入
xpos2 = this._xmouse;
ypos2 = this._ymouse;
//move 関数に引数を送る
   move(mc_name, xpos, ypos, (Math.floor(Math.random()*120)-90)/2, 2,houkou);
};
```

次のページに続く

move() は次のとおりです。

SOURCE move()

```
function move(mc_name, xpos, ypos, rpos, speed, houkou) {
   var idou_kaisuu = 1;//1 回目移動か２回目移動を判別するフラグ
   _root[mc_name].onEnterFrame = function() {
        this._x += (xpos-this._x)/speed;//X 動作
        this._y += (ypos-this._y)/speed;//Y 動作
        this._rotation += (rpos-this._rotation)/speed;// 角度動作
        // 移動の残り量からブラーの適用度を求めて、ブラーオブジェクトに代入して適用する
        var xhoge = Math.abs(xpos-this._x);//X 残り距離
        var yhoge = Math.abs(ypos-this._y);//Y 残り距離
        var rhoge = Math.abs(rpos-this._rotation);// 残り角度
        blur.blurX = Math.floor(xhoge/speed);//X ブラー適用度
        blur.blurY = Math.floor(yhoge/speed);//Y ブラー適用度
        this.filters = [blur];// このオブジェクトにブラーオブジェクトを適用する
        if (xhoge<=5 && yhoge<=5 && idou_kaisuu == 1) {
             //1 回目の移動で残り量が 5 以下であったら
             speed *= 1.7;// 減速を大きく
             rpos *= 2;// 角度を大きく
             if (houkou == "x") {//1 回目が X 方向であったら
                  this._x = xpos;
                  ypos = ypos2;// クリック時に設定した ypos2 を代入
             } else {// それ以外は
                  this._y = ypos;
                  xpos = xpos2;// クリック時に設定した xpos2 を代入
             }
             idou_kaisuu = 2;// 移動回数フラグは 2 回目に
        } else if (xhoge<=0.5 && yhoge<=0.5 && rhoge<=0.5 && idou_kaisuu == 2) {
             //2 回目でそれぞれの係数が 0.5 以下になったら停止
             this._x = xpos;
             this._y = ypos;
             this._rotation = rpos;
             delete this.onEnterFrame;
        }
   };
}
```

この関数のポイントは、１回目か２回目かを判別して、動作を変更しているところにあります。最初に回数を設定し、その移動が終了に近付いたら、２回目の動作の係数に変更しています。１回目の if 文を見ればわかると思いますが、５ピクセルほど減速を飛ばして瞬間移動させています。完全に減速してからだと動きにおもしろさが出ないので、このような動きにしています。

サンプルファイルには最初に作ったスクリプトも入っているので、それと比較したら、どう動作しているかわかりやすいでしょう。

完成したサンプルの動き

このサンプルは流用しにくそうですが、実際は直角に物が動いたほうがよいケースもあります。たとえば、表組みされたものをセレクトする場合など、直角に動作したほうが直感的なケースが多々あると思います。そんなときに活用してみてください。

TIPS どうやってスクリプトは勉強するのか？

今回のライター陣の中で、最弱な自分がいうのはおこがましいですが、よく聞かれるのでちょびっとだけ書きたいと思います。必要は発明の母なので、誰かに「こういうの作れ!!」と命令されて、逃げ道がなければ、自然に技術は上達していくのではないかなと思っています。

そうやって尻を叩く人がいない場合は、自分から宣言するのがよいと思います。僕の場合は、できるかできないかわからないアイデアを出すようにしてます。「いった手前やらざるを得ない…」という感じで追い込むようにしたら、なんとかなるもんです。

実際の勉強は、書籍がほとんどです。よさそうなサンプルが載っていたら、買っておきます。すぐには読まず必要になったときにやっと読むという感じですが（ちなみに、さうなまんさんは 1 行でも使えそうなスクリプトがあれば、その書籍を買うそうです）。

まぁ、こんな感じでダラダラとやっているので、いまだに ActionScript1.0（以下、AS1.0）しか使えないというところがあるのですが、今のところ ActionScript2.0（以下、AS2.0）じゃないと動かないということはないので、しばらくこのままでいきそうな感じです。とはいっても、もうすぐ ActionScript3.0（以下 AS3.0）が出るのでそこで追いつきたいとか思っています（AS2.0 はすっ飛ばしちゃう予定で。AS3.0 になると完全にスクリプトが一新されるのでチャンスかと）。ここで追いついて、ギリギリ業界で生き残っていきたいかなと思います。

AS3.0 に関する情報は、上条さんのブログでの情報が充実してるので、のぞいてみることをお勧めします。

URL合 **akihiro kamijo**
http://weblogs.macromedia.com/akamijo/

スクリプト
クロスフェード

２番目に作ったのが、このサンプルです。よくあるイメージビューアですが、今まで画像で行っていたブラーでのクロスフェードをスクリプトで再現しました。こちらのサンプルは、前のサンプルと違い流用できるサンプルとなりました。

画像の切り替え処理

まずは必要画像をMCにしてリンケージさせます。今回は３枚使用しているので "image_1" ～ "image_3" という形で識別子を付けました。

初期設定は以下の形です。ここでの切り替わり速度がブラーでの初期速度になります。

SOURCE 初期設定

```
var i = 0;//ID ナンバー
var next_i = 0;// 選択されたイメージ（現在のイメージ）
var syoki_sokudo = 4;// 切り替わりの初期速度（出現）
var image_maisu = 3;// イメージの枚数
```

ボタンはイメージの枚数から動作する、しないを判別しています。ボタンが押されたら、切り替わり関数に識別子と速度を送信します。

SOURCE ボタンの動作

```
next_bottun.onPress = function() {
    if (image_maisu>next_i) {// イメージ枚数より選択イメージが少なかったら
        next_i += 1;// 選択イメージを加算
        var mc_name = "image_"+next_i;// 識別子の名前を変更して
        change_mc(mc_name, syoki_sokudo);// 切り替わり関数に送る
    }
};
back_bottun.onPress = function() {
    if (2<=next_i) {// 現在のイメージが 2 以上だったら
        next_i -= 1;// 選択イメージを減算
        var mc_name = "image_"+next_i;// 識別子の名前を変更して
        change_mc(mc_name, syoki_sokudo);// 切り替わり関数に送る
    }
};
```

切り替わりの考え方ですが、選択された画像を attachMovie() で生成し、ID から出現と消滅を判別して動作させます。前のサンプルではルートに配置したインスタンスに blur オブジェクトを適用していましたが、こちらでは生成したインスタンスに適用しています。

ポイントは、ブラーの速度変化になります。ブラー表現というのは、ピントが合っていく感じを出さないと違和感があります。実際のカメラでのピント合わせのように、最初はすばやく、ピントが合ってくるとゆっくりという感じです。

下図のように比較すればわかりやすいと思うのですが、ブラーの表現で使える適用度部分はかなり少ないです。減速を使わず一定速度で変化するようにすると、オートフォーカスもびっくりな速度でピントが合ってしまいます。ここでは、演出としてのブラーを見せなければいけないので、マニュアルフォーカスのように、最初は大きく動かして、ラストはゆっくりという表現をスクリプトで再現してます。

Flash でのぼかし適用度の比較

SOURCE ブラーの速度変化

```
function change_mc(mc_name, syoki_sokudo) {
    i++;//ID を加算して
    var new_name = "image_"+i;// インスタンス名を設定
    var syoki = {_alpha:0, i:i, sokudo:syoki_sokudo};// 初期値を設定
    _root.main.attachMovie(mc_name, new_name, i, syoki);//main の中に生成する
    // 生成された MC にブラーフィルターを適用する
    _root.main[new_name].blur = new flash.filters.BlurFilter();
    _root.main[new_name].blur.quality = 2;
    _root.main[new_name].blur.blurX = 100;// 最初はボケボケ状態に
    _root.main[new_name].blur.blurY = 100;
    _root.main[new_name].onEnterFrame = function() {
         if (_root.i == this.i) {
              // ルートの ID とこの MC の ID が同一だったら以下の出現処理
              if (this.blur.blurX<=0) {
                   this._alpha = 100;
                   this.sokudo = 2;
                   this.blur.blurX = 0;
                   this.blur.blurY = 0;
              } else {
                   // こちらの処理を先にして、上の処理にいく
                   if (this.sokudo<=0.3) {
                        this.sokudo = 0.3;// 最小速度
                   } else {
                        this.sokudo *= 0.96;// ブラーの速度を減速
                   }
                   // ブラー値とアルファ値を変更する
                   this.blur.blurX -= this.sokudo;
                   this.blur.blurY -= this.sokudo;
                   this.filters = [this.blur];
                   this._alpha += 5;
              }
         } else {
              // 別の画像が呼び出されたら以下の消滅処理
              this.blur.blurX += this.sokudo;
              this.blur.blurY += this.sokudo;
              this.filters = [this.blur];
              this._alpha -= 5;
              if (this._alpha<0) {
                   // α値が 0 未満になったらこの MC を消す
                   this.removeMovieClip();
              }
}
    };
}
```