

scan & draw

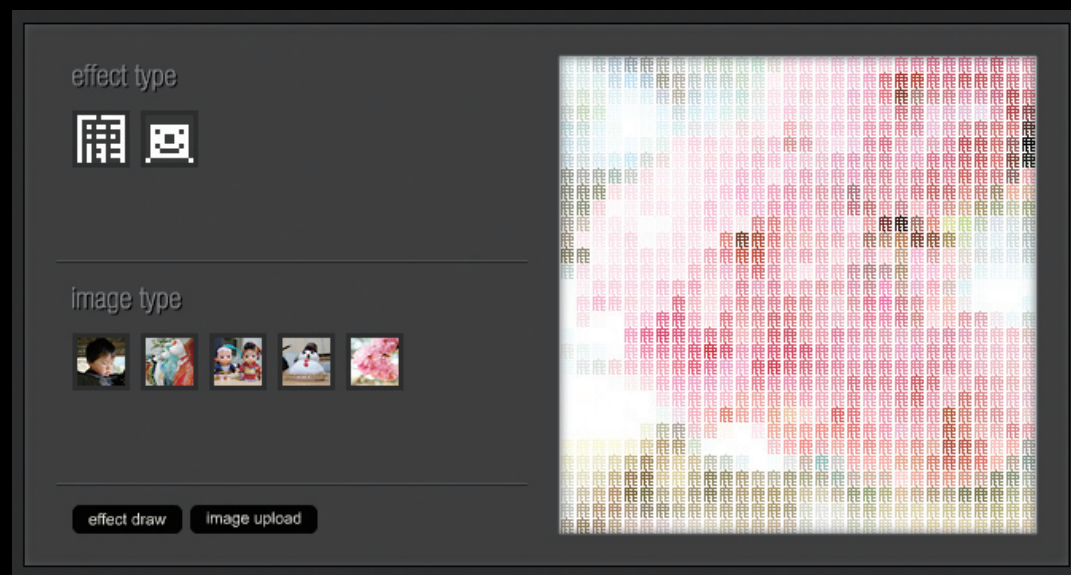
Creator

鹿倉公維 (KOUJ.JP)

ピクセル単位での写真のカスタマイズ

10倍に拡大表示した画像とドット絵のコラージュ。

同時に、Flash MX 2004 と Flash 8 の画像処理機能の違いを検証する。



Title

scan & draw

Sample URL

http://www.koui.jp/dotFla/scan_draw/

Archive

scandraw.zip

File

04

スクリプト

ActionScript 2.0

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～デザイン

写真が構成されているピクセルをカスタマイズ

このサンプルは、スキャナで何枚もの写真をスキャンしているときに思いついたアイデアです。何枚もの写真を延々とスキャンしているとき退屈になりませんか？ 私はとても退屈になります。同じような写真ならなおさらです。そんなときにふと思ったのです。同じ写真でも、違う表情が出れば退屈にはならないんじゃないか、写真に落書きをするように、写真が構成されているピクセルを四角ではなく自分でカスタムできたらおもしろいんじゃないか、そんな、退屈なときに単純すぎる発想で思い付いたのがこのサンプルの始まりです。

○ 1 ピクセルを 10 倍で表示

まず、1 ピクセルを好きなようにカスタムすることはできません（1 ピクセルなので）。そこで、30 × 30 ピクセルの画像を 10 倍の 300 × 300 ピクセルで表示することにしました。1 ピクセルが 10 ピクセルになるので、10 × 10 ピクセルの落書きスペースができるというわけです。

30 × 30 ピクセルを 10 倍で表示



○ Flash 8 と Flash MX 2004 の違い

ここでサンプルに盛り込んだ機能や実装方法を紹介していく流れになるのですが、実は、このサンプルを作ったとき、まだ Flash 8 は発売されておらず、当初は Flash MX 2004 で作成していました。今回、Flash 8 で作成し直したわけですが、この差は非常に大きなものでした。

ここでは、サンプルの機能ごとに、Flash MX 2004 での作成時と Flash 8 作成時での実装の違いについてまとめておきます。

● 画像とドット絵のコラージュ部分のドット絵が透過でなくてはならない

Flash MX 2004

- ・ loadMovie での読み込みは JPG と SWF しか読めない
- ・ JPG は透過できないので不可能
- ・ SWF を読み込む形しかないが、ドット絵は動的に生成するので「Ming」を駆使し作成

Flash 8

- ・ loadMovie で透過 PNG が読み込めるので何の問題もなし

● 画像とドット絵のコラージュ部分のドット絵を読み込む

Flash MX 2004

- ・ 10 × 10 = 100 回、同じドット絵の画像を読み込まなくてはならず、全部読み込み終わったら実行するようなフラグを作り実行させなくてはならない

Flash 8

- ・ BitmapData クラスを使ってコピーすればよいので、同じ画像を何回も読み込まなくても大丈夫

● ファイルのアップロード

Flash MX 2004

- ・ ファイルのアップロード機能はなく、HTML で別ウィンドウで作成
- ・ アップロード完了通知などは、ダミーの SWF より LocalConnection で通信しながら親ウィンドウの SWF を実行させる

Flash 8

- ・ FileReference クラスを使えば何の問題もなし

この他、今回は搭載していませんが、画像の色取得を Flash 上でできるようになっています（私は、これにより PHP に向かい合う時間が減りました）。

💡 TIPS Ming

Ming とは SWF を動的に生成するためのライブラリで、PHP や Perl など、他のプログラムで使用できます。公式サイトよりソースパッケージをダウンロードし、サーバへインストールすれば誰でも無料で使用できます。ただし、自前サーバならよいのですが、レンタルサーバの場合、モジュールをインストールできない場合があります。

[URL](http://ming.sourceforge.net/) Ming—a SWF output library and PHP module
<http://ming.sourceforge.net/>

スクリプト

ドット絵とのコラージュとファイルのアップロード

ここでは、ポイントになる部分、

- ・ draw モード
- ・ upLoad モード

の2つのスクリプトを説明していきます。

○ draw モード

draw モード、ドット絵の書き込み・削除は DrawMode.as に記述しています。

まず、初期設定で各マス目にドットが描かれているか、描かれていないかを判定する変数をセットします。

SOURCE マス目のドットを判定する変数のセット

```
//mx=10;
//my=10;
function _init(){
    for(var y:Number=0;y<my;y++){
        for(var x:Number=0;x<mx;x++){
            this["x"+x+"y"+y]=false;
        }
    }
}
```

これで、this["x"+0+"y"+0]～this["x"+9+"y"+9]までの変数がセットされます。これは、次の図のような位置の判定 (true,false) に割り当てられます。

位置の判定

x0y0	x1y0	x2y0	x3y0	x4y0	x5y0	x6y0	x7y0	x8y0	x9y0
x0y1	x1y1	x2y1	x3y1	x4y1	x5y1	x6y1	x7y1	x8y1	x9y1
x0y2	x1y2	x2y2	x3y2	x4y2	x5y2	x6y2	x7y2	x8y2	x9y2
x0y3	x1y3	x2y3	x3y3	x4y3	x5y3	x6y3	x7y3	x8y3	x9y3
x0y4	x1y4	x2y4	x3y4	x4y4	x5y4	x6y4	x7y4	x8y4	x9y4
x0y5	x1y5	x2y5	x3y5	x4y5	x5y5	x6y5	x7y5	x8y5	x9y5
x0y6	x1y6	x2y6	x3y6	x4y6	x5y6	x6y6	x7y6	x8y6	x9y6
x0y7	x1y7	x2y7	x3y7	x4y7	x5y7	x6y7	x7y7	x8y7	x9y7
x0y8	x1y8	x2y8	x3y8	x4y8	x5y8	x6y8	x7y8	x8y8	x9y8
x0y9	x1y9	x2y9	x3y9	x4y9	x5y9	x6y9	x7y9	x8y9	x9y9

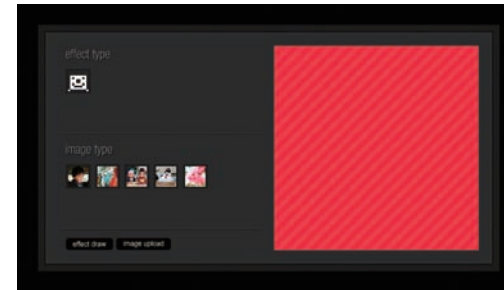
次に、クリックされたときの判定を行います。

SOURCE クリックの判定

```
function onMouseDown(){
    if(this.stage.hitTest(_root._xmouse,_root._ymouse,true)){
        if(pt){
            dotCreate();
        }
    }
}
```

マウスダウンされたとき、下図の赤で示した範囲内をクリックしている場合に dotCreate() を実行させるようにしています (pt は UP ボタンが押されたら無効になるように条件を入れています)。

dotCreate() の実行



これで初期設定とクリック判定ができましたので、実際に処理を行っている関数 dotCreate() を見ていきましょう。

まず、左図の範囲内のどの位置をクリックしているかを判定します。

SOURCE クリックの位置を判定

```
//px=30 ←マス目の大きさ
var dx:Number=Math.floor(this._xmouse/px);
var dy:Number=Math.floor(this._ymouse/px);
```

これで、

this._xmouse が 0 ～ 29.99999... の場合 [dx=0]

this._xmouse が 60 ～ 89.99999... の場合 [dx=2]

のように前ページの図の「x0y0」などを「x"+dx+"y"+dy」という形にすることができます。

これを、その位置にすでにドットが打ち込まれていた場合は削除、打ち込まれていなかった場合に表示させます。

SOURCE ドットの削除・表示

```
if(!this["x"+dx+"y"+dy]){
    this.stage.attachMovie("dot","p"+dx+"_"+dy,this.stage.getNextHighestDepth());
    this.stage["p"+dx+"_"+dy]._x=dx*px;
    this.stage["p"+dx+"_"+dy]._y=dy*px;
    this["x"+dx+"y"+dy]=true;
}else{
```

次のページに続く

```

this["x"+dx+"y"+dy]=false;
this.stage["p"+dx+"_"+dy].removeMovieClip();
}

```

このとき attachMovie() し配置する MC の名前を dx、dy に関連されているものになると、削除する際など管理しやすくなります。

次に、どのようにサーバへ送っているのかを見ていきましょう。サーバへの送信は SubmitData.as 部分です。サーバへの送信時には、

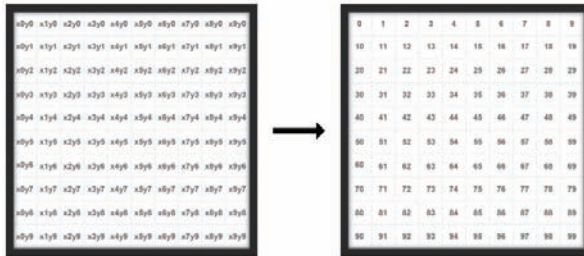
```

x0y0=true
x0y1=false
x0y2=true

```

のようなデータをカンマ区切りにし、文字列で送ります。そのため Obj 配列にまとめ、[true,false,true] という形にします。配列の長さは x0y0 ~ x9y9 までありますので、Obj は Obj[0] ~ [99] までになります。

Obj 配列に格納



SOURCE Obj 配列に格納

```

//my=10;
//mx=10;
var Obj:Array=new Array();
var i:Number=0;
var a:Number=0;
for(var y:Number=0;y<my;y++){
    a=x*y;
    for(var x:Number=0;x<mx;x++){
        i=a+x;
        Obj[i]=MC["x"+x+"y"+y];
    }
}

var d:LoadVars=new LoadVars();
d.Obj=Obj;
d.sendAndLoad("pxcreate.php",d_back,"POST");

```

これで、draw モードの一連の流れは完成です。

Flash 内の Obj[] の形に戻している イメージ

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

サーバサイドでどのような処理が行われているかという、Obj[] は Obj="true,false,true" のようにカンマ区切りの文字列で送られますので、カンマ区切りで配列に変換しています。Flash 内の Obj[] の形に戻しているイメージです。

それを true ならドットの打ち込み、false なら空白（透過）という形の条件処理をし、10 × 10 ピクセルの写真のピクセル用と 30 × 30 ピクセルのナビゲーション用の 2 つの PNG を作成しています。

10 ピクセル × 10 ピクセルと
30 ピクセル × 30 ピクセルの
2 つの PNG を作成



○ upload モード

続いて upload モードを説明していきます。

Flash 内からファイルをアップロードするのはご存知のとおり、Flash 8 から新しく追加された FileReference クラスを使います。私もまだ探っている段階なので、どんなものなのか？という感じでさわっていきました。

はじめに、image upload ボタンをクリックすると UpLoader.as 内の FileCheck() が実行されます。まず、どのファイルタイプをアップするかを設定します。

SOURCE アップするファイルタイプの設定

```

var ImageArray:Array=new Array();
var ImageType:Object=new Object();
ImageType.description="Images";
ImageType.extension="*.jpg";
ImageArray.push(ImageType);

```

ImageType.description="Images" でファイルの種類を設定し、ImageType.extension="*.jpg" でファイル拡張子を設定します。

GIF や PNG もアップできるようにする場合は

```
ImageType.extension="*.jpg;*.gif;*.png";
```

となります。

次に、処理ごとに実行される関数をリスナーに登録します。

SOURCE 実行する関数をリスナーに登録

```
var c=this;

// 画像選択後の処理
listener.onSelect=function(file:FileReference){
    file.upload("dummy.php");
    mc.gotoAndPlay("check_frame");
}

// ファイルアップロードが正常に完了したときに行う処理
listener.onComplete = function(file:FileReference){
    c.download(file.name);
}

// ダイアログボックスを閉じると行われる処理
listener.onCancel=function(file:FileReference){
    mc.gotoAndPlay(1);
}

//FileReference 設定
// リスナー登録
var myFile:FileReference=new FileReference;
myFile.addListener(listener);
myFile.browse(ImageArray);
```

画像を選択するとアップロードが開始し、アップが完了すると関数 download() が実行されます。

● 画像の縦・横のサイズを取得したい

ここまでできあがったときに気付きました。選択したファイルの名前、ファイルタイプなどは取得できるのですが、このままでは画像の縦・横のサイズを取得することができません。このサンプルでは、30 × 30 ピクセルの画像に限定しているので、画像の縦・横サイズが取得できないとまずいのです。

そこで、これは力技で解決するようにしました。まず、「dummy」フォルダに一時的に画像をアップし、関数 download() でその画像をステージ外に loadMovie し、Flash 内でその画像の縦・横幅を取得します。その画像が 30 × 30 ピクセルならば「im」フォルダに正式に格納し、そうではない場合、「dummy」フォルダに一時的に置いていた画像を削除、Flash 上でエラーを表示させます。

SOURCE dummy フォルダに画像をアップ

```
function download(file:String){
    btn_mc.createEmptyMovieClip("dummy",100);
    btn_mc.dummy.loadMovie("dummy/"+file);
    btn_mc.dummy._x=-1000;
    btn_mc.dummy._y=-1000;

    downloadProgress(btn_mc,file);
}
```

ここでステージ外の画像を読み込みます。
読み込みが完了したら、関数 checkWH() を実行させます。

SOURCE ステージ外の画像を読み込み、サイズを判定

```
function checkWH(_obj:MovieClip,file:String){
    var wMax:Number=30;// 横幅指定数
    var hMax:Number=30;// 縦幅指定数

    var w:Number=_obj._width;// 実際の横幅
    var h:Number=_obj._height;// 実際の縦幅

    var flag:String;// 判定のフラグ
    if(w==wMax && h==hMax){
        flag="yes";
    }else{
        flag="no";
    }

    if(flag=="no"){
        btn_mc.gotoAndPlay("err_frame");
    }

    _obj.removeMovieClip();// ダミーボックスを削除
    filePlay(file,flag);
}
```

ここで、30 × 30 ピクセルならば flag を "yes"、それ以外ならば flag を "no" とし、エラーアナウンスを実行させます。

最後に、関数 filePlay() で flag をサーバに送ります。

SOURCE flag を送信する

```
var v:LoadVars=new LoadVars();
v.file=file;
v.flag=flag;

v.sendAndLoad("imageup.php",v_back,"GET");
```

サーバでは、先に書いたとおり flag が "yes" の場合、「im」フォルダに正式に格納し、flag が "no" の場合、「dummy」フォルダにある画像を削除しています。これで、upload モード完成です。

今回は、アップ画像の縦・横のサイズ取得を上記のようにかなりの力技で押し切り、完成させました（もちろん、もっと他のやり方があるかもしれません）。このような問題はもちろん仕事にもあり、スケジュールがタイトな場合はなおさら、ものすごい冷や汗をかくこととなります。しかし、どうかこのような力技で押し切っています。私の場合、突然現れる壁を力技で回避できる力を身に付けるには、時間が空いたときにできるだけサンプル作りをし、いろいろなケースを体験してみることでと思っています。