my brain

Creator
鹿倉公維（KOUI.JP）

動的なテキストに「ぼかし」効果を付ける

Flash 8 から可能となった動的なテキストへのエフェクト。
ここでは、「ぼかし」効果を使用する。

Title
my brain

Sample URL
http://www.koui.jp/kouinet/time/

Archive
mybrain.zip

File
02

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～デザイン
制作中のバグから生まれたアイデア

　このサンプルのアイデアは、次に紹介する「color palette」（32P）の制作途中にスクリプト記述のミスからなるバグが発生したときに浮かんだ、何とも神降臨的なサンプルです。

スクリプト記述のミスから偶発的に発生

　その神降臨な意味合いを何かに落とし込んで、あとでスクリーンセーバーにできるようなサンプルにしよう！　というのが、このサンプル作りのはじまりです。

○ プロトタイプの作成

　まず、「color palette」とは別物に落とし込んだだけのサンプルを制作しました。

　「color palette」はコメント送信したテキストがマスの中に入り、アクティブなテキストだけがステージ内を動き回るというものですが、今回はマスを外し、アクティブなテキスト時のようにステージ内のランダムな位置に配置されていく（動き回るのも削除しました）ように作成しました。

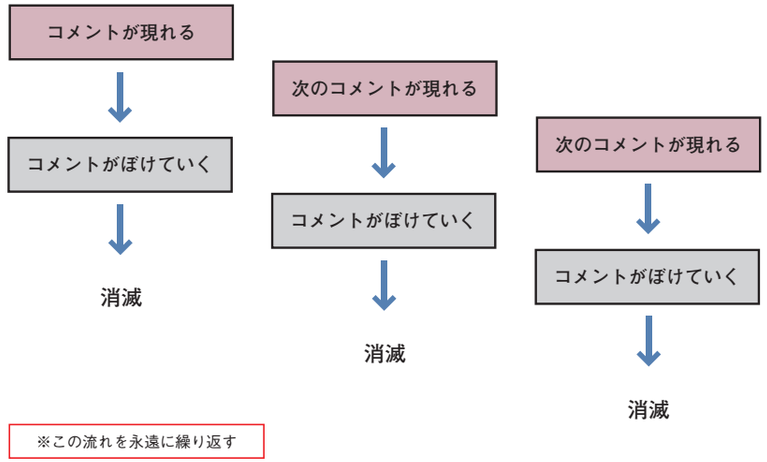
プロトタイプ

　これはこれで発色がきれいでいいじゃないか！　と思ったのですが、どうしても文字が重なるたびにどんどん文字が読めなくなっていきます。これではコメントを送信する意味がなくなってしまうので、ボツです。

○「人の記憶」のイメージを落とし込む

　次に人の記憶的なキーワードを落とし込もうと考えました。人の記憶というより、私の記憶能力といっておいたほうがよいでしょう。私の記憶の特徴は、一言で「すぐに忘れる」です。何かいわれ、次に別のことをいわれると、先にいわれたことの半分は薄れているという何ともお粗末な記憶の持ち主です。しかし、そこはポジティブ思考で、そのお粗末な記憶力のイメージをダイレクトに落とし込もうと考えました。

サンプルの構想



　これでタイトルの意味もわかってもらえたでしょうか？　そうなのです。このサンプルは、私の脳内をイメージしてできたビジュアルなのです。そのため、タイトルを「my brain」としました。

完成イメージ

スクリプト
動的なテキストにエフェクトをかける

このサンプルの全体の流れは以下のとおりです。

- XML を読み込む
- テキストが現れる
- テキストが現れるごとに画面をキャプチャし、ぼかしを付ける

もちろん、テキストはコメント送信されたものを使用します（なぜコメント送信のサンプルが多いかというと、私の性格がものすごい寂しがり屋なことが原因です…）。

このうち、XML の読み込みとコメント送信のスクリプトは「new year card」（12 ページ参照）と同じものを、またテキストが現れるスクリプトは「color palette」（32 ページ参照）と同じものを使用しますので、そちらを参照してください。

ここでは、このサンプルのコアな部分、「テキストが現れるごとに画面をキャプチャし、ぼかしを付ける」スクリプトを解説していきます。

BitmapData クラスと BlurFilter クラス

今までの Flash では、動的なテキストにぼかしなどのエフェクトを付けることはできませんでしたが、Flash 8 から搭載されている BitmapData クラスと BlurFilter クラスで実現することができるようになりました。これらのクラスを使用し、コアな実行を行っているのが bitCreate(depth) という関数です。

SOURCE BitmapData を MC に登録し、ぼかしを適応

```
_global.study.txt_stageClass.prototype.bitCreate=function(depth){

    var target=this;
    var w=Stage.width;
    var h=Stage.height;

    // ぼかし設定
    var bx=2;
    var by=2;
    var quality=1;
    var filter=new BlurFilter(bx,by,quality);

    //this.num= 作成 MC のナンバリング
    //depth= 新規 MC の深度

    this.createEmptyMovieClip("bmp"+this.num,depth);
```

次のページに続く

```
    var bit=new BitmapData(w,h,true,0x0);
    bit.draw(target);
    this["bmp"+this.num].attachBitmap(bit,this.getNextHighestDepth());

    // ぼかし適応
    bit.applyFilter(bit,bit.rectangle,new Point(0,0),filter);

    // 削除
    this.del();
}
```

関数 bitCreate() の流れは以下のとおりです。

1. 空の新規 MC を作成
2. 画面をキャプチャ
3. 空の新規 MC にキャプチャを貼り付ける
4. 貼り付けたキャプチャをぼかす
5. 1 つ前のキャプチャ MC と文字を削除

順を追って説明していきましょう。

❶ 空の新規 MC の作成

まず、空の新規 MC を作成します。

SOURCE 空の新規 MC の作成

```
this.createEmptyMovieClip("bmp"+this.num,depth);
```

this.num は新規 MC のナンバリングで、depth は MC の深度です。depth は、毎回一番上にくるように this.getNextHighestDepth() を引数に入れて送っています。

❷ 画面をキャプチャする

次に、❷ の画面をキャプチャする部分の説明です。処理を行っているのは以下のスクリプトになります。

SOURCE 画面のキャプチャリング

```
var bit=new BitmapData(w,h,true,0x0);
bit.draw(target);
```

bit という BitmapData オブジェクトを作成し、それに target(this) を描画しています。new BitmapData(w,h,true,0x0) にある w と h が

```
w=Stage.width
h=Stage.height
```

でステージの大きさにしています。

❸ 空の新規 MC にキャプチャを貼り付ける

空の新規 MC にキャプチャを貼り付ける処理は以下の部分です。

SOURCE 新規 MC にキャプチャを貼り付ける

```
this["bmp"+this.num].attachBitmap(bit,this.getNextHighestDepth());
```

これで、空の MC（this["bmp"+this.num]）に bit オブジェクトの登録が完了しました。

❹ 貼り付けたキャプチャをぼかす

貼り付けたキャプチャをぼかします。

SOURCE キャプチャをぼかす

```
bit.applyFilter(bit,bit.rectangle,new Point(0,0),filter);
```

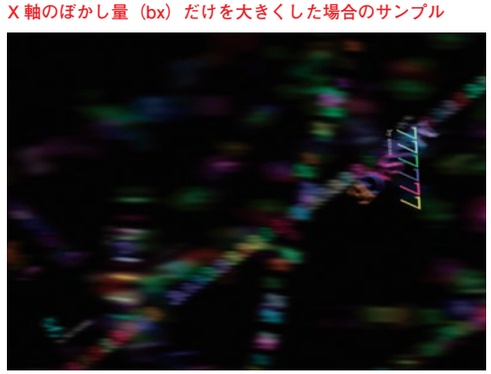
filter は先に作成しておいた BlurFilter オブジェクトです。

SOURCE BlurFilter オブジェクト

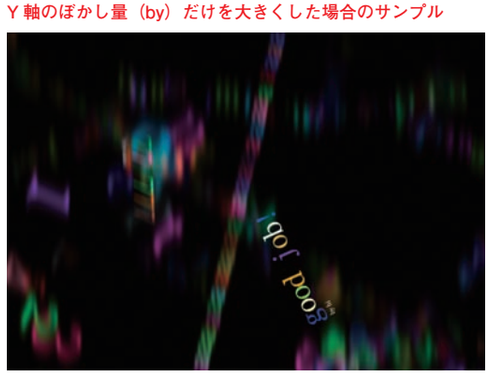
```
var bx=2;
var by=2;
var quality=1;
var filter=new BlurFilter(bx,by,quality);
```

bx,by は、それぞれ X、Y 軸に対するぼかし量（強さ）です。quality は、ぼかしの実行回数です。この数値を変更するといろいろなぼけ方ができますので、数値を変えて試してみてください。

X 軸のぼかし量（bx）だけを大きくした場合のサンプル



Y 軸のぼかし量（by）だけを大きくした場合のサンプル



❺ 1 つ前のキャプチャ MC と文字を削除する

最後に、1 つ前のキャプチャ MC と文字の処理です。これらの削除処理は、関数 del() で実行されています。関数 del() のスクリプトは以下のとおりです。

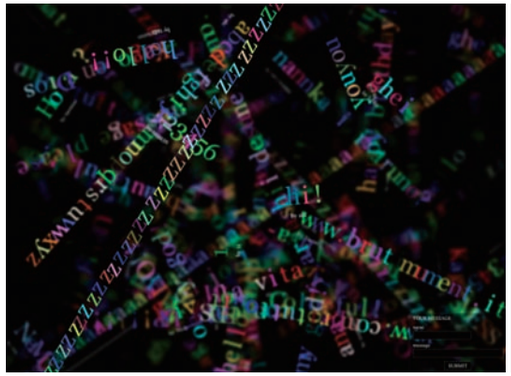
SOURCE 関数 del()

```
_global.study.txt_stageClass.prototype.del=function(){
    this["p"+(this.num-1)].removeMovieClip();
    this["bmp"+(this.num-1)].removeMovieClip();
}
```

this["p"+(this.num-1)].removeMovieClip() でテキストの塊を削除、this["bmp"+(this.num-1)].removeMovieClip() で 1 つ前のキャプチャを削除します。

これで関数 bitCreate() は完成です。この関数を 1 秒ごとに実行すれば、サンプルのような動きになります。

完成サンプル



関数の実行速度ですが、このサンプルでは 1 秒ごとにしています。実は、このサンプルのテキスト表示の実行タイミングとして考えていたのは、ユーザーがマウスダウンするごとに呼び出されるという形でした。しかしこの時点で重大なことを忘れていました。

最初に記載したように、プランニングの際、このサンプルは「後にスクリーンセーバーにできるようなサンプルにしよう！」と思っていたのです。そうであれば、ユーザーアクションがなくてもつねに実行させなくてはなりません。にもかかわらず、マウスダウンで実行させようとしていました…。それに気付き、時間とともに忘れていく自分の記憶を表現するために、1 秒ごとで実行することにしました。