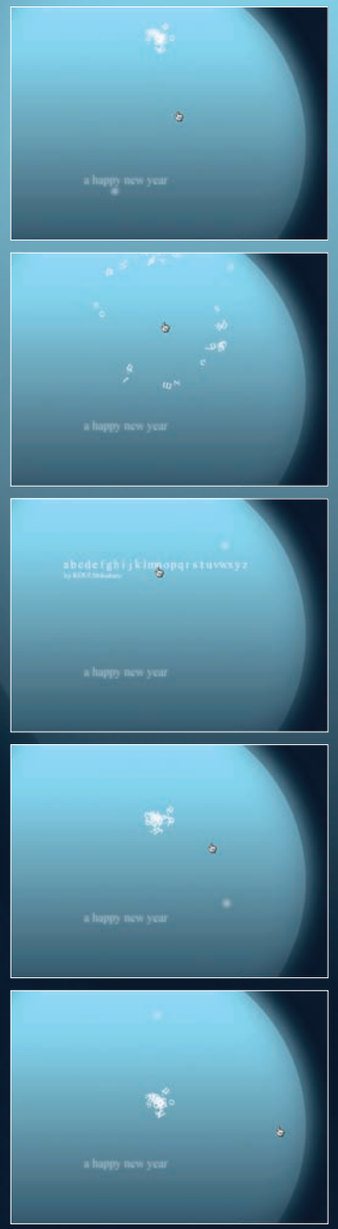


new year card

Creator
鹿倉公維（KOUI.JP）

テキストを塊にして雪のように降らせる

XMLでメッセージを受け取り、ムービー上に雪のように降らせる。
ポイントは、1文字ごとにMCを作成し位置情報をセットすること。

Title
new year card

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

Archive
newyearcard.zip

File
01

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
定番のアイテムを使わない

このサンプル「new year card」は年賀状として、クライアント、友だち関係なく送らせていただいたものです。このサンプルをデザインする際にポイントとしておいていたことは、

1. 干支を使わないで何かで表現する
2. 去年はお世話になりました。今年もよろしくお願いいたします！　というメッセージ性

の２点です。
「去年はお世話になりました。今年もよろしくお願いいたします！」というメッセージは出しつつも、数多くの年賀状の中で埋もれさせたくないというのがポイントでした。

❶ 干支を使わないで表現する

干支を表現するものは、たくさんあると思います。それを使って表現すれば、何枚も年賀状をいただく方にとっては他と同じものなわけで、それでは記憶に残らないと思いました。

暖色系・寒色系

そこで、何で年賀状でこれ？　というようなネガティブなところから記憶に残してもらおうと考え、すべてを設計していくことにしました。色味も紅・白を使う暖色系ではなく、黒・青を使った寒色系に、１月１日という初日の出感ではなく真冬感…、このように考えていき、干支に代わるものは雪しかないと思い、モチーフは雪に決定しました。

❷ メッセージ性

これは、すぐに浮かびました。「今の自分はあなたのおかげで存在しています!!」というメッセージをそのままに、ユーザーのコメントがなければ成り立たないデザインということで、コメント送信でログを残していき、それをビジュアル化していくことにしました。

テキストを重ねて雪の結晶に

❶、❷ を合わせ、雪をモチーフにテキストを表現することにしました。しかし、具体的にどうすればそうなるのでしょうか。いろいろと模索しましたが、雪の結晶の写真を見ているうちにテキストを重ねたときの雰囲気が非常に似ていると思い、そこにヘンにデザインを入れるよりもそのままでいこうと決定しました。これで全体のデザイン構想が完成しました。

スクリプト
XML 情報を雪の結晶にし降らせる

コメント送信で受け取ったテキストを解析し、「結晶化」「降らせる」などの動きをスクリプトで実現していきます。スクリプトは、大きく分けて３つあります。

- XML を解析するスクリプト
- XML 情報を雪の結晶にし降らせるスクリプト
- コメントを送信するスクリプト

ここでは、それぞれのパートごとに解説していきます。
このうち、「XML を解析するスクリプト」「コメントを送信するスクリプト」の２つは、私のサンプル４つのうち、３つで使っています。まず、この部分から説明していきましょう。

○ XML を解析するスクリプト

XML を解析するスクリプトでの処理は、以下の流れになります。

1. XML を読み込む
2. _global.xmlArray 配列を作成
3. _global.xmlArray[ ] オブジェクトを作成し、_global.xmlArray[].name、_global.xmlArray[].message プロパティを作成し、値を入れる

まず、XML を読み込む部分です。

SOURCE　XML の読み込み

```
_global.xmlLoad=function(frame){

    var xmlurl=" ＊＊＊ .xml";

    var myXML=new XML();
    myXML.ignoreWhite=true;
    myXML.onLoad=function(access){
         if(access){
              _global.xml_attri(this,frame);
         }
    }
    myXML.load(xmlurl);
}
```

引数（frame）には、処理完了後、移動するフレームラベルまたはフレーム数を入れるようにしておきます。

XML が読み込まれたら _global.xml_attri() が実行されます。_global.xml_attri() では、前述のうち、2、3 の処理を行います。

SOURCE 読み込んだ XML データを配列に格納

```
_global.xml_attri=function(_obj,frame){
    var firstObj=_obj.firstChild.childNodes;
    _global.countMax=firstObj.length;
    _global.xmlArray=[];
    for(var i=0;i<countMax;i++){
         _global.xmlArray[i]=new Object();
         _global.xmlArray[i].name=firstObj[i].attributes.name;
         _global.xmlArray[i].message=firstObj[i].attributes.message;
    }

    _root.gotoAndPlay(frame);
}
```

_global.xmlArray=[] で xmlArray という配列を作成し、name と message のプロパティをセットしています。

```
_global.xmlArray[i]=new Object();
_global.xmlArray[i].name=firstObj[i].attributes.name;
_global.xmlArray[i].message=firstObj[i].attributes.message;
```

たとえば、次のような XML データを読み込んだとします。

SOURCE XML データの例

```
<?xml version="1.0" encoding="UTF-8" ?>
<flashXML>
   <node name="KS" message="OK!" />
   <node name="KOUI.Shikakura" message="Happy new year!!" />
</flashXML>
```

name と message には、下記の値が入ります。

SOURCE name と message に値を格納

```
_global.xmlArray[0].name="KS";
_global.xmlArray[0].message="OK!";

_global.xmlArray[1].name="KOUI.Shikakura";
_global.xmlArray[1].message="Happy new year!!";
```

これで、XML データを読み込み、_global.xmlArray.name プロパティ、_global.xmlArray.message プロパティに値を格納するという処理は完了です。

○ コメントを送信するスクリプト

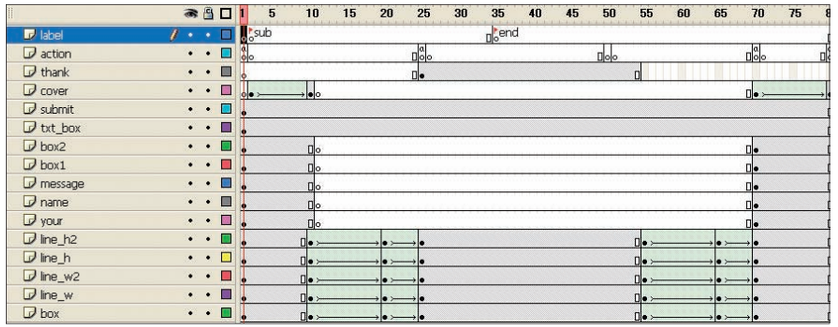
次に、コメントを送信するスクリプト部分です。このサンプルでは、post_box のフレームに記述しています。post_box には、コメント入力用のテキストフィールドを配置しています。まず、post_box 内のタイムラインを見てください。

通常は 1 フレーム目で停止しており、コメントが送信されると sub へ行き、送信が完了したら end へ進みます。

SUBMIT ボタンをクリックすると関数 submit() が呼び出されます。submit() はまず、テキストフィールド name_txt と message_txt に文字が入っているかをチェックします。

message_txt に記入がない場合、クリックは無効になります。message_txt に記入がある場合、name_txt をチェックし、name_txt に記入がなければ "NO NAME" を代わりに値として入れ、postData() でサーバ送信を実行するとともにフレームが sub へ進みます。

post_box のタイムライン



SOURCE submit() でテキストフィールドの文字入力をチェックし、値を取得

```
//SUBMIT ボタンが押された (onRelease) 時の処理
_global.study.postBoxClass.prototype.submit=function(){

    var myName=this.name_txt.text;
    var myMessage=this.message_txt.text;

    if(myMessage!=""){
         if(myName==""){
              myName="NO NAME";
         }
         this.name_txt._visible=false;
         this.message_txt._visible=false;
         this.btn._visible=false;
         this.gotoAndPlay(this.submitFrame);
         this.postData(myName,myMessage);
    }
}
```

postData() では、myName、myMessage を loadVar にセット、loadVar をサーバに送信します。返り値が戻ってきたら、post_box のフレームを end に進め、XML をリロードし完了です。

SOURCE postData() で値を loadVar にセットし、サーバに送信

```
_global.study.postBoxClass.prototype.postData=function(v1,v2){

    var c=this;

    var loadVar=new LoadVars();
    loadVar.myName=v1;
    loadVar.myMessage=v2;


    //返り値
    var loadVar_back=new LoadVars();
    loadVar_back.onLoad=function(access){
         if(access){
              c.gotoAndPlay(c.endFrame);
              _global.xmlLoad("restart");
         }
    }

    loadVar.sendAndLoad("＊＊＊.php",loadVar_back,"POST");

}

Object.registerClass("post_box",_global.study.postBoxClass);
```

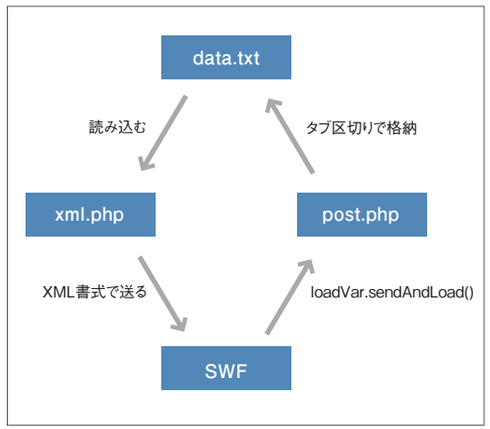
ダウンロードしたアーカイブを参照していただけるとわかるように、今回のサンプルでは、loadVar の送り先（＊＊＊.php）は post.php、読み込む XML（＊＊＊.xml）は xml.php になっています。

post.php に送られたデータは XML に書き込まれるのではなく、data.txt にタブ区切りで格納されている状態になっています。それを xml.php で XML に変換し、Flash 内に読み込ませています。

なぜ XML にしないで data.txt に格納するのか？これは、同じリソースを別の XML 書式で使用したいと思った際、タブ区切りやカンマ区切りで格納していると容易に変更できるからです。

汎用性を持たせるために、私はよくこのような形で作成しています。

post.php に送られたデータを data.txt に格納し、xml.php で XML に変換



○ XML 情報を雪の結晶にして降らせるスクリプト

いよいよ、このサンプルのメインである「XML 情報を雪の結晶にして降らせるスクリプト」の部分です。このスクリプトのポイントは

マウスオーバーするとはじけた動きをした後、文字列の並びになる

というところです。

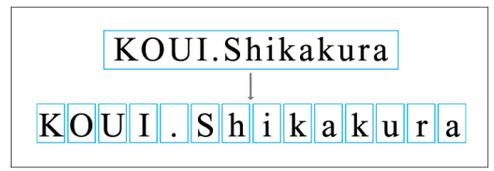
コメント送信機能を付けるからには、コメントを読ませるようにする必要があります。結晶から文字列に戻す際のきっかけとして「はじけてから文字列の並びになる」というのを考えました。

この動きを考えるとき、自分の小学校時代を思い出していました。東北育ちの私はよく冬に雪投げをして遊んでいました。小さい頃好きな子にちょっかいをかけたがる男の子がいませんでしたか？　私は、まさにそんな男の子でした。そして雪合戦、その女の子に集中して雪を投げているとその女の子の顔面に直撃、彼女は大泣き、みんなから責められる…、今でも覚えています。女の子の顔面に当たったときの雪のはじけっぷり、その記憶の中の動きを再現してみました。

● 1 文字ごとに MC にし位置情報をセット

まず、文字を複製する際、1 つの文章を 1 つのムービークリップ（以下、MC）で作成するのではなく、1 文字ごとに MC を作成していきます。それにより、「はじけてから文字の並びになる」という動きを制御します。

1 文字ごとに MC を作成



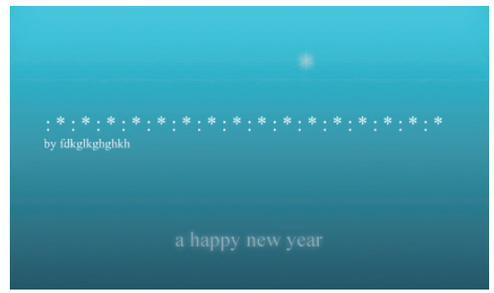
その 1 文字ごとの MC に、はじめに下記の情報をセットします。

1. dx, dy：雪として落下しているときの位置
2. maxX, maxY：はじかれたときの位置
3. acX, acY：文字の並びになったときの位置

dx, dy：雪として落下しているときの位置

maxX, maxY：はじかれたときの位置

acX, acY：文字の並びになったときの位置



雪として落下しているときの位置（dx, dy）は、次のようにセットします。

SOURCE dx,dy の位置

```
radian=Math.random()*360;
this.dx=this._x=Math.cos(radian*(Math.PI/180))*(Math.random()*r);
this.dy=this._y=Math.sin(radian*(Math.PI/180))*(Math.random()*r);
```

これにより、半径 r（今回の場合、22.5）の円の中にランダムに配置されます（下図左）。
はじかれたときの位置（maxX, maxY）は、次のようにセットしています。

SOURCE maxX, maxY の位置

```
var mr=120;

var mx=this._x-centerX;
var my=this._y-centerY;
var atan=Math.atan2(my,mx);

this.maxX=(Math.cos(atan)*mr)+this._x;
this.maxY=(Math.sin(atan)*mr)+this._y;
```

atan で MC の位置と MC の中心からなる角度を取得、その角度の半径 mr（120）の位置を取得し、現状の位置を足すことで、雪として落下しているときの位置（dx, dy）から atan の角度で半径 mr の円周上に移動する感じになります（下図右）。

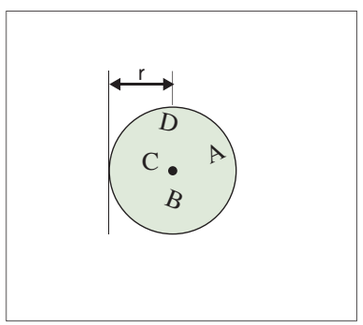
文字の並びになったときの位置（acX, acY）は、文字の個数を取得し、「文字の個数× 1 文字の大きさ」を計算し、全体の横幅を取得後、中心にくるように、全体の横幅 /2 を引いています。要は、

```
var myCenter=（文字の個数× 1 文字の大きさ）/2
this.acX=（その文字の位置× 1 文字の大きさ）－ myCenter
```

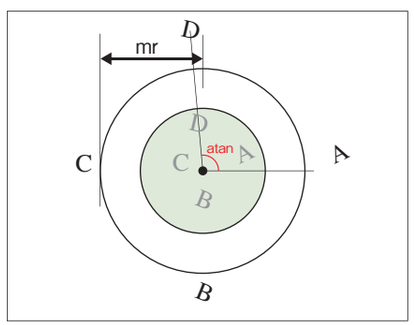
となります。

これらを文字数だけ繰り返すので、実際の式は以下のようになります。

半径 r の円内にランダムに配置



dx, dy の位置から atan の角度で半径 mr の円周上に移動する

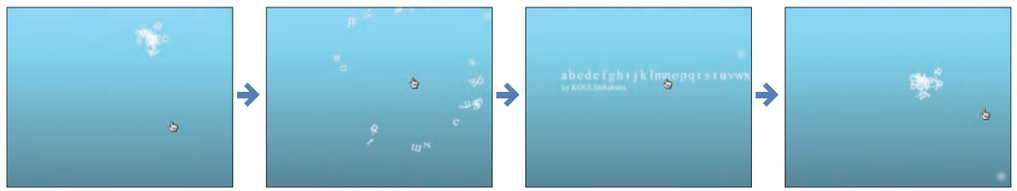


SOURCE acX, acY の位置

```
//_obj= メッセージの文字
this.m=_obj.length;// 文字の個数
//this.txtSize=1 文字の大きさ
this.txtW=this.m*this.txtSize;// 全体の横幅

for(var i=0;i<this.m;i++){
    //dx,dy
    //this.r= 半径
    this["txt"+i].dx=this["txt"+i]._x=Math.cos(radian*(Math.PI/180))*(Math.random()*this.r);
    this["txt"+i].dy=this["txt"+i]._y=Math.sin(radian*(Math.PI/180))*(Math.random()*this.r);

    //maxX,maxY
    var mx=this["txt"+i]._x-this.centerX;
    var my=this["txt"+i]._y-this.centerY;
    var atan=Math.atan2(my,mx);

    this["txt"+i].maxX=(Math.cos(atan)*mr)+this["txt"+i]._x;
    this["txt"+i].maxY=(Math.sin(atan)*mr)+this["txt"+i]._y;

    //acX,acY
    var myCenter=this.txtW/2;
    this["txt"+i].acX=(i*this.txtSize)-myCenter;
    this["txt"+i].acY=0;
}
```

これらの位置（dx, dy、maxX, maxY、acX, acY）を各動作時に呼び出せば、サンプルのような動きが完成です。この一連のスクリプトは、_root レイヤーの Obj の関数 txtCreate() 内に記述しています。

雪として落下し、はじかれて文字の並びになり、また雪に戻る一連の動き



● 文字 MC の呼び出し

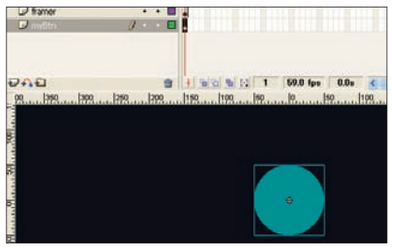
では、次にどのように 1 文字ずつの MC を呼び出しているかを見ていきます。

ObjMC の中には、「mybtn」というボタンインスタンスが含まれています。

この mybtn を

ロールオーバーすると「文字がはじかれ、文字の並び」に

mybtn インスタンス



ロールアウトすると「雪として落下している状態に戻る」に

なります。

SOURCE mybtn のスクリプト

```
//_obj=this.mybtn
var c=this;
_obj.onRollOver=function(){
    c.Preview();
}
_obj.onRollOut=_obj.onReleaseOutside=function(){
    c.rePreview();
}
```

文字がはじかれ、また文字の並びになる動き

「はじかれ（maxX, maxY）、一定の時間を過ぎると文字の並びになる（acX, acY）」動きは Preview() で行います。Preview() は、_root レイヤーの Obj に記述しています。

はじめにタイムスタンプとある一定の時間をセットします。

SOURCE タイムスタンプとある一定の時間のセット

```
var _num=0;// タイムスタンプ
var _flag=25;// ある一定の時間
```

次に、タイムスタンプ（_num）とある一定の時間（_flag）を比較し、_num<_flag のときには「はじかれる（maxX, maxY）、それ以上のときは acX, acY」になるようにします。

SOURCE 「はじかれ、一定時間を過ぎると文字の並びになる」動き（1 文字分）

```
//MC=1 文字の MC
onEnterFrame=function(){
    if(_num<_flag){
        diff=0.2;
        xPos=MC.maxX;
        yPos=MC.maxY;
    }else{
        diff=0.3;
        xPos=MC.acX;
        yPos=MC.acY;
    }

    MC._x+=(xPos-MC._x)*diff;
    MC._y+=(yPos-MC._y)*diff;

    _num++;
}
```

実際は、上記のスクリプトを文字数分繰り返すので次のとおりになります。

SOURCE 「はじかれ、一定時間を過ぎると文字の並びになる」動き（文字数分繰り返す）

```
//this.m= 文字の長さ
onEnterFrame=function(){
    for(var i=0;i<this.m;i++){
        if(_num<_flag){
xPos=this["txt"+i].maxX;
            yPos=this["txt"+i].maxY;
        }else{
            xPos=this["txt"+i].acX;
            yPos=this["txt"+i].acY;
        }

        this["txt"+i]._x+=(xPos-this["txt"+i]._x)*diff;
        this["txt"+i]._y+=(yPos-this["txt"+i]._y)*diff;

        _num++;
    }
}
```

雪として落下している状態に戻る動き

こちらは、雪の位置（dx, dy）に現状の位置から戻る処理を実行しています。これも _root レイヤーの Obj に記述しています。

SOURCE 「雪の位置に戻る」動き（1 文字分）

```
//MC=1 文字の MC
var diff=0.2;
onEnterFrame=function(){
    MC._x+=(MC.dx-MC._x)*diff;
    MC._y+=(MC.dy-MC._y)*diff;
}
```

こちらも、実際はスクリプトを文字数分繰り返すので以下のとおりになります。

SOURCE 「雪の位置に戻る」動き（文字数分繰り返す）

```
//this.m= 文字の長さ
var diff=0.2;
onEnterFrame=function(){
    for(var i=0;i<this.m;i++){
        this["txt"+i]._x+=(this["txt"+i].dx-this["txt"+i]._x)*diff;
        this["txt"+i]._y+=(this["txt"+i].dy-this["txt"+i]._y)*diff;
    }
}
```

これで、「new year card」、別名、好きな女の子に雪顔面直撃コンテンツの完成です。