尚美学園大学芸術情報学部紀要　第８号

FLASHにおける分散インスタンス制御方式

四方　義昭、野地　朱真

Control methods for scattered instances in Flash movies

SHIKATA Yoshiaki and NOJI Suma

Abstract

Control methods for movie-clip instances in FLASH movies, which move independently and are controlled by outside events such as mouth click, key down and timer explosion, are presented. The distributed control where the movie-clip action is prepared for each instance, and the stage division concentrated control where the movie-clip action for all instances are preprared in each control stage such as moving independently, gathering to one point and scattering to different direction are considered. In this paper implementation results of these two methods are also shown.

Key Word: movie-clip instances, FLASH Movie, outside events, distributed control, stage division concentrated control.

[要約]

多数のムービークリップインスタンスの動作を独立して制御し、かつ外部イベント（マウスクリック、キー押下、タイムアウト処理）を検出して全インスタンスが統一した処理を行うよう制御するためのFLASH Actionscript構成法とその特徴について述べる。

スクリプト構成法として、①全てのインスタンス動作をムービークリップアクションとして各インスタンスに分散して記述する独立分散制御方式と、②キャラクタの動きを動作ステージ毎に分割し、各ステージにおいては全インスタンス分をまとめてフレームアクションとして記述するステージ分割集中制御方式を取り上げ、これら２方式の実現例を示す。また、実際のスクリプト作成結果およびムービーの動作結果から、2方式のメリット、デメリットを評価することにより各々の適用領域を明らかにする。

キーワード：ムービークリップインスタンス、スクリプト、外部イベント、独立分散制御、ステージ分割集中制御

１．まえがき

Flashは優れたデザインを持つインタラクティブなWebページを作成するためのコンテンツ作成ソフトウェアとして広く使用されている[(1)]。また、最近はWebサーバとの連携によ

るリッチクライアント型Webコミュニケーションシステムを開発するためのツールとしても使用されている。さらに、Flash がインプリメントしているActionscriptを使用することにより、より複雑な動きをするアニメーションやゲームに代表される高度なインタラクティブ性を実現することが可能となった。

本稿では、多数（数十）の動き回るインスタンスを制御するための複数のActionscript構成法を実際の作成例を元に明らかにする。Actionscript構成法として、①個々のインスタンス動作の全てを一括して1つのスクリプト内に記述することにより、動作の流れをわかりやすく記述することを狙いとした独立分散制御方式と、②インスタンスの動作ステージ（独立して動き回る、一箇所に集合する、散らばる等）毎にスクリプトを分割して記述することによって、個々のスクリプトを単純化することを狙いとしたステージ分割集中制御方式を取り上げる。ステージ分割集中制御方式によれば、インスタンス（ムービークリップ）そのものをステージ毎に入れ替えることによって、より表現力豊かなFLASHムービーを実現することも可能となる。また、これらの作成結果の比較・評価を行うことにより、それぞれのスクリプト構成法のメリット、デメリットと適用領域を明らかにする。

２．Actionscriptの概要

ActionscriptはFLASHでインプリメントされているスクリプト言語であり、Webクライアント（ブラウザ）側で使用されているスクリプト言語であるJavascript等と似たプログラム構造を持っている。Actionscriptの記述方法は以下のとおりである[(2)]。

①FLASHのフレームに記述し、アニメーション全体の動きや時間の経過とともに変化するインスタンスの動きを制御する（フレームアクション）

②ムービークリップインスタンスに記述してそのインスタンスの動きを直接制御したり、インスタンスとの対話を実現する（ムービークリップアクション）。

③ボタンインスタンスに記述し、ボタン動作によって引き起こされる動作を制御する（ボタンアクション）。

３．実現するアニメーション

作成するアニメーションの概要は以下のとおりである。

①数十のキャラクタ（昆虫等、図１）が各々独立して一定方向に移動し、境界線（虫箱の端）に達すると進行方向を反転する（図２）。

②キャラクタが衝突すると、お互いに避けあって逃げるように移動する（図２）。

③マウスを押すと（餌を置くと）全キャラクタがマウスの位置に集合する（図３）。

④集まった状態でキーを押す（キャラクタを脅かす）と全キャラクタがびっくりしてあわてて逃げる（図４）。

⑤④の状態から一定時間が経過すると①の状態に戻る。

図１　動き回るキャラクタ

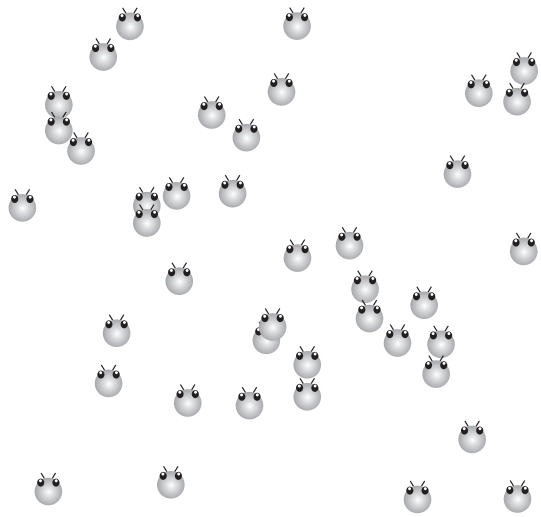

図２　動き回る様子

図３　集まる様子

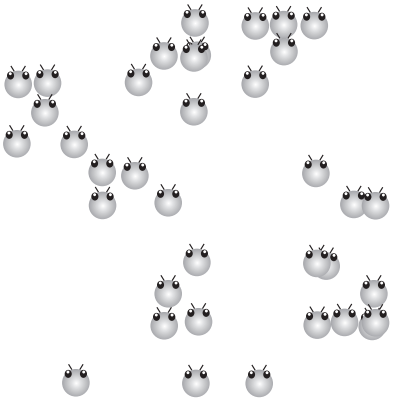

図４　散らばる様子

４．Actionscript構成法

３で示したアニメーションを実現するためのActionscript構成法として、

①各々のキャラクタ（インスタンス）の独立した動作を全てムービークリップアクションとして記述し、外部動作の監視とキャラクタの動作を制御する制御フラグの設定のみをフレームアクションとして記述する（独立分散制御方式）。

②3項の①―④に示した動作ステージ毎に独立したスクリプト構成とし、全てのキャラクタの動きを動作ステージ毎にまとめてフレームアクションとして記述する（ステージ分割集中制御方式）。

が考えられる。

構成①の狙いは、個々のインスタンス動作の全てを一括して1つのスクリプト内に記述することにより、動作の全貌をわかりやすく記述することにある。また、構成②の狙いは、インスタンスの動作ステージ（独立して動き回る、一箇所に集合する、散らばる等）毎にスクリプトを分けて記述することにより、個々のスクリプトを単純化するとともに、インスタンス（ムービークリップ）そのものをステージ毎に入れ替えることによって、より表現力豊かなFLASHムービーを実現することを可能とすることにある。

４.１　独立分散制御方式

４.１.１.　レイヤおよびフレーム構成

キャラクタを配置する「キャラクタレイヤ」と、インスタンスの複製や外部動作の監視を行う「インスタンス共通制御レイヤ」で構成する。またスクリプトの全てをムービークリップアクションとして記述するため、フレーム数は1でよい。

４.１.２.　インスタンスの独立動作

以下の構成を持つスクリプトを全インスタンス（キャラクタ）に実装する。

（１）インスタンスの初期設定（onClipEvent (load)**）**

onClipEvent (load)には、そのインスタンスが初めてタイムラインに登場した時のイベントを記述する。

x軸方向（正負）、ｙ軸方向（正負）のフレーム毎の移動距離をインスタンス毎にランダムに計算する。

（3項①と④のフレーム毎移動距離の記述例）

```
moveY = Math.floor(Math.random()*7)-3;   // 遅い移動
moveXs = Math.floor(Math.random()*29)-14;   // 早い移動
```

（２）動き回る動作（onClipEvent (enterFrame)**）**

onClipEvent (enterFrame)には、そのインスタンスのフレームが再生される度に動作するイベントを記述する。

(a) **制御方式**

3項①、③、④の各ステージ動作を「ルートフラグ」により制御する。即ち、インスタンス共通制御レイヤに記述したマウスダウン検出機能またはキー押下検出機能（**４.１.３**節参照）により「ルートフラグ」を設定し、このフラグによってムービークリップアクション処理を以下の(b)－(d)に分岐する。

(b) **直線移動**

基本的にはフレーム毎にインスタンスのｘ、ｙ座標に**（１）**で求めた移動距離を加算し、境界位置に達すると進行方向を反転するのみである。

（記述例）

```
_x += moveX;
if (_x <= 50  or _x >= 500) {
                    moveX = -moveX;
                   }
```

(c) **衝突検出と逃避**

キャラクタ同志が衝突すると、互いに相手を避けて逃げていくスクリプトの記述例は以下のとおりである。

```
for (i = 10; i <=30; i++) {   // 全て（21個）のインスタンスについて繰返し
```

```
        // インスタンス番号は2桁（10から）
        inname = this._name;    // 自分のインスタンス名を取得
        if (i != parseInt(inname.substr(5,2))){ // 自分以外のインスタンスであれば
          if (this.hitTest(_root["chara"+ i ])) {    // 衝突したら逃避
                moveX = -moveX;
                 _x += Math.floor(Math.random()*9)-4+moveX; //ｙ軸方向も同様
          }
        }
    }
```

(d)マウスを追いかける

インスタンス共通制御レイヤに記述した外部イベント検出機能でマウスダウンを検出した際にマウス座標を記憶し、その座標に向かってキャラクタが一定の速度で移動するよう制御する（例えば、インスタンスのＸ座標＜マウスのＸ座標 → _x += 3;）。

（3）タイマ制御

インスタンス共通制御レイヤにおいて、外部イベント検出機能でキー押下を検出した際に _root.startTime に時間をセーブし、ムービークリップアクションの中で経過時間を測定し、規程時間になれば状態を遷移させる。

```
nowTime = getTimer();
passTime = nowTime - _root.startTime;
if (passTime >= 2000) {
  _root.flaga = 0 ;
 }
```

インスタンス共通制御レイヤでは、ムービー起動時と外部イベント発生時にのみソフト処理が行われるため、連続した時間のチェックが困難である。そこで、フレーム（タイムライン）処理が連続するムービークリップアクション内で連続時間を監視する必要がある。また、全てのムービークリップアクションでこの処理が実行されるが、競合等の不具合は発生しない。なお、本タイムアウト処理は共通制御レイヤを複数フレーム化し、その中に記述することも可能と考えられる。

４.１.３． 共通制御

（1）インスタンスの複製

ムービークリップイベントを記述したインスタンスを複製するために duplicateMovieClip ( ) メソッドを用いてインスタンスを複製し、ランダムに配置する。記述例は以下のとおりである。

```
for (i = 11; i <=30; i++) {
        newName = "chara" +i ;
```

```
        new_mc  = chara10.duplicateMovieClip (newName, i);
        new_mc._x  = Math.floor (Math.random( ) *500);
        new_mc._y  = Math.floor (Math.random( ) *500);
}
```

（2）外部監視とルートフラグの設定

マウスがクリックされたこと、またはキー（どれでもOK）が押下されたことを検出し（イベントリスナーを用いる）、ムービークリップのアクションを制御するためのルートフラグを設定する。記述例を以下に示す。

```
flaga=0;
myListener = new Object( );
myListener.onMousedown = function ( ) {
   flaga = 1;
   xmou = _root._xmouse;   // マウスのｘ座標をセーブ
   ymou = _root._ymouse;   // マウスのｙ座標をセーブ
};
Mouse.addListener(myListener);
```

（3）タイマ開始

4.1.2（3）節で示したように、タイマ開始のための現在時間の取得と共通変数へのセーブ処理のみを行う

4.2　ステージ分割集中制御方式

4.2.1　レイヤおよびフレーム構成

動作ステージ間の移動を制御するcontrolレイヤ、キャラクタの動きを直接制御するcharaレイヤおよびフレーム名を表すlabelレイヤで構成する。

また、3項で示したキャラクタの動作ステージに対応してフレームを分割し、各々collフレーム（①②に対応）、gatherフレーム（③に対応）、scatフレームと呼ぶ。このようにフレームを動作ステージ分割することにより、

(a) 各ステージの動作が簡便に記述でき、動作追加・変更に対応し易い

(b) ステージ毎にキャラクタの形状や表情を変化させることができる

等のメリットがある。

4.2.2.　ステージ内処理

以下に各ステージの機能概要と特徴的なスクリプト構成を示す。

（1）control レイヤ、

(a) coll フレーム

イベントリスナーによりマウスクリックを検出し、マウス位置をセーブしたあと、gather

フレームに飛ぶ（gotoAndPlay("gather")）（**4.1.3項（2）**参照）。

(b) gather**フレーム、**

イベントリスナーによりキーダウンを検出し、現在時刻をセーブ（**4.1.3項（3）**参照）したあと、scatフレームに飛ぶ（gotoAndPlay("scat")）。

（2）chara**レイヤ**

全てのインスタンスについて、3項①、②の動作を制御するスクリプト（**4.1.2**項とほぼ同様のスクリプト）を記述する。ただし、それ以外に以下のスクリプトを記述するフレームが必要である。

・ステージ間を遷移（coll→gather、gather→scat、scat→coll）する場合、遷移元のステージで全インスタンスの座標情報をセーブし、遷移先のステージにおいて復元する（記述例は以下の各項参照）。

(a) coll**フレーム**

・インスタンスを複製する（1フレーム目）

ライブラリにあるムービークリップシンボルをattchMovie( )メソッドを使って必要数複製し、各々のインスタンスの座標をランダムに決め、変数にセーブする。

```
for (i = 0; i <=40; i++) {
        newName = "chara" +i ;
        new_mc  = this.attachMovie("chara_mc", newName, i );
}
```

・インスタンスの座標をセーブした変数からインスタンスの位置を初期設定するとともに、各インスタンスの移動方向と速度をランダムに決定する（4.1.2項**（2）**参照）。scatステージからcollステージに遷移する場合には本フレームに戻って同様の処理を行う。（2フレーム目）

・全インスタンスをランダムな速度でランダムな方向に動かしながら、自分以外のインスタンスとの衝突制御を行う。（3フレーム目）

```
for (i =0 ; i<= 40 ; i++) {
    for (j = 0; j <=40; j++)  {
        if (i != j){
            if (_root["chara"+i ].hitTest(_root["chara"+j])) {
                _root["moveX_"+i ]  = - _root["moveX_"+i ];
                _root["chara"+i]._x  +=_root["moveX_"+i]
                                    +Math.floor(Math.random()*9)-4;
                // Y軸方向についても同様の処理を記述
        }
```

```
        }
    }
}
```

4フレーム目では全インスタンスの位置座標をセーブした後、前のフレームに戻ることにより、動作を連続させる。位置座標をセーブするのは、連続動作中にマウスがクリックされ、gatherステージに遷移した場合にインスタンスを元の位置（collステージの位置）に復帰させるためである。

```
for (i =0 ; i<= 40 ; i++)  {
   _root["posx_"+i ]  = _root["chara"+i ]._x;
   _root["posy_"+i ]  = _root["chara"+i ]._y;
}
gotoAndPlay(_currentframe-1);
```

(b) gather**フレーム**

・全インスタンスの位置をcollフレームから受け継いだ座標に復帰させる（５フレーム目）。

```
for (i =0 ; i <= 40 ; i++)  {
   _root["chara"+i ]._x =_root["posx_"+i ] ;
   _root["chara"+i ]._y =_root["posy_"+i ] ;
 }
```

・全てのインスタンスについて4.1.2項(2)(b)と同様の処理を行う（6フレーム目）。
・全インスタンスの位置座標をセーブした後、前のフレームに戻ることにより、動作を連続させる（7フレーム目）。

(c) scat**フレーム**

・全インスタンスの動作方向、動作速度（一斉に早く逃げる速度）を設定し、インスタンス位置をgatherフレームから受け継いだ座標に復帰させる。（8フレーム目）
・全インスタンスをランダム方向に移動させる。また、現在時刻から経過時間を計算し、所定の時間が経過すればcollフレーム（2フレーム目）に戻る。（9フレーム目）
・全インスタンスの位置座標をセーブした後、前のフレームに戻ることにより、動作を連続させる（10フレーム目）。

５．評価と適用領域

５.１　独立分散制御方式

（ア）個々のインスタンスの振る舞いをほぼムービークリップアクションのみで記述することとなり「書き下ろし」感覚で記述できるため、動作の全貌が把握し易い。

（イ）１つのプログラム内に全ステージの処理を分岐しながら記述するため、プログラムが長くなり、処理ステージ数が多い場合には構造が複雑になる。
（ウ）ステージ間で引き継ぐ情報は無いため、プログラムの総量は少なくて済む（4.1節で述べた実現例の場合、約90ステートメント）。
（エ）ステージ毎にキャラクタの大きさや回転角度を変化させることはできるが、表情や形状そのものをステージ毎に変化させる必要がある場合に対応不可能である
（オ）タイムラインの連続処理が簡便（onClipEvent (enterFrame)）に記述できる
（カ）動作可能なインスタンス数はステージ分割集中制御方式の場合よりも少ない（処理能力が低い）。動作環境1（CPU:Pentium4 1.6GH、MM:256MB）ではインスタンス数が25以上になると動作速度が低下。動作環境２（CPU:Pentium4 3GH、MM:512MB）ではインスタンス数が35以上になると動作速度が低下。
（キ）インスタンスの数を変更する場合の変更箇所が少ない（2箇所）。
（ク）衝突チェックの記述が複雑（自分のインスタンス番号も取り出し）

５.２　ステージ分割集中制御方式

（ア）プログラムがステージ毎に分割して記述できるため、わかりやすい。
（イ）本例ではインプリメントしなかったが、インスタンス（ムービークリップ）そのものをステージ毎に入れ替えることによって「より表現力豊かな」FLASHムービーを作成する場合に有利である。
（ウ）各インスタンスの位置情報のようにステージ間で引き継ぐ情報が必要となるため、プログラム量が増える（4.2節で述べたの実現例では約120ステートメント）。
（エ）タイムラインの連続処理の記述が複雑（gotoAndPlay(　)等）
（オ）動作可能なインスタンス数は独立分散制御方式の場合よりも多い（処理能力が高い）。動作環境1（CPU:Pentium4 1.6GH、MM:256MB）ではインスタンス数が35以上になると動作速度が低下。動作環境2（CPU:Pentium4 3GH、MM:512MB）ではインスタンス数が45以上になると動作速度が低下。したがって、インスタンス数を多くしたい場合に処理能力的には有利である。
（カ）インスタンスの数を変更する場合の変更箇所が多い（12箇所）。ただし、インスタンス数を表す変数を準備すれば問題ない。
（キ）衝突チェックの記述が容易（自分のインスタンス番号の取り出しが不要）

６．今後の課題

キャラクタ動作表現法の改善、インスタンスをさらに増やした場合の処理能力の改善が今後の課題である。キャラクタ動作については、常に同じ速度で動き回るだけでなく、例えば、①動作領域（虫篭）の端に近づくにしたがって動作速度が落ちる、②ランダムに動作速度や動作方向が変化する、等の表現法が考えられる。また、キャラクタ同志が衝突した後の逃げ方についても、①最初はあわてて逃げるが、そのうちにゆっくりした動作となる、②お互い

に避ける方向や速度に一定のルールを適用する、等の表現法が考えられる。さらに、キャラクタがマウスに近づいてくる場合には、マウスまでの距離に反比例する速度とすることにより、より現実感のある動作表現とすることができると考えられる。マウスから逃げる場合にも、速度ベクトルにマウスからの距離を反映させて反比例するよう設定すれば、より高度な動作表現が実現できると考えられる。一方、処理能力の改善については、衝突検出法の改善、位置変更処理の高度化等、について検討が必要である。

７．あとがき

多数のインスタンスの動作を独立して制御し、かつ外部イベント（マウスクリック、キー押下、タイムアウト処理）により、全インスタンスが統一された処理を行うよう制御するためのActionscript構成法について検討し、具体的なスクリプト構成を明らかにした。

スクリプト構成法としては、全てのインスタンス動作をムービークリップアクションとして各インスタンスに分散して記述する独立分散制御方式と、全キャラクタの動きを動作ステージ毎に集中してフレームアクションとして記述するステージ分割集中制御方式が考えられる。本稿では2方式の実現例を示すとともに、実際のスクリプト作成結果およびムービーの動作結果から、2方式の得失を評価することのより各々の適用領域について明らかにした。

今後は、動作ステージ毎にキャラクタの形態が変化しながら、さらに複雑な動作をする多数のインスタンスを効率的に制御するためのスクリプト構成法について検討を行う予定である。

参考文献

（1）土屋徳子：Macromedia FLASH MX for Windows、株式会社ラトルズ、2002年
（2）植木友浩：FLASH SERVER-SIDE SCRIPT SAMPLES、株式会社翔泳社、2002年