経営論集　第17巻第1号　2007年　211～226頁

Action Scriptを用いたプログラミング教育

小　松　香　爾

はじめに

JavaScriptがWebページ専用のプログラミング言語であると同様に、ActionScriptはFlash専用のスクリプト言語である。もともとFlashは純粋なアニメーション制作ソフトであり、インタラクティブな仕組みはボタン以外になく、本格的なアプリケーション制作には使用できなかった。ただし、HTMLのフォームに相当する部品は、画像やムービーとして作成できた。これらの部品を制御するために開発された言語がActionScriptである。まず、単純な枠組みがあり、後から制御用に開発された言語として、JavaScriptとActionScriptは類似点がある。またActionScriptの言語仕様は、JavaScriptと同じECMAScriptに基づいている。従って、ActionScriptを習得すれば、JavaScriptの習得も容易に習得できるといえる。

現在、高校、大学、企業のどこにおいても、プログラミング教育は体系化されていない。特に、インターネットの登場後は、ソフトウェアに求められるものが、速いペースで変化しているので、体系化することには無理がある。現状を見ると、ソフトウェア開発はWeb関連の案件が多くなった。しかし、最小限必要な知識は、変わっていない。それは、イベントとイベントハンドラ、および制御構文の理解である。前者が必要な理由は、人間がコンピュータを操作する限り、UIは消えてなくならないことである。後者が必要な理由は、手続き型言語には、全て制御構文が存在するからであることと、手続き型言語は、コンピュータアーキテクチャがフォン･ノイマン型である限り、主流であり続ける。また、最近の言語は、全てオブジェクト指向になっているので、オブジェクトへの理解も必要といえる。

本論文では、ゲーム制作の実例を通して、ActionScriptの教育用言語としての使用を提案する。

1　各種プログラミング言語

工学系の学科におけるプログラミング教育は、ハードウェア制御、電子計測などを目標に行われる。例えばロボット制御用のプログラムなどは、ソフトウェアの規模が小さく、速度が重要であるため、主にアセンブリ言語やC言語で記述される。数値計算の分野ではC言語の代わりにFortranが使われる場合もある。アセンブリ言語では、プロセッサのレジスタ操作や、アキュミュレータの値のメモリへのストア操作まで、全て人間が記述しなければならない。C言語では、マイクロプロセッサへの変数の割り当てなどは自動化されているが、メモリ管理やポインタによるアドレスの参照を避けて通ることができない。アセンブリ言語は低級言語であるが、

も高級言語の中では、抽象度が極めて低いといえる。したがって、これらの言語の習得には、マイクロプロセッサやメモリなどに関する概念の習得が不可欠である。C言語で書けば、何でも書けてしまうが、それゆえに初心者には危険な言語である。アセンブリ言語もC言語も、ハードウェアよりの速度を優先した言語であるため、人間による可読性や、アルゴリズムの記述のしやすさ、プログラムの再利用性などは、重視された設計になっていない。

情報系の学科では、プログラミング言語自体が研究対象となることもある。現在、主流となっているJava、C++などのオブジェクト指向言語も、ソフトウェア開発の効率化、プログラムの再利用性などの研究から誕生した経緯がある。また、Lisp、Haskellなどの関数型言語、Prologなどの論理型言語など非手続き型言語も教育、研究されてきた。しかし、現状では、手続き型以外の言語は、研究段階に留まりつづけている。理由として、大規模なプログラムの組みにくさ、生産性の低さなどが挙げられる。また、利用者の少なさから、ライブラリや書籍が充実しないという側面もある。ただし、Lispの「プログラムをデータとして扱える」という性質は、現在使用されているプログラミング言語の設計に大きな影響を与え続けている。また、Lispのエッセンスだけを抽出し、Schemeは米国で教育用プログラミング言語として使用されている。Schemeは初期のLisp同様、言語設計が非常にコンパクトである。文法規則は、“(”と“)”の対応と、演算子と非演算子の前置記法ぐらいしかない。初期のLispでは、変数のスコープが動的スコープであったが、Schemeでは、安全な静的スコープが採用されている。また、プログラム自体をデータとして扱えるため、複雑なアルゴリズムを簡潔に記述することができる場合が多い。実際、計算機科学の教科書として定評がある「Structure and Interpretation of Computer Programs」は、Schemeの実行環境を実例として書かれている。総合すると、Schemeは、アルゴリズムを教育するための言語に向いていると言える。しかし、現在主流である手続き型言語とは、あまりに文法が異なり、しかも動作がわかりにくい。プログラミング教育に時間を割けない学科では、教育用言語として適さない。

現在、企業で使用されている言語はC、C#、C++、Java、VisualBasic、VBA、Ruby、JavaScript、PHPなどである。このうち、JavaScript、PHP以外はデスクトップアプリケーションやアプリケーションサーバを作成するために使用される。また、その際に、Eclipseなどの統合開発環境や、NETやRailsなどのフレームワークの使用が前提になることが多い。「Hello World」と画面に表示する程度のプログラムを作るために、統合開発環境の使い方を覚えたり、環境を整えなければならないのでは、教育用言語としては使用しにくい。

2　教育用プログラミング言語

以前は、教育用言語としては、Basicが用いられてきた。現在もBasicは存在しているが、Visual Basic.netとなり、教育用として、手軽な存在ではなくなった。本節では、JavaScriptの重要性と、JavaScriptから派生したAction Scriptの教育用言語としての利点を述べる。

2-1　JavaScript

JavaScriptの重要性は、近年、従来にもまして、高まってきた。GoogleのWebサービスのほとんどがJavaScriptで記述されているからである。従来は、パソコンのアプリケーションソフトとして実現されていた機能が、Googleによって、オンラインサービスに置き換えられつつある。ただし、JavaScriptの習得を習得するためには、Webサイト構築、少なくともHTMLのタグ打ちの知識が必須である。JavaScriptはHTML文書の中に埋め込まれる目的で作られた言語だからである。HTMLは「ハイパーテキストをマークアップする言語」という名称が示すとおり、ハイパーリンクの仕組みを持つ文書に対して意味づけを行うデータ記述言語である。したがって、HTML文書は構造化された静的な文書であり、プログラムとして実行されるわけではない。また、初期のHTMLには、リンク以外のインタラクティブな仕組みはなかった。後に、フォーム、すなわちボタン、テキストボックス、テキストエリア、ラジオボタン、チェックボックスなどのユーザインターフェース構築の為のHTML要素が加わった。フォームに属性を指定することで、サーバサイドではCGI、クライアントサイドではJavaScriptのプログラミングが行われるようになった。

JavaScriptとPHPは、第1節で挙げた他の言語より、遙かに手軽にプログラミングができるといえる。「HTML埋め込み型」という特徴があるからである。ただし、JavaScriptがクライアントサイド（ブラウザ）でプログラムが動くのに対して、PHPはサーバサイドで動く。HTMLに埋め込まれたPHPのプログラムは、Webサーバ上で動作し、Webサーバがブラウザに送り返すデータは通常のHTMLである。したがって、PHPのプログラムを動かすためには、Apache、IIS、AN HTTPDなどのWebサーバがインストールされている必要がある。また、クライアント･サーバモデルを理解していないと、PHPが動く仕組みは理解できない。JavaScriptはブラウザがインストールされていれば、プログラムを動かすことができる。HTML文書をローカルのフォルダに置いても、ブラウザさえあれば問題が起きない。現在、JavaScriptの実行環境は、ほぼすべてのブラウザに標準的に装備されている。学生が自宅のパソコンでも自習できるという点では、教育用プログラミング言語として適している。

従来、JavaScriptは、「ポップアップウィンドウを出す」、「動的なメッセージを表示する」、「時計を埋め込む」、「HTMLのフォームに入力された内容をチェックする」など、単純な目的に使用されてきた。ところが、Google社が2005年に、HTMLマップなどをリリースしたことで、JavaScriptは再評価された。ブラウザ上で、デスクトップアプリケーションに迫る操作性を実現できることが実証されたのである。最近では、JavaScriptで、様々なWebサービスが開発されている。「DOM（Document Object Model）を使用して、HTMLやXMLの要素を操作する」、「XmlHTTPRequestメソッドで、ブラウザとサーバの間でXMLデータ通信を行う（いわゆるAjax）」、「prototype.js、jQueryなどのライブラリを用いて、Ajaxの実現、DOM操作」などのテクニックが用いられている。しかし、JavaScriptには、未だに多くの問題がある。[1] 特に一番

な問題は、ブラウザ間の互換性である。同じソースコードでも、ブラウザによって、動かなかいことがある[1]。また、教育用言語という観点でみると、先にHTMLを覚えなくてはならないという点には問題がある。HTMLは簡単ではあるが、Webページ記述用「言語」である。プログラミング言語を学ぶために、他の言語を覚えなくてはならないのは、本末転倒である。また、HTMLに埋め込むという記述形式も、言語の中に言語を記述することになり、初心者にとって混乱を招きやすいといえる。

2-2　ActionScript

プログラムは、どのタイミングで、どのような処理を行うかを、コンピュータにも分かる言葉で書いたものに過ぎない。プログラムの全体像を明確に見渡せる言語こそ、教育に向いた言語といえる。その点において、FlashのActionScriptには、最初に画像があるという利点がある。また、画像（正確にはムービークリップやボタン）に、直接、プログラムが書けるという特徴は、オブジェクト指向の概念の習得に適している。ただし、「オブジェクト」に関しては、様々な定義がある。最初のオブジェクト指向言語であるSmallTalkにおける定義は、「データにプログラムを書いたもの」である[2]。この定義に従えば、ActionScriptでは、画像にプログラムを書くことにより「オブジェクト」を生成していることになる。「最初にデータありき」というオブジェクト指向の思想を、画像にプログラムを書くことで、具現化しているといえる。ActionScriptを使ったプログラミングでは、「インタラクティブアニメーション」、「Webページ」、「Webページのユーザインターフェース」、「ゲーム」などが挙げられる。本論文では、インタラクティブ性がもっとも高いゲームプログラミングを取り上げる。

3　Action Scriptによるゲームプログラミング

なぜプログラミング教育の題材として、ゲームを選ぶかについて述べる。実際にAction Scriptを用いたゲーム制作例を題材とし、Action Scriptによるゲーム制作がプログラミング教育に適していることを示す。

3-1　ゲームグラミングの利点

ゲームとして成立させるためには、ユーザの操作に対する反応が必要である。それゆえ、イベントやイベントハンドラの重要性が強調できる。だが、インタラクティブ性の重要性だけならば、ユーザインターフェースの構築を通してでも学べる。ゲームの利点は、プログラミングの結果、出来上がったもので遊ぶ楽しさがあることと、シューティング、アクション、ロールプレイング、アドベンチャー、サウンドノベル、パズルなど、様々なジャンルが存在することである。ゲーム制作の面白さは、プログラミングの世界では、「ゲームを作ることが最大のゲーム」という格言があることからも証明されている。Webのユーザインターフェースならば、HTMLのハイパーリンクだけですますことができるが、ゲームの場合は、どのようなゲームを作るかによって、プログラムの種類も全く異なるものになる。また、現在の10代～20代は、子供の頃からビデオゲームに親しんできたという事実から、学生になじみが深く、目標がわか

りやすいことも利点に挙げられる。

3-2　Action Scriptによるゲームの制作

Action Scriptによる、ゲームプログラミングの最大の利点は、図形描画の為のプログラミングを、ほとんど行わなくてもよいことにある。通常のプログラミング言語の場合、図形を描画するためには、ライブラリの関数を覚えなければならない。また、図形描画ライブラリがない場合は、自作ライブラリを作らなければならない。Flashは、もともとアニメーション制作ツールであるため、図形描画は通常のグラフィックスツールの感覚で行える。

ActionScriptで作成できるゲームの種類は多数ある。ただし、教育目的に適するものは、限られる。アクションゲームは、ユーザが操作するキャラクタのジャンプ時などに、物理法則などの知識が必要になることに加え、スクロールの技術が必要になる場合が多い。ロールプレイングゲームは、構成する要素の多さから簡易データベースを作成する必要があり、またファイルのロード、セーブ知識も必要である。逆に、アドベンチャーゲームやサウンドノベルは、シーン遷移の仕組みさえ作ればいいので、教育目的には簡単すぎるといえる。本論文では、教育用の題材として、シューティングゲームを取り上げる。シューティングゲームは、ユーザが操作する自機、自機から発射される弾、ぶつかってくる敵があれば、楽しめるゲームが作れる。

FlashとActionScriptでゲームを制作すれば、Web上で手軽に公開することができる。Flashの再生環境である「Flash Player」の普及率は極めて高い。「Adobe」社のサイトによると、日本のパソコンの98.3%に、「FlashPlayer6」以降がインストールされている［2］。この98.3%という数字は、Windowsの標準ブラウザである「Internet Explorer」の普及率を上回ることはもちろん、「JavaScript」 を有効にしているブラウザの率をはるかに上回っている。なお、「Flash Player」はパソコンだけでなく、PDAやゲーム機などにもインストールできる。携帯電話には、あらかじめ「Flash Light」［3］という「Flash Player」が搭載されている。

3-3　教育目的のゲームプログラミング

Action Scriptには、ActionScript1.0、2.0、3.0があり、仕様や推奨される記述方法に違いがある。本論文では、「データにプログラムを記述する」というオブジェクト指向の原点を見据え、「ActionScript1.0」の記述方法を採用する。「ActionScript1.0」の記述方法では、オブジェクトアクションとフレームアクションを同時に使用する。なお、「ActionScript2.0」以降では、オブジェクトアクションは推奨されていない。デザイナーとプログラマの完全分業、スクリプトの実行タイミングの問題(3)を考慮すると、確かに、オブジェクトアクションは使用しないほうがよいといえる。しかし、オブジェクト指向になれていない学生にたいし、オブジェクトの概念を教育するためには、オブジェクトアクションを使用したほうがよい。描画した画像に、直接プログラムを書けるという、他の言語にはないAction Scriptの特徴が生かせるからである。また、教育用という観点から、できるだけ構成部品を減らす方針が望ましい。本論文では、ムービークリップは5つしか作成されない。図1にスタート画面を示す。スタート画面には、自機を表す三角形、自機から発射される弾を表す円、敵を表す長方形、ゲームスタートボタンを表す角

丸長方形が配置されている。これらは全てムービークリップシンボルのインスタンス（ムービークリップオブジェクト）である。ただし、ゲームスタートボタンは、ボタンシンボルとしても作成できる。しかし、「ActionScript1.0」以降では、ムービークリップオブジェクトにも、ボタンシンボル用のイベントハンドラを記述できるようになった。本論文では、「オブジェクトに書くスクリプト（オブジェクトアクション）は、全てムービークリップオブジェクトに書く」という教育方法を推奨する。その理由は二つある。第1に、ボタンオブジェクトの相対パスのターゲットパス記述が、ムービークリップオブジェクトと異なるということである。Flashでは、ボタンシンボルは、ムービークリップシンボルの付属的な扱いである。それゆえ、ボタンシンボルのインスタンス（ボタンオブジェクト）に「this」というターゲットパスを記述すると、ボタンオブジェクトが配置された、ムービークリップシンボルのインスタンスが参照されることになる。それに対して、ムービークリップオブジェクトに「this」と記述すると、記述されたムービークリップオブジェクト自身が参照されることになる。通常のオブジェクト指向言語ならば「this」は、「記述されたオブジェクト自身」が参照されるので、後者の方が一般的な解釈といえる。オブジェクト指向言語では、「this」は頻出する。したがって、他のオブジェクト指向言語の習得につなげるという側面からも、ボタンオブジェクトは使用すべきではない。第2に、ムービークリップオブジェクトの方が、様々な効果や動作を実現できるということである。これは、ムービークリップクラスのほうが、ボタンクラスより、プロパティやメソッドが豊富に用意されていることによる。ボタンオブジェクトで表現可能なことは、ムービークリップオブジェクトで全て表現可能である。

Action Scriptで書かれたプログラムの大まかな構造は、Flashのタイムラインを確認することにより把握できる。それが、通常のテキストベースのプログラミング言語との違いの一つでもある。本論文のゲームのキーフレームは、3つである。最初のキーフレーム（1フレーム）に開始画面を表示させる。この開始画面には、スタートボタンのムービークリップオブジェクトを配置する。本論文のゲームのレイヤーは、3つである。一番下のレイヤーは、ゲーム画面を表示するレイヤーである。真ん中のレイヤーは、ゲームスタートボタン、リスタートボタンを配置するレイヤーである。一番上のレイヤーは、スクリプトを書くためと、キーフレームに名称をつけるためのレイヤーである。

3-4　最初のフレームのスクリプト

一般的な、ゲームの場合、スタートボタンを押すまで再生を進めてはならない。したがって、最初のキーフレームの1フレーム目に「this. stop();」というフレームアクションを書く。この場合、thisはメインタイムラインを持つムービークリップを表す。stop()はムービークリップの再生を停止するというメソッドである。ピリオドは、通常のオブジェクト指向言語で、オブジェクトのメソッドを呼び出すために用いられるドット記法である。スクリプトの意味は、「メインタイムラインに対して、再生を止めるというメソッドを実行する」という意味になる。図1の中央の、ゲームスタートボタンを押すと、ムービークリップオブジェクトに記述されたスクリ

プトが実行され、次のキーフレーム（5フレーム）が再生される。このボタンのムービークリップには、図2のように、以下のスクリプトを直接記述する(4)。

```
On (release) {
        this._parent.gotoAndStop("main");
}
```

「on」がイベントハンドラ、「release」がイベントで、「on（release)」は、「マウスクリックが離されたとき」という意味になる。このスクリプトの意味は、「ムービークリップオブジェクトの上で、マウスクリックが離されたときに、「main」と名付けられたフレーム（以降、メインのフレームと呼ぶ）に移動して、再生をストップする」である。「this._parent」というオブジェクトは、メインムービークリップ（メインタイムラインを持つムービークリップ）を表すことになる。Flashでは全てのムービークリップの直接のムービークリップは入れ子関係になり、ボタンのムービークリップはメインムービークリップの直接の子供である。したがって、ボタンのムービークリップから、メインムービークリップを参照する場合、_parentと指定することになる。メインのフレームには、「this.stop();」というスクリプトを、フレームアクションとして記述する。記述する理由は、最初のキーフレームの場合と同様である。したがって、次のフレームが勝手に再生されることはない。

3-5　メインのフレームのスクリプト

メインのフレームには、自機、敵、弾を表すムービークリップに加えて、得点を表すダイナミックテキストを配置する。

3-5-1　自機のムービークリップのスクリプト

自機のムービークリップには、図3のように以下のスクリプトを記述する。

```
onClipEvent (enterFrame){
        this._x = this._parent._xmouse;
}
```

「onClipEvent」は、ムービークリップ用のイベントハンドラであり、「enterFrame」は、このフレームが再生された時というイベントである。「onClipEvent (enterFrame)」は、「フレームが再生されるたびに、ムービークリップは～の動作をする」という意味になる。タイムラインで再生が止まっていても、止まっているフレームは再生しつづけられている。フレームレートが12fpsならば、1秒間に12回、フレームを再生して、再描画を行うことになる。「this._x = this._parent._xmouse」は、「メインのムービークリップ上のマウスのx座標の値を、自機のムービークリップのx座標の値に代入する」という意味である。ユーザがマウスを操作するとマウスの横の動きだけが、自機の動きに反映されることになる。ムービークリップのx座標のプロ

パティに、メインのムービークリップ上のマウスのx座標を代入するということが、決して数学の「等しい」という意味ではないことを学ばせる。

3-5-2　弾のムービークリップのスクリプト

弾のムービークリップには、図4のように以下のスクリプトを記述する。「load」は最初に「ムービークリップが登場したとき」を意味するイベントであり、続く中括弧の中のスクリプトは、1回しか実行されない。「_visible」はムービークリップクラスのプロパティであり、「false」という値を代入することにより、最初は、見えない状態にしている。「speed」は、変数であり、弾の移動速度として20を代入している。この「speed」は、自作プロパティとも解釈できる。hitは自作メソッドであり、弾が敵に当たったときに、弾を不可視にして、点数を1増やすスクリプトが定義されている。これは、「オブジェクトのプロパティは、そのオブジェクトに定義されたメソッドによってのみ変更されるべきである」という、カプセル化を学ばせるためである。カプセル化は情報隠蔽とも呼ばれ、オブジェクト指向言語の基本概念である。なお、「tensuu」も変数名であり、ダイナミックテキストの変数名に「tensuu」とつけてフレームに配置しておく。図5の左上の0の並びがダイナミックテキストである。下記の「mouseDown」は、「マウスボタンが押された」というイベントであり、もし、このとき、弾の「_visible」プロパティが偽である場合は、真にして、自機（houdai_mcというインスタンス名をつけておく）の先端に配置されるようにx､y座標を設定する。

弾が可視状態である場合は、フレームが再生される度に「this._y -= speed」で、弾をステージの上方に移動させる。弾がステージ外に出た場合、すなわちy座標がゼロより小さくなった場合には、弾を不可視状態にする。不可視状態になっているということは、再びマウスをクリックしたときに、自機の先端に、弾が配置されるということになる。

```
//最初に登場した時は、不可視状態でスピード20に設定
onClipEvent (load) {
        this._visible = false;
        speed = 20;
        function hit() {
                this._visible = false;
                this._parent.tensuu += 1;
        }
}

//マウスをクリックした時に、不可視状態だったら可視状態にして、砲台の先端に配置
onClipEvent (mouseDown) {
```

```
        if (this._visible == false) {
                this._visible = true;
                this._x = this._parent.houdai_mc._x;
                this._y = 350;
        }
}
onClipEvent (enterFrame) {
        //可視状態の場合、フレーム再生ごとに上方向に移動
        if (this._visible == true) {
                this._y -= speed;
                //ステージ外に出たら、不可視状態に戻す
                if (this._y<0) {
                        this._visible = false;
                }
        }
}
```

3-5-3　敵のムービークリップのスクリプト

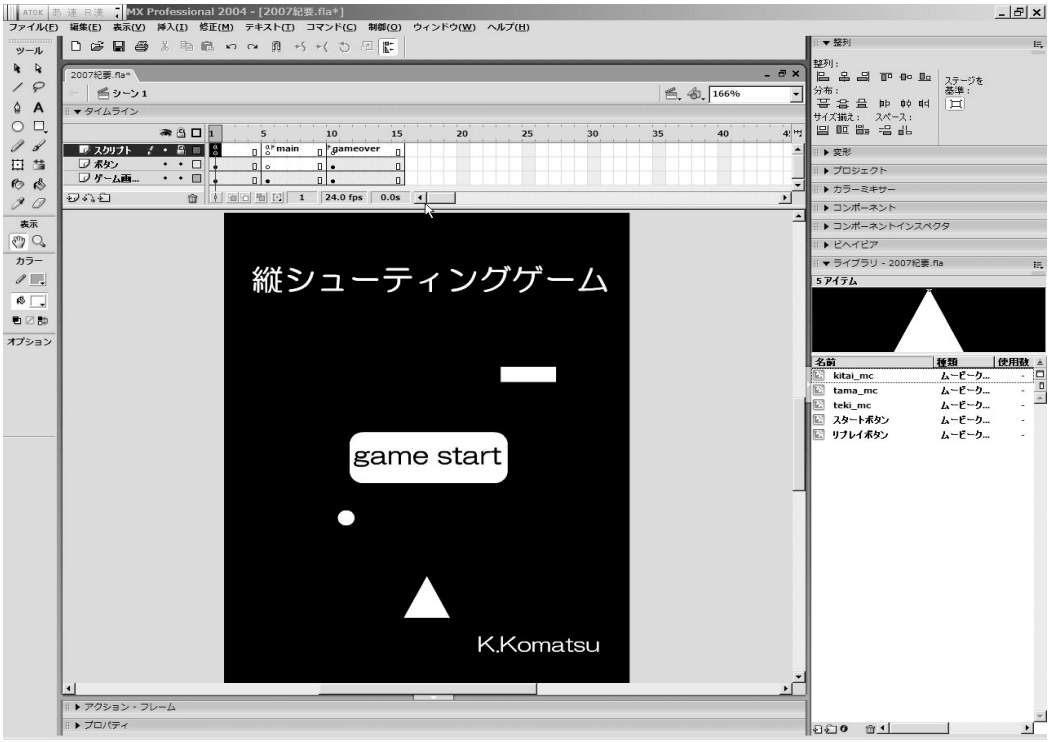
敵のムービークリップには、図6のように、以下のスクリプトを記述する。load時にステージの上端に配置し、縦方向のスピードをyspeed、横方向のスピードをxspeedという変数に代入する。当たり判定は、処理が複雑で、スクリプトも長くなるので、hitというメソッドで定義している。まとまった処理を関数にするという、手続き型言語の基本に従わせることを学ばせる意図がある。hitでは、この敵が可視状態であった場合に、ムービークリップクラスのメソッドである「hitTest」という関数を用いて、弾との当たり判定を行う。hitTest(this._parent.tama_mc)が真の場合、敵を不可視状態にし、弾に定義しておいた、hitメソッドを「this._parent.tama_mc.hit()」で呼び出す。もし、敵が可視状態で、かつ自機との当たり判定「this.hitTest(this._parent.houdai_mc)」が真である場合、ゲームオーバーの場合のフレームへタイムラインを移動させて、再生をストップさせる。次に、フレームが再生されるたびに行われる処理を記述している。「enterFrame」イベントにより、フレームを再生する度に、縦方向と横方向に敵を移動させる。ただし、横方向については、ステージ外に出た場合、「xspeed = -xspeed」で進む向きを逆にする。移動後に、敵のムービークリップオブジェクト自体に定義したhitメソッドを「this.hit()」で呼び出す。それにより、弾および自機との当たり判定をフレーム毎に行うことになる。最後に、敵がステージの下から出た場合についての処理を記述している[(5)]。まず、「this._y = -5」によって、敵をステージの上端の上に戻す。そして、この時、敵が可視（すなわち弾に当たっていない）である場合は、速度を上げる。敵が不可視（すなわち途中で弾に当たった）場合は、可視状態に戻す。「Math.random()」は、Mathクラスのrandomメソッドで、0より大きく1より小さい値を

返す。Mathクラスはインスタンスを生成することなく、メソッドを使用できる。「this._x=Stage.width* Math.random()」でステージの横幅いっぱいのランダムな位置に、敵を再配置する。

```
//ロード時にステージ上端に配置し、スピードを10に設定
onClipEvent (load) {
	this._y = -5;
	yspeed = 10;
	xspeed = 10;
	function hit() {
		if (this._visible == true) {
			if (this.hitTest(this._parent.tama_mc)) {
				this._visible = false;
				//弾に定義したhitメソッドを呼び出す
				this._parent.tama_mc.hit();
			}
		}
		if (this._visible == true && this.hitTest(this._parent.houdai_mc)) {
			this._parent.gotoAndStop("gameover");
		}
	}
}

onClipEvent (enterFrame) {
	//フレーム再生ごとにスピード分だけ下と左右に進める
	this._y += yspeed;
	this._x += xspeed;
	if (this._x<0 ¦¦ this._x>Stage.width) {
		xspeed = -xspeed;
	}
	this.hit();
	//もしステージの下端に到達したら
	if (this._y>Stage.height) {
		//ステージ上端に戻す
		this._y = -5;
		// もし打ち落としていないならば
```

```
if (this._visible == true) {
        yspeed += 5;
        xspeed += 5;
} else {
        this._visible = true;
}
//左右の位置は、ステージの幅の中でランダムにする
this._x = Stage.width*Math.random();
```




```
    }
}
```

図1

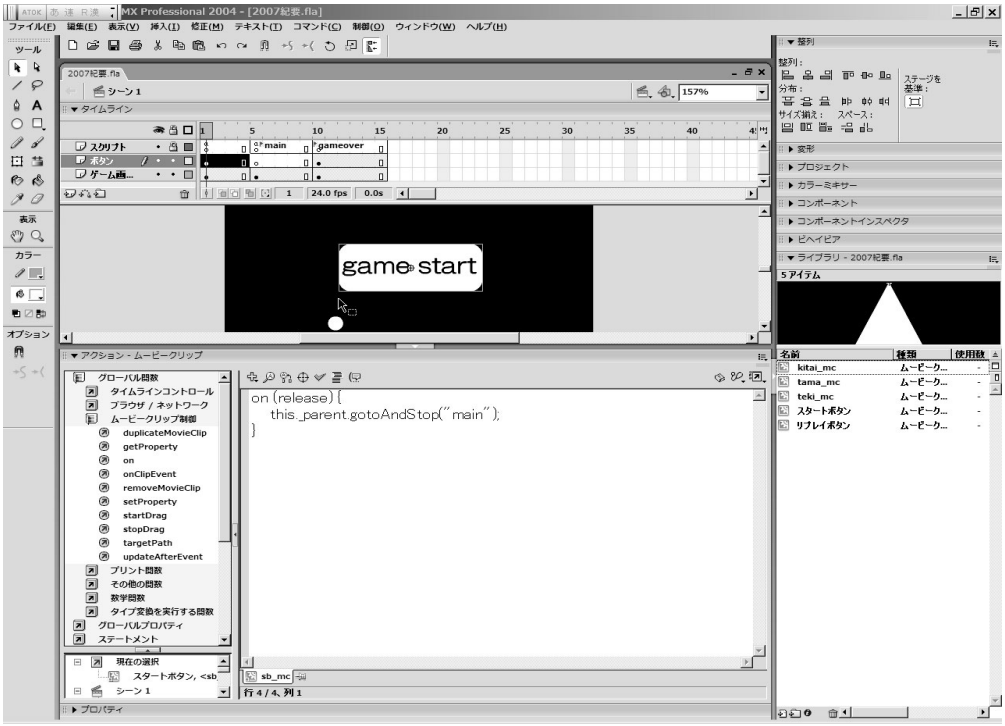

MX Professional 2004 - [2007紀要.fla]
game start
on (release) {
this._parent.gotoAndStop("main");
}

図2

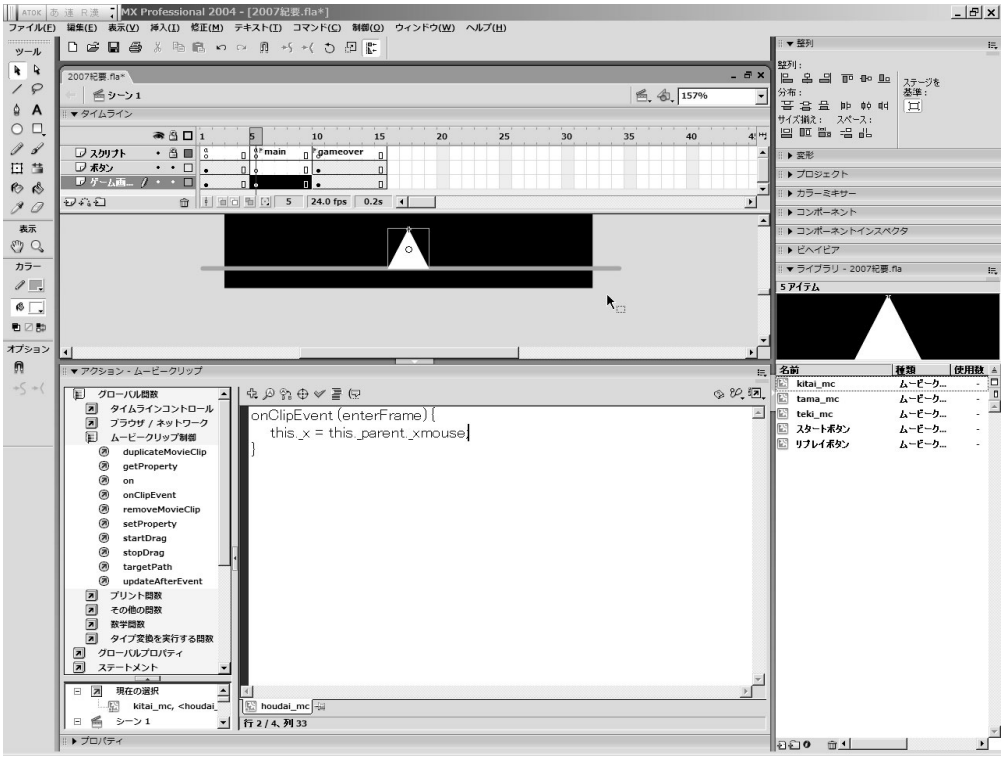

MX Professional 2004 - [2007紀要.fla*]
onClipEvent (enterFrame) {
this._x = this._parent._xmouse;
}

図3

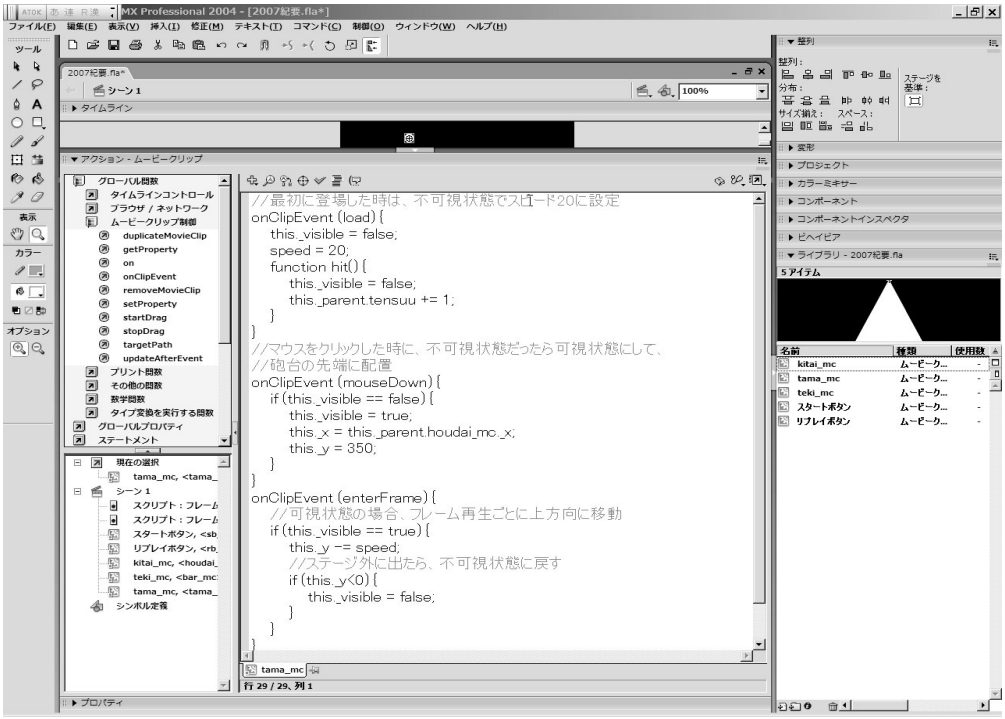

MX Professional 2004 - [2007紀要.fla*]
ファイル(F) 編集(E) 表示(V) 挿入(I) 修正(M) テキスト(T) コマンド(C) 制御(O) ウィンドウ(W) ヘルプ(H)
2007紀要.fla*
シーン 1
100%
タイムライン
アクション - ムービークリップ
グローバル関数
タイムラインコントロール
ブラウザ / ネットワーク
ムービークリップ制御
duplicateMovieClip
getProperty
on
onClipEvent
removeMovieClip
setProperty
startDrag
stopDrag
targetPath
updateAfterEvent
プリント関数
その他の関数
数学関数
タイプ変換を実行する関数
グローバルプロパティ
ステートメント
//最初に登場した時は、不可視状態でスピード20に設定
onClipEvent (load) {
this._visible = false;
speed = 20;
function hit() {
this._visible = false;
this._parent.tensuu += 1;
}
}
//マウスをクリックした時に、不可視状態だったら可視状態にして、
//砲台の先端に配置
onClipEvent (mouseDown) {
if (this._visible == false) {
this._visible = true;
this._x = this._parent.houdai_mc._x;
this._y = 350;
}
}
onClipEvent (enterFrame) {
//可視状態の場合、フレーム再生ごとに上方向に移動
if (this._visible == true) {
this._y -= speed;
//ステージ外に出たら、不可視状態に戻す
if (this._y<0) {
this._visible = false;
}
}
}
tama_mc
行 29 / 29、列 1
プロパティ
整列
ライブラリ - 2007紀要.fla
5 アイテム
名前
種類
使用数
kitai_mc
tama_mc
teki_mc
スタートボタン
リプレイボタン

図4

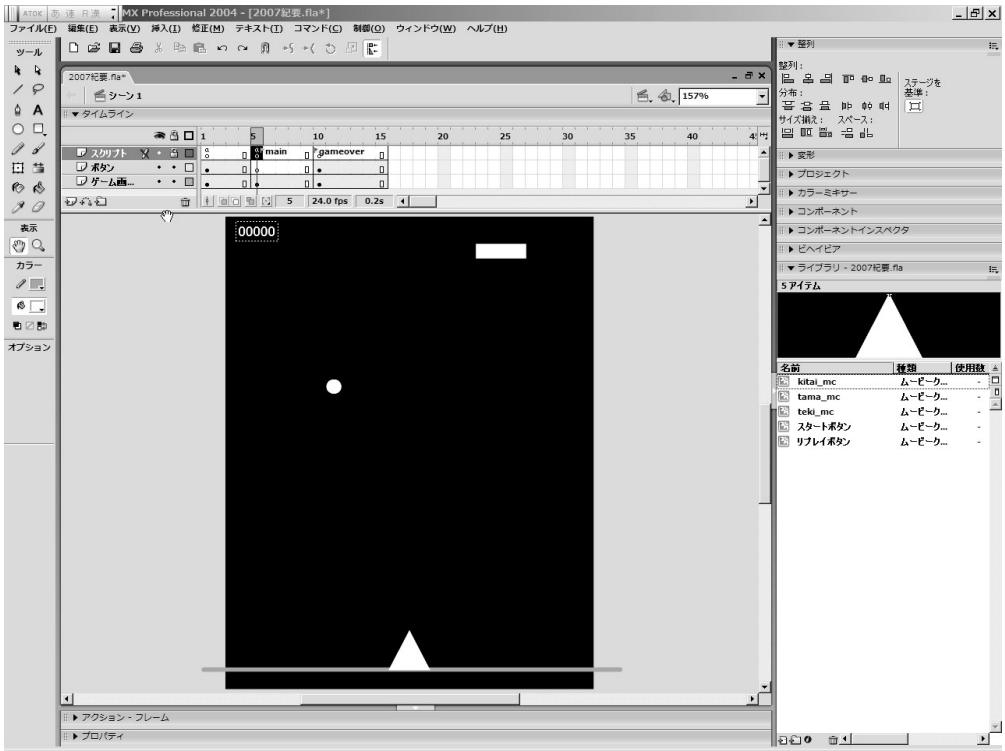

MX Professional 2004 - [2007紀要.fla*]
ファイル(F) 編集(E) 表示(V) 挿入(I) 修正(M) テキスト(T) コマンド(C) 制御(O) ウィンドウ(W) ヘルプ(H)
2007紀要.fla*
シーン 1
157%
タイムライン
スクリプト
ボタン
ゲーム画...
main
gameover
24.0 fps
0.2s
00000
アクション - フレーム
プロパティ
整列
ライブラリ - 2007紀要.fla
5 アイテム
名前
種類
使用数
kitai_mc
tama_mc
teki_mc
スタートボタン
リプレイボタン

図5

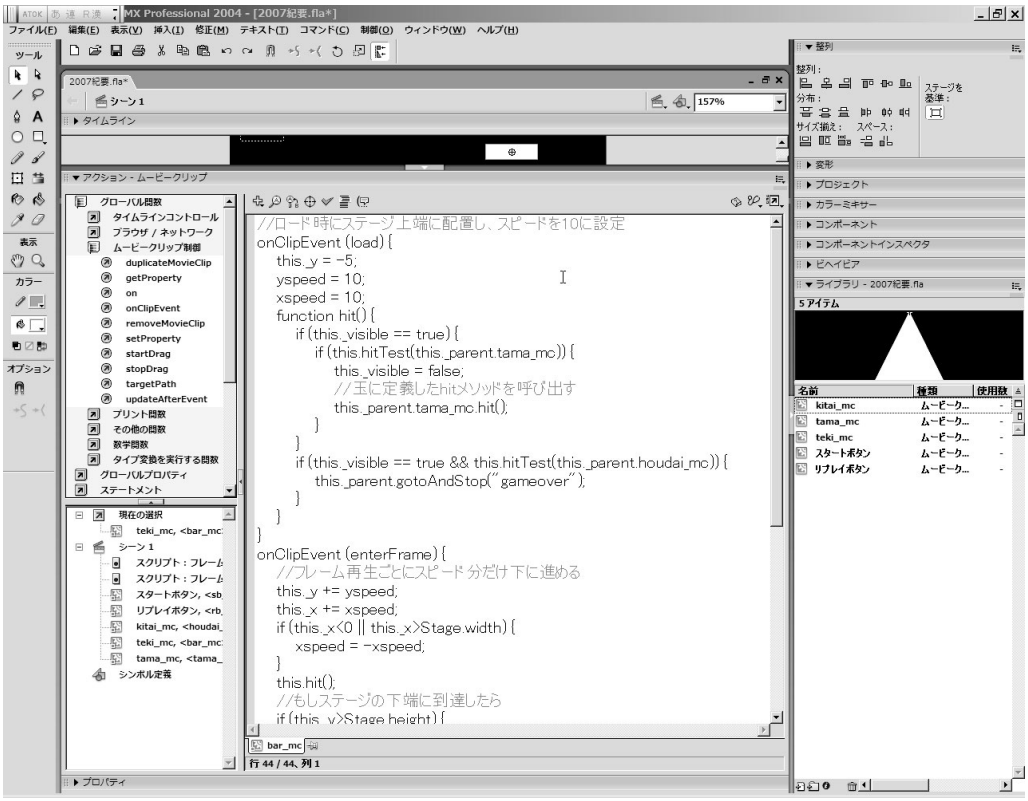


図6

3-6　最後のフレームのスクリプト

敵が自機に当たった場合に、「this._parent.gotoAndStop("gameover")」が実行される。そのとき、最後のフレームへ、タイムラインのカレントフレームが移動する。「gotoAndStop」というメソッドは、フレームへ移動したあと、再生を止める。したがって、最後のフレームには「this.stop()」というスクリプトを書く必要がない。図7の中央の、リプレイボタンを押すと、ムービークリップオブジェクトに記述されたスクリプトが実行され、最初のフレーム（1フレーム）が再生される。このボタンのムービークリップには、以下のスクリプトを直接記述する。

```
on (release) {
        this._parent.gotoAndStop(1);
}
```

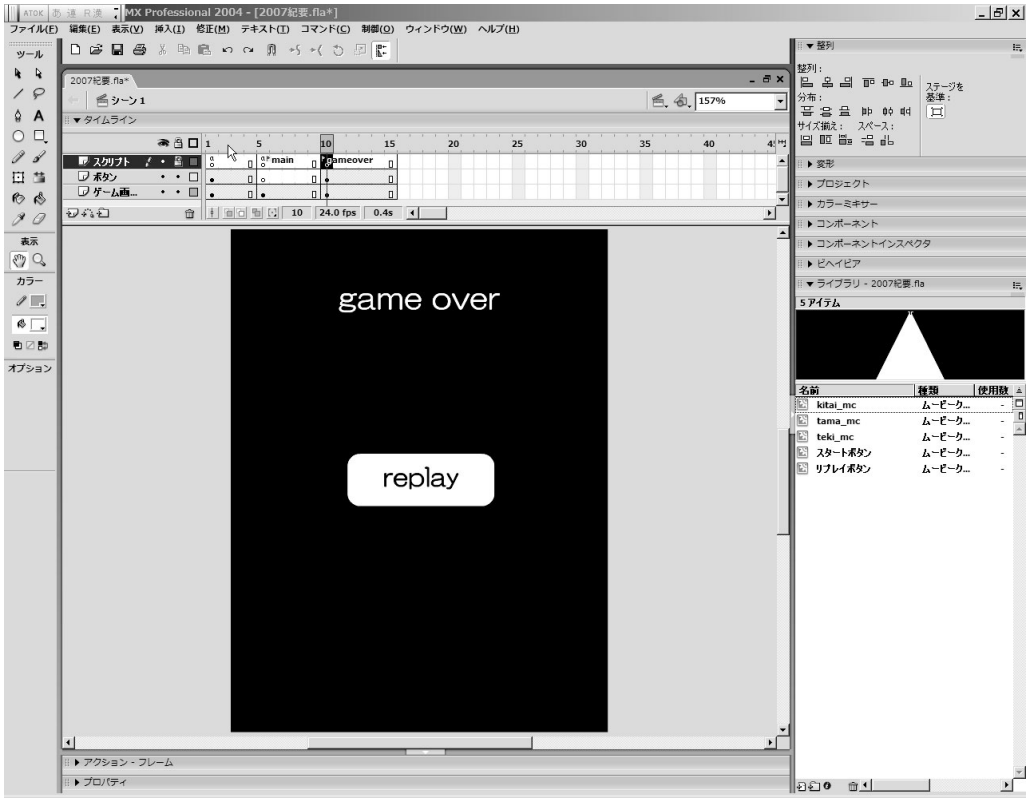


図7

4　まとめ

本論文では、オブジェクト指向の原点を教育するために、Flash の ActionScript を使用することを提案した。ゲームプログラミングの実例を通して、「データへのプログラムの記述」および「オブジェクト間のメッセージパッシング」が ActionScript で、直感的に理解できることを示した。題材としたゲームは、構成要素を最小限に絞った。具体的には、ゲームを構成する3つのムービークリップ、全体を制御するためのボタンを表す2つのムービークリップ、大きな状態推移を表す3つのキーフレーム、ムービークリップを配置するための3つのレイヤーである。ActionScript1.0 のコーディングスタイルでは、オブジェクトに直接スクリプトを記述する[6]。このスタイルで記述すれば、オブジェクトのスクリプトが、イベントが引き金となって動作することと、他のオブジェクトに書かれたメソッドが、メッセージとして呼び出されることを、直感的に理解しやすいプログラミングで最も難しいのは、現実世界の現象を抽象化して手続きに落とし込むことである。その点で、Flash でのスクリプティングは、もともとアニメーションがある状態で行うため抽象化を行う度合いが低いといえる。

（注）

（1）ただし、この問題はprototype.jsといったJavaScriptライブラリを使用することで、ある程度回避できる。

（2）Smalltalk以外の最近のオブジェクト指向言語は、closure（closed procedure）、すなわちデータを含んだプログラムの意味で、オブジェクトを使用している。

（3）フレームアクションの方が、オブジェクトアクションより先に実行されるということである。

（4）「on」は、本来、ボタンオブジェクト用のイベントハンドラであった。「release」も、ボタンオブジェクトに起こるイベントであった。「FlashMX」以降の「ActionScript1.0」では、これらがムービークリップオブジェクトにも記述できるように変更された。

（5）このゲームでは、敵に弾があたっても、敵が不可視状態になるだけで、敵のムービークリップオブジェクトは下に移動し続ける。

（6）FlashMX2004で搭載されたActionScript2.0では、全てのスクリプトは、最初のフレームにフレームアクションとして記述される。

（参考URL）

［1］http://www.crockford.com/javascript/javascript.html

［2］http://www.adobe.com/products/player_census/flashplayer/version_penetration.html

［3］http://www.adobe.com/jp/products/flashlite