卒 業 研 究 報 告

題　　目

Web ベースの大学探検アドベンチャーゲームの開発

指 導 教 員

綿森道夫助教授

報 告 者

学籍番号: 1030193
氏名: 小松 大洋

平成　15 年　1 月　27 日

高知工科大学 電子･光システム工学科

目次

第 1 章　はじめに

本学の学校紹介を見て私はすばらしいとも思います。しかし、実際に本学を歩き学ぶ楽しさをたまたまホームページを開いただけの学生により接しやすい視点で伝えきれていないのではないかとも感じます。本研究ではヴァーチャルの世代に対応した、本学の学校紹介にはない形であるゲーム性を盛り込んだ学校紹介に挑もうと考えました。

第 2 章　研究目的

　本研究では、整ってきた情報通信に対応して Web 上の学校紹介に新たな形態を提案することを目的とします。
また、Macromedia　FLASH の制御言語である ActionScript を習得することにより、“オブジェクト指向プログラミング”の考え方を養い、より円滑にプログラムを構成することを学びます。

第 3 章　使用ソフトウェアの説明

3－1　Macromedia FLASH

人の目を引く Web ページを作成する上で近道なのは画面上で起こるさまざまな動きではないでしょうか。FLASH はそれをより円滑にダイナミックに行うことを可能にしてくれるソフトウェアです。自分で少しずつ絵を変化させながらタイムラインに配置し動きを表現することも、代表的な絵のみを使用しその制御言語である ActionScript でより高度に制御することで視覚に訴える事もできます。また、このソフトウェアは Web 掲載に使用することを考慮しているので、さまざまな環境においてもそのファイルを実行できるように“パブリッシュ”化する機能がつき、受信環境によってのダウンロード状況を想定したストリーミング再生も可能です。

本研究では、主にこのソフトウェアを使用しゲームを作成します。そして、ActionScript で主な動作を制御しています。また、ストリーミング機能を使用し、より Web 向きに調整を行います。詳しくは 4 章以降で説明します。

3－2 Macromedia Fireworks

Web にきれいな写真や絵を載せればもちろんそれはページの“うり”になります。しかし、それによってデータが重くなりページが円滑に表示できないのは、大きな欠点です。その兼ね合いを調整することができるソフトウェアがこれです。画像を自分好みに編集することができ、さらに 4 ウインドウに表示して画像の保存方法（ファイルの形式と色数）による変化や受信側の転送率にあわせた表示時間も見ることができる Web に根ざした機能があります。

本研究では、Web に使用する画像の編集やデジタルカメラで撮影した本学をゲーム内で使用する絵に書き下ろす作業に使用しています。

3－3 Macromedia Dreamweaver

いよいよ Web ページの作成というとき、それを強力にサポートしてくれるのがこのソフトウェアです。自由なデザインをレイヤーという機能を使って可能にしていると共に簡単にですが絵を動かすことができます。また、タグに余計なものがつかないのも“うり”です。そして最大の“うり”は Macromedia Fireworks との連携です。これにより画像をよりたくみに扱うことができます。

本研究では、最終目的である Web 掲載のためのページ作りに使用していきます。

第 4 章　ゲームの製作

4－1　ゲームのイメージ

本研究でゲームを作成する上でまず作成するイメージを固めます。まず考えたのはこんなことです。

- 高知工科大学を紹介するものにすること。
- 大学全体をマップで使用する。
- 平面を移動しながら遊ぶゲームにすること。
- テキスト表示で大学の情報をあたえることする。
- 操作は十字キーとマウスのクリックで行う。
- ロールプレイングゲームにし、バトルシーンやレベルアップをすることにより、プレイヤーの飽きを起こさせないものにする。
- プレイ時間は約 1 時間にする。
- ストーリー
- 主人公と敵キャラクターの設定や絵

4－2　絵の作成

　まず、学校をデジタルカメラで撮影し実際にゲームで使用する全体マップを書くためにデータを収集します。

図 4－1　実際に撮影したカフェテリア

図　4－2　実際に撮影した K 棟

図 4−3　実際に撮影した講堂

図 4−4　実際に撮影したドミトリー

図 4－5　実際に撮影した A 棟

図 4－6　実際に撮影した B 棟

これらの写真を MacromediaFLASH に取り込みます。

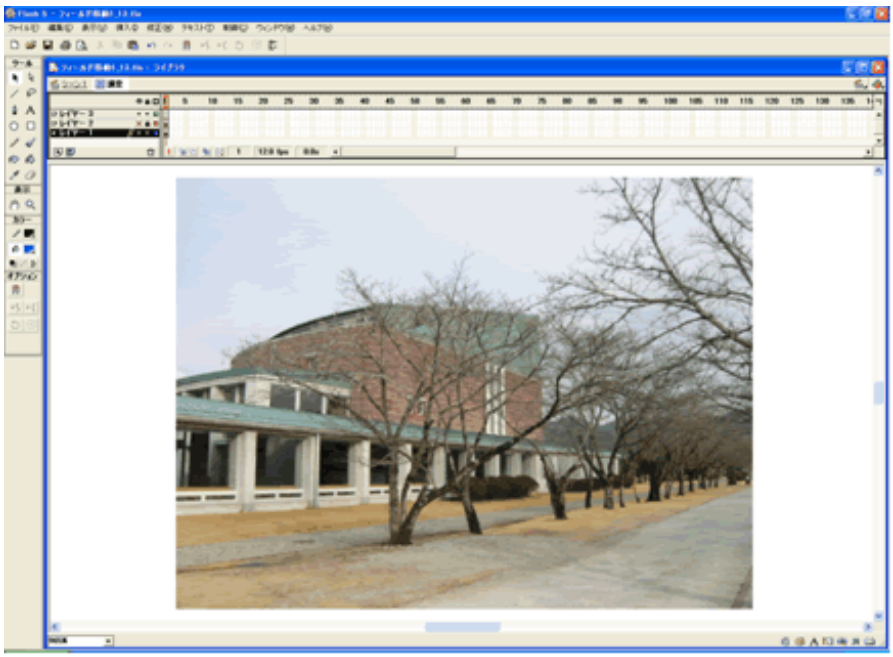

図 4－7　撮影した講堂の写真を MacromediaFLASH に“読み込む”を行った

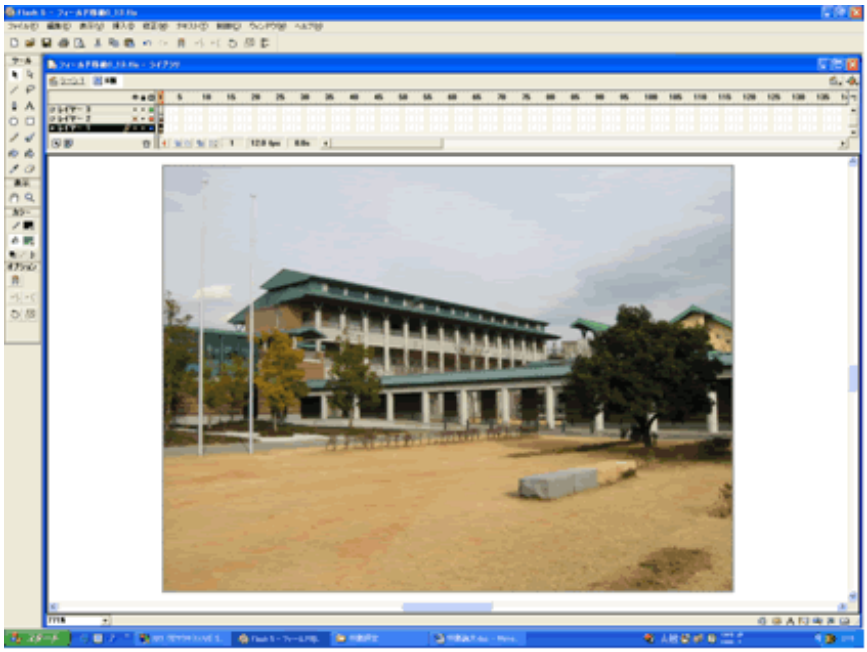

図 4－8　撮影した K 棟の写真を MacromediaFLASH に“読み込む”を行った

ここで FLASH のタイムラインの説明をしたいと思います。

図 4−9　講堂を読みこんだファイルのタイムライン

これは FLASH のタイムラインです。まず、この部品は“ シーン 1 ”の下階層にある“ K 棟 ”というムービークリップであることを表しています。また、レイヤー3、絵、写真という“ レイヤー ”があり“ 1 フレーム目 ”の絵レイヤー、写真レイヤーに“ オブジェクト ”が配置されています。そして、12 フレームが 1 秒間の速さで表示され、絵レイヤーは“ 非表示 ”になっています。鍵マークに×印があると“ 編集不可 ”、四角をクリックするとそのレイヤーに配置されたオブジェクトが線のみの表示になります。最後に、レイヤーは上にあるほど画面に表示される優先度があります。

レイヤーは上にあるほど画面に表示される優先度あるという機能を利用して写真を配置したレイヤーより上のレイヤーに建物の絵をタブレットで作成していきます。

図 4－10　実際に使用したタブレット

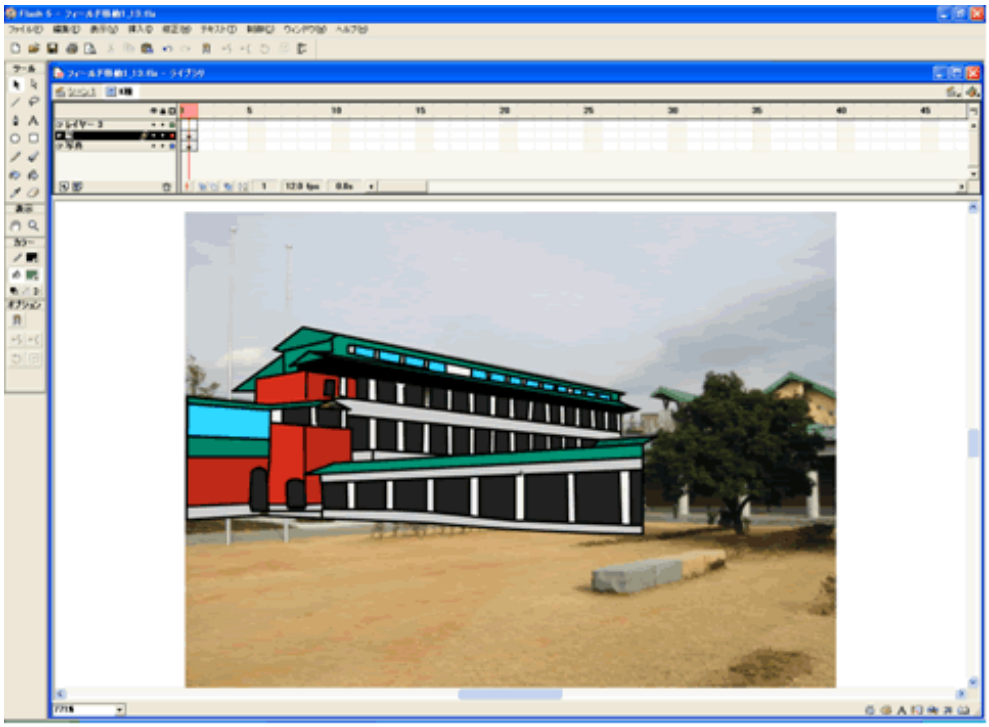

図 4－11　実際に描いた K 棟

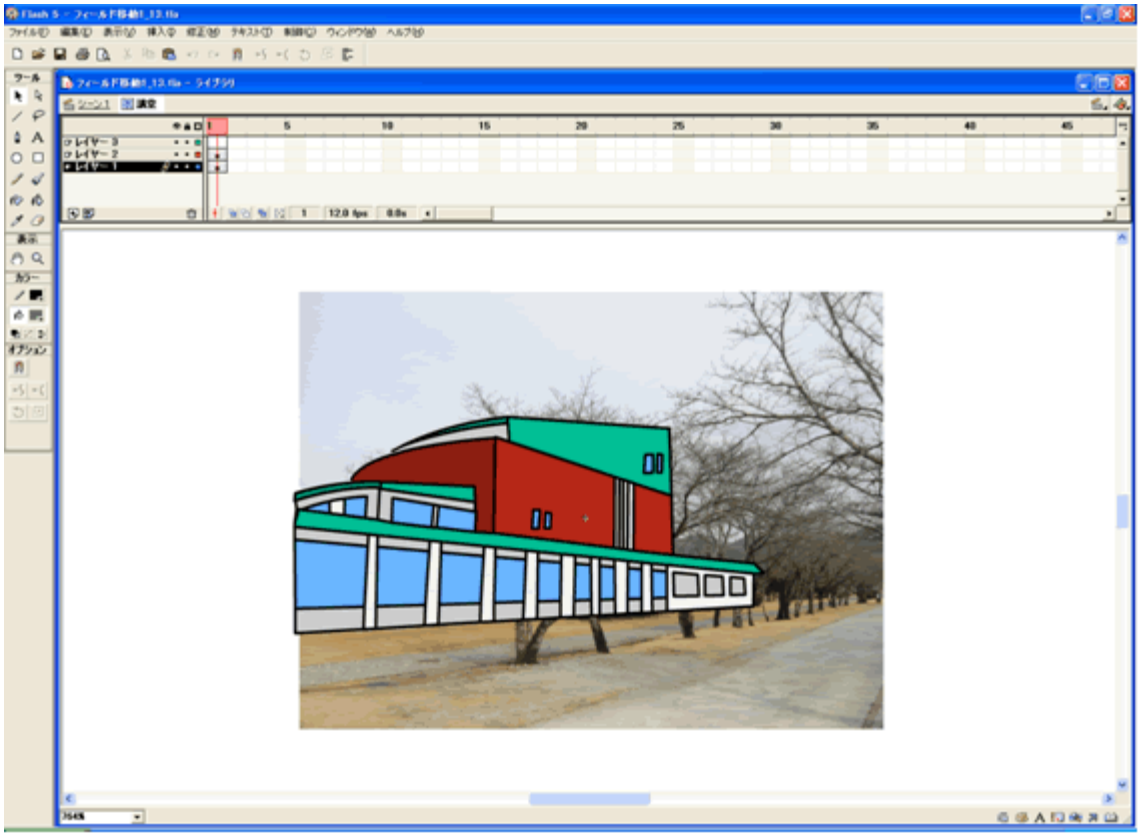

図 4－12　実際に描いた講堂

このようにパーツを作成していきます。

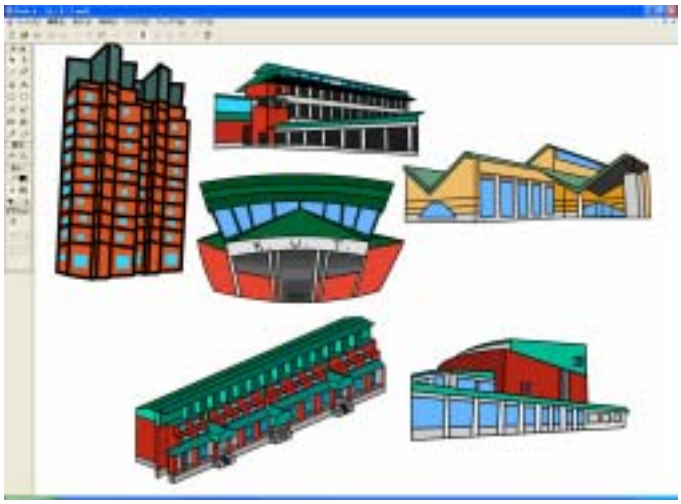

図 4－13　実際に作成したパーツたち

図 4－13 で作成したパーツを組み合わせて全体マップを作っていきます。

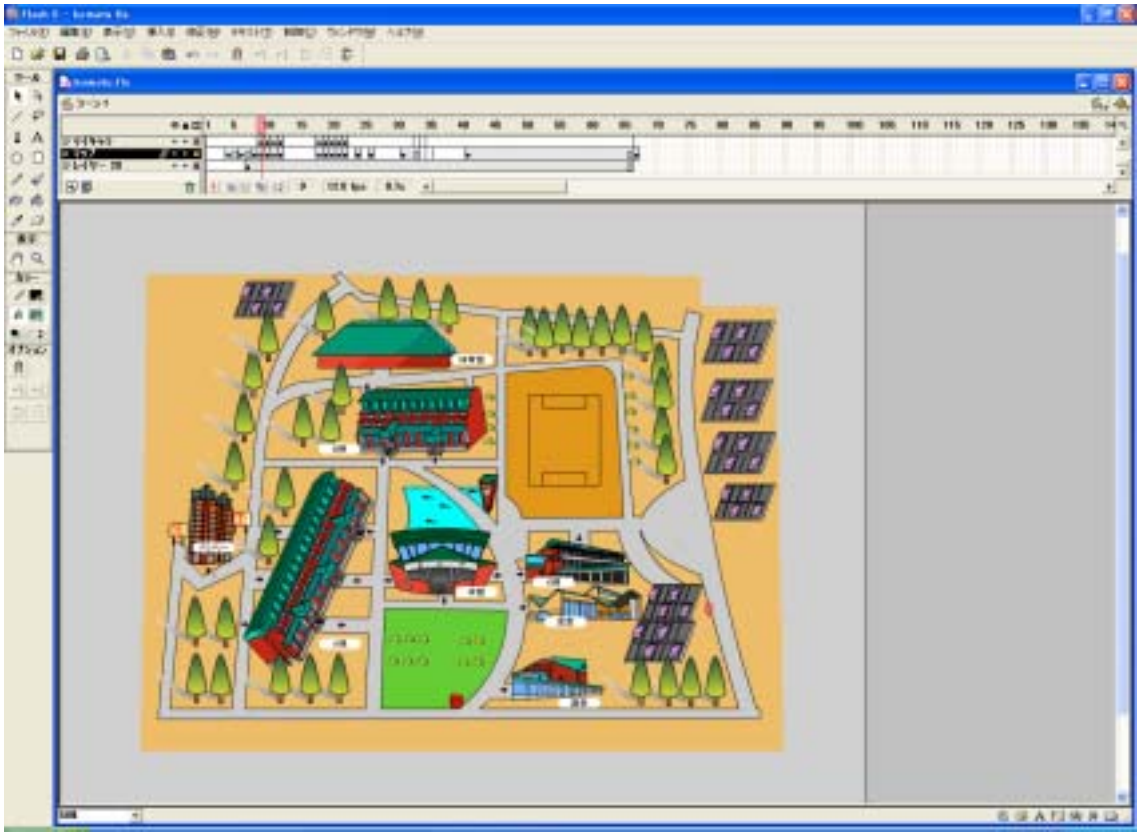

図 ４－14　完成した全体マップ

次に建物内の絵の作成に入ります。これも学校のところどころに配置してある学校の部屋案内を元に作成していきます。

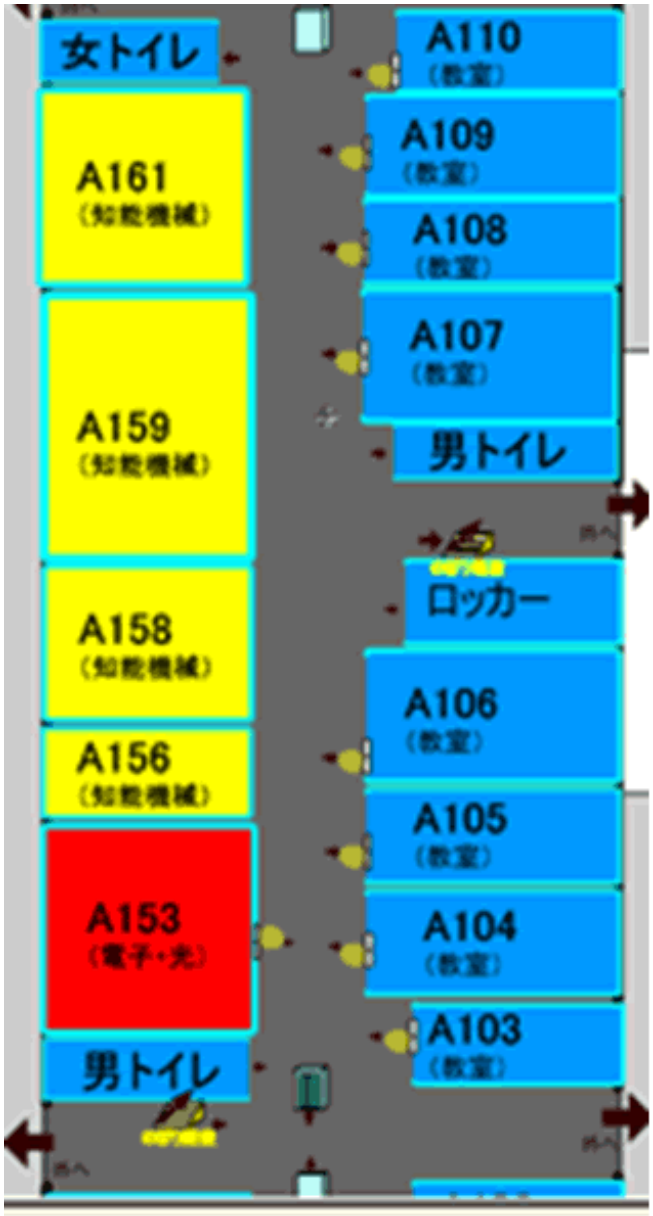

図 4－15　作成した A 棟 1 階内部

さらに完成した全体マップと内部マップの下レイヤーにイベントを制御したり、キャラクターの進入を阻むために使用するムービークリップを作成し、別々のインスタンス名をつけて配置していきます。
どのように制御するのかは ActionScript の説明に記します。

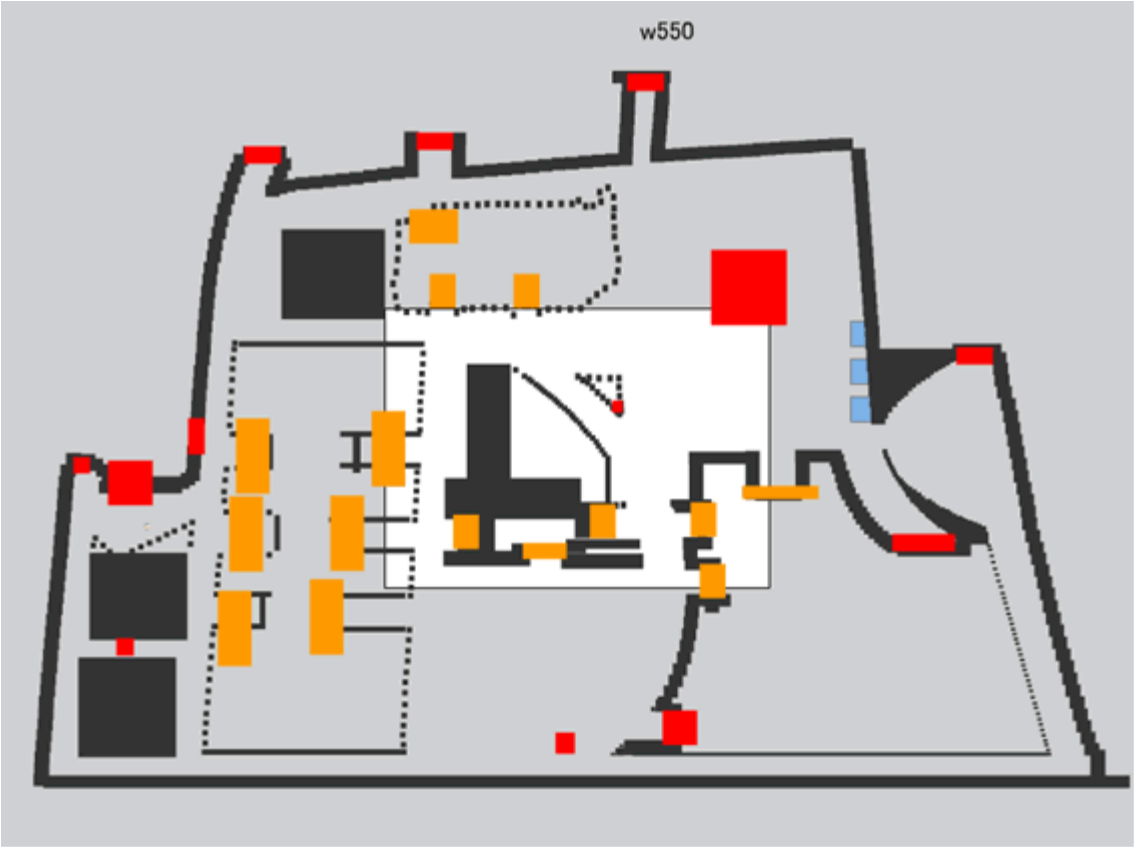

図 4－16　全体マップに“壁”＝黒、“イベント”＝赤、“別の場所に飛ぶ”＝オレンジを配置した全体マップ　壁は w1、w２と名前をつけていって w550 までいってしまいました。

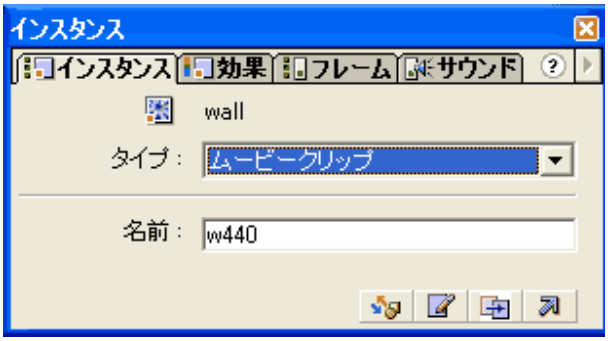

図 4－17　wall という名前でライブラリに登録されたムービークリップ。インスタンス名は w440 です。

ここで、FLASH で使用するシンボルについて説明します。

ムービークリップ…独自のタイムラインを持ちそれだけでアニメーションすることが可能でライブラリに登録できる。インスタンス名をつけて区別することができる。

ボタン　　…マウスでクリックしたり、上にカーソルを乗せたときの絵を描いて、ムービーを制御する時に使ったりします。マウスがヒットする範囲も指定できます。これにはインスタンス名はつけられませんし、ActionScript で制御をする必要があります。

グラフィック　…絵です。これそのものを制御することはできません。

　これらのオブジェクトをライブラリに登録しステージに配置されたものをインスタンスといい、ライブラリ内のシンボルを修正することですべてのシンボルに適用できるので作成の際に活用できます。

次に、主人公は歩く向きで違う絵が表示されるようにするために何パターンも絵を準備します。

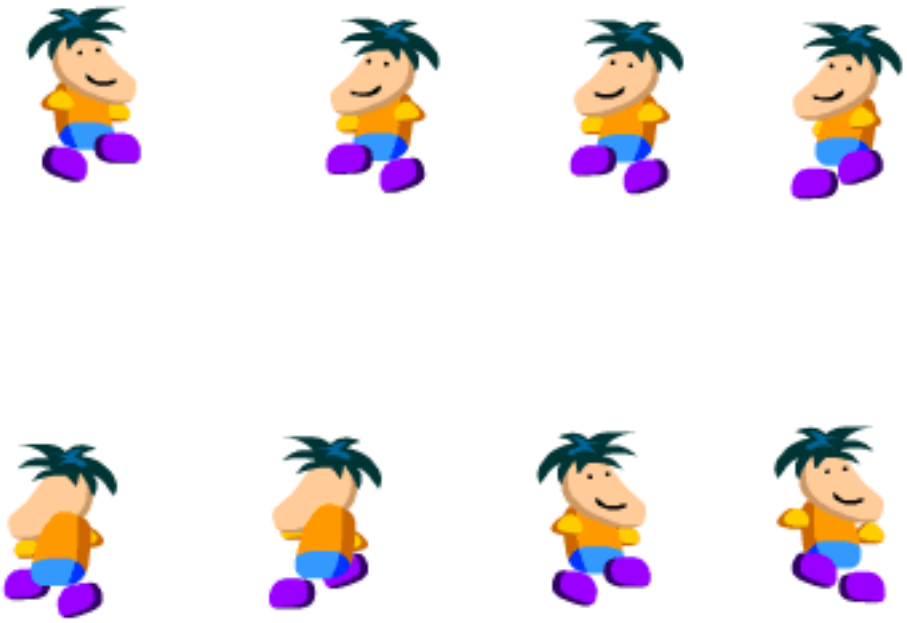

図 4－18　上の左二つは正面の時、右二つは左歩きの時、下の左二つは後ろに歩く時、下の右二つは右に歩く時

さらにバトルシーンで使用する敵の絵を作成していきます。
まず、主として動きのない敵の絵を作成しました。

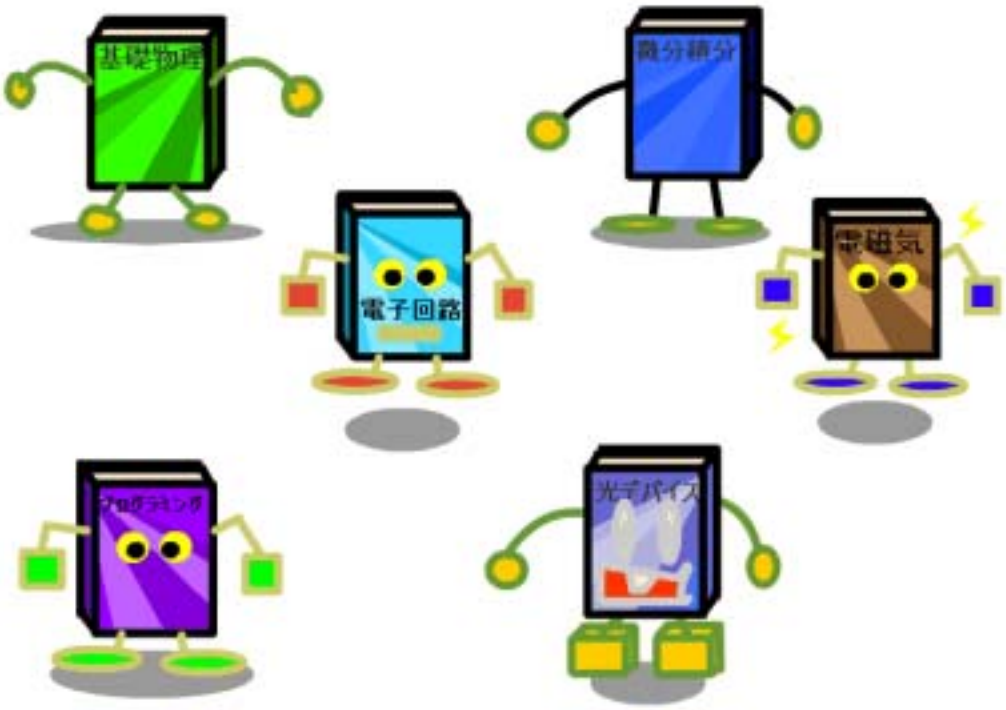

図 4－19　作成した敵その 1

図 4－20　作成した敵その 2

最後の敵の作成に入ります。ゲームの最後に出てくることもあり、動きのあるものにします。まず一枚の絵を作成します。

図 4－21 作成した 1 枚絵

これをさまざまなパーツに分けて“グループ”化させます。そうすることにより 1 枚の絵を別々に制御して動きをつけることができるようになります。

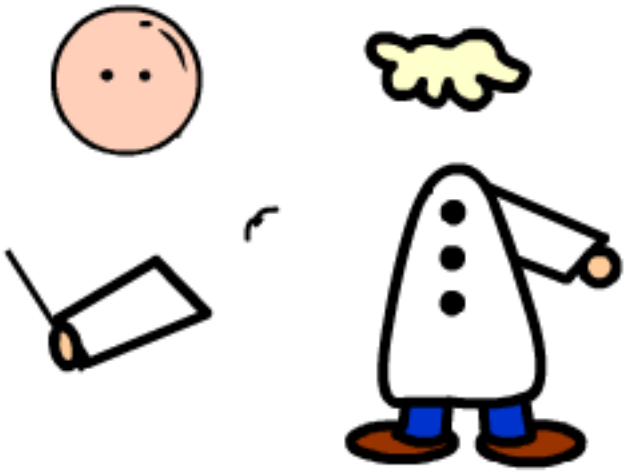

図 4－22　頭、右腕、髪、髭、その他にクループ化して分割しました

これをわずかに動かしながら動きをつけていきます。

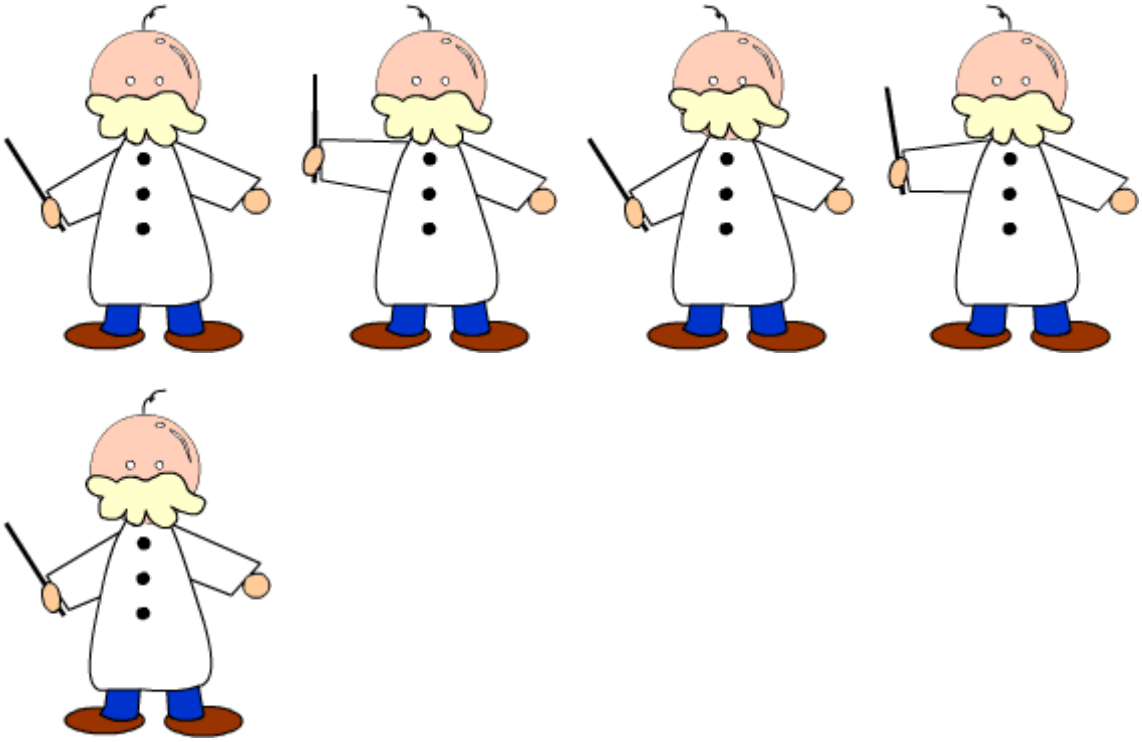

図 4−23　このように微妙にずらした絵を作ります

この絵をそれぞれのフレームに配置することで動きます。
これで敵キャラクターの作成は終わりです。

絵の作成の最終段階です。入ることができる部屋の内部を作成していきます。

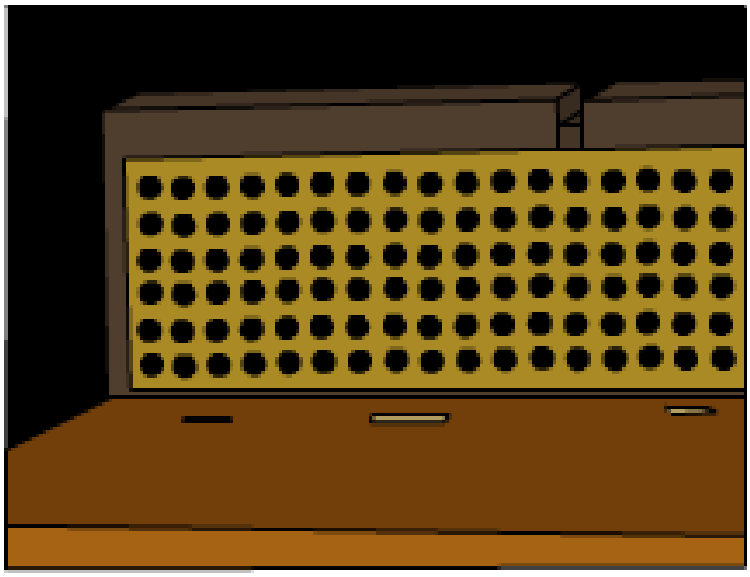

図 4－24　作成した図書館の LAN 端末のある机（実際には使用しませんでした。）

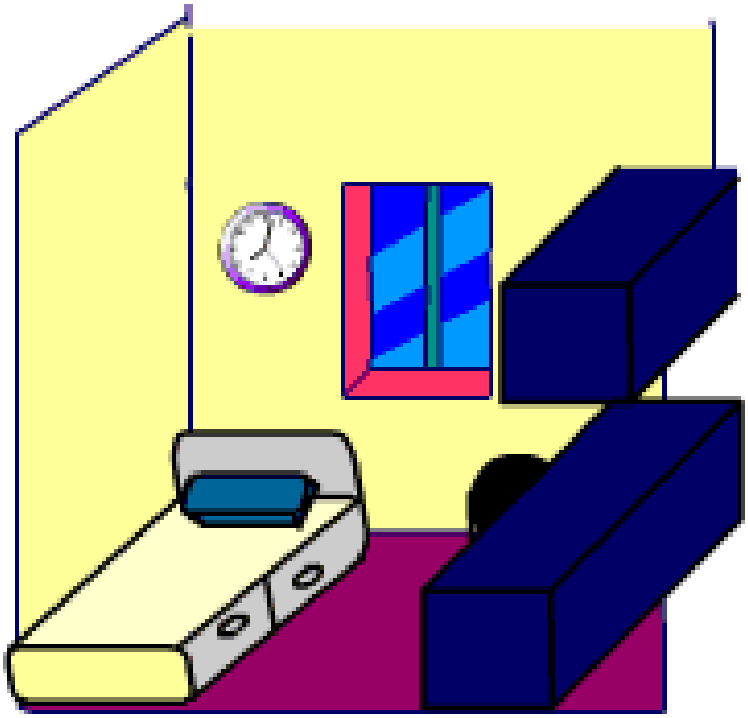

図 4－25　作成した主人公のドミトリーの部屋

とりあえず絵の作成が終わりました。

4－3　タイムラインとレイヤーの構成の概要

いよいよ、絵の作成も終わり、ゲームの根幹を決めていく作業に入ります。それはどのフレームでどんな作業を行わせていくのかを決める作業です。

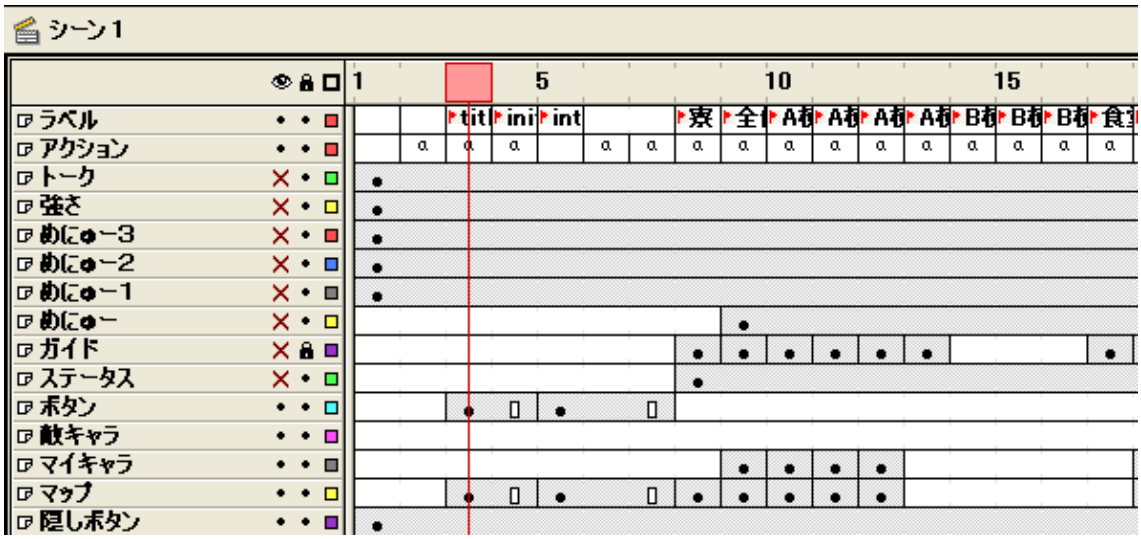

図 4－26　実際のゲームで作成したタイムラインその１

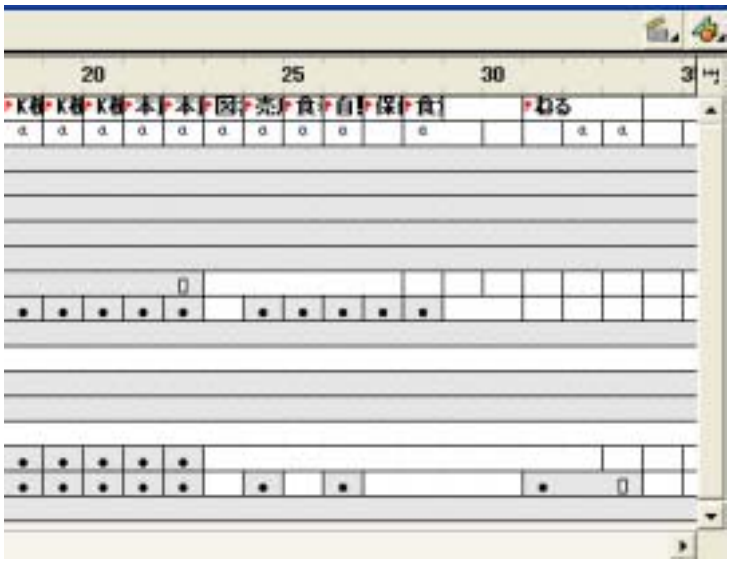

図 4－27　実際のゲームで作成したタイムラインその２

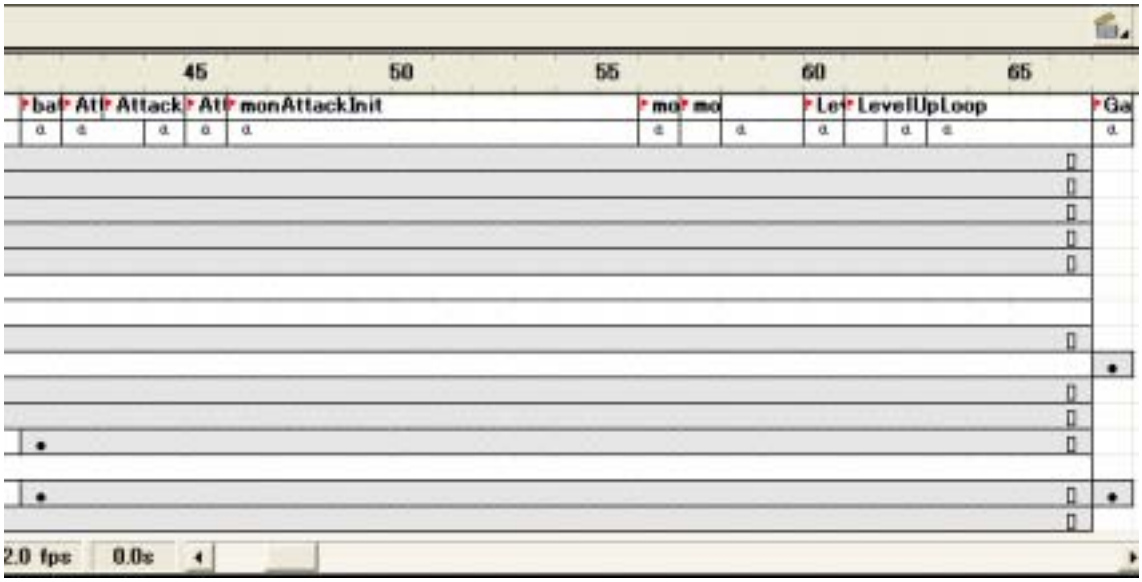

図 ４－28　実際のゲームで作成したタイムラインその 3

上の三つの図のように決めました。

・レイヤーの構成

ラベル…　フレームでやっている作業が製作中にわかるように書き出したレイヤー。ラベルをつけるとフレームを飛ばす ActionScript の目標にすることができ、便利である。

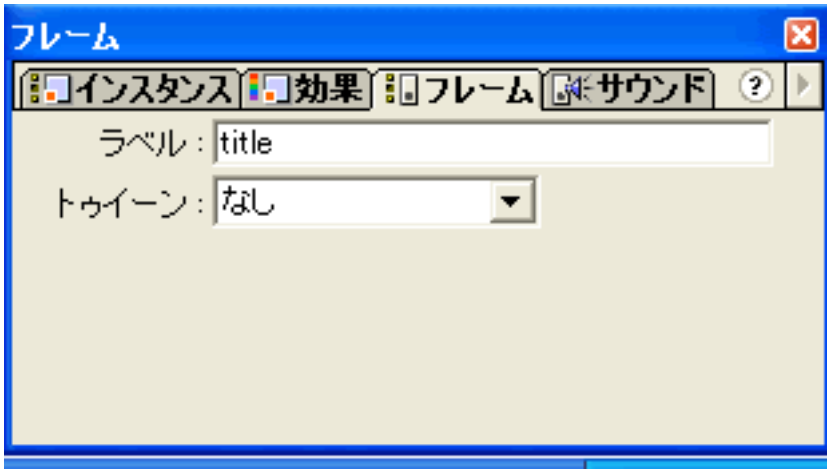

図 ４－29　ラベルを書く（3 フレーム目）

アクション…　それぞれのフレーム自体に ActionScript を書き込みたい時に使用します。

トーク…　テキストを表示させることを前提に作ったムービー DispTalk を配置したレイヤー。

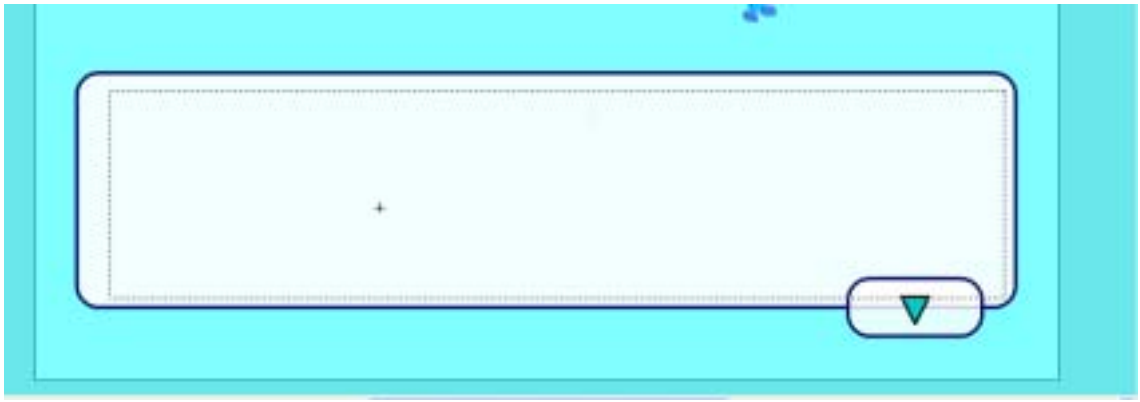

図 4－30　ムービークリップ DispTalk（透明の青色の部分はこのムービーの表示が消えるまでマウスクリックを防ぐために配置した“から”ボタン）

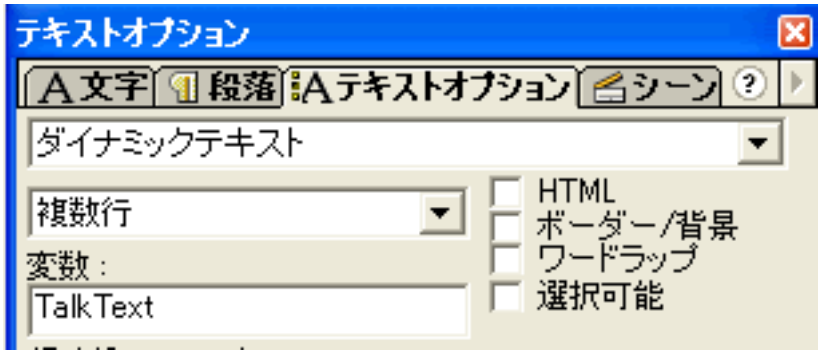

図 4－31　ムービークリップ DispTalk のテキストオプション

上の図 4－31 のようにテキストオプションでダイナミックテキストを選ぶとそのテキストフィールドに変数を置く事ができ、それに代入することで文字を表示することができます。実際にどのように変数に代入していくのかについては ActionScript の説明で追記します。

強さ…　主人公の詳細の能力を見ることができるムービーDisp つよさを配置したレイヤー。

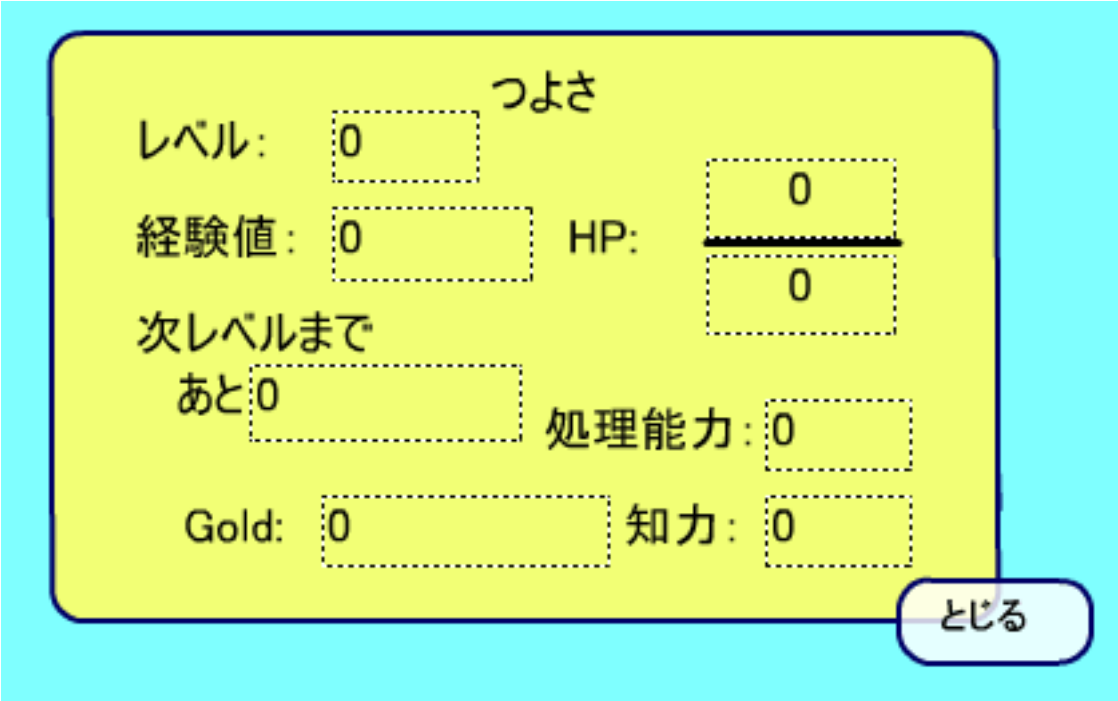

図 4−32　ムービークリップ Disp つよさ（透明の青色の部分は表示が消えるまでマウスクリックを防ぐために配置した“から”ボタン）

図 4−32 の Disp つよさではダイナミックテキストでレベルには Level、経験値には Exp、次のレベルまでには NextExp、Gold には Gold、HP の上には HP、下には MaxHP、処理能力には Attack、知力には Defence の変数が埋め込まれています。現在表示されている 0 は、仮に表示させているものです。

めにゅー１、２、３…　ゲームで行動を選択するメニューを表示するために用意したムービークリップ menu にそれぞれ menu1、menu2、menu3 とインスタンス名を付け配置したレイヤー。

図　４－33　memu1 とインスタンス名が付けられた menu

このムービークリップ menu は後に示す図4－34と図4－35のムービークリップ menuItem と図4－36のムービークリップ MenuBottom で構成されており、menuItem には上から 0,1,2,…,15 までのインスタンス名が与えられ、MenuBottom には MenuBottom というインスタンス名が与えられています。

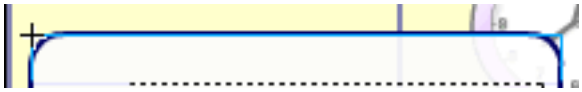

図 4－34　menu の一番上についている絵

図 4－35　menu の中に 16 個ある menuItem

図 4－36　menu の一番下にある MenuBottom

これらはメニューに表示されるべき項目の数にあわせて表示するための構成です。menu の整形の仕方は ActionScript の説明で追記します。
ちなみにムービークリップ menuItem はその中にムービークリップ mark とムービークリップ MarkArm とボタンが配置されています。十字キーの入力に対して mark が移動し、ボタンクリックに対して mark が点滅したり、マウスを上に乗せる（FLASH ではオーバーという）と表示が変わります。MarkArm はそこに表示されたもの（装備品）が装備状態にあれば表示されるマークです。マウスクリックで起こる現象は ActionScript で制御を行っています。ActionScript のくだりで説明します。

図 4－37 ムービークリップ mark

図 4－38　ムービークリップ MarkArm の 2 フレーム目に書かれた絵。1 フレーム目には何もありません。

めにゅー…　ムービークリップ menu の menu1 にマップマップ移動時のメニューを表示するボタンを配置したレイヤー。

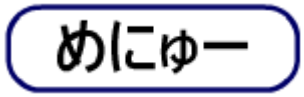

図 4－39　menuBtm

図 4－40　実際のゲームで menuBtm をクリックした図

ガイド…現在地を知らせる絵を配置したレイヤー。

図 4－41　ガイドレイヤーに配置されている絵の例

ステータス…強さレイヤーに配置されていたDispつよさの要点のみを表示したものを配置したレイヤー。

図 4－42　ムービークリップ DispStatus ダイナミックテキストを配置している

ボタン…ボタンとテキストが置かれているレイヤー。

敵キャラ…バトルシーンで使用する敵キャラクターを配置したレイヤー。絵の作成で作った敵の絵をひとつのムービークリップに配置し、制御が行いやすくしたものを配置してある。あと、戦闘で使用するエフェクトも配置した。

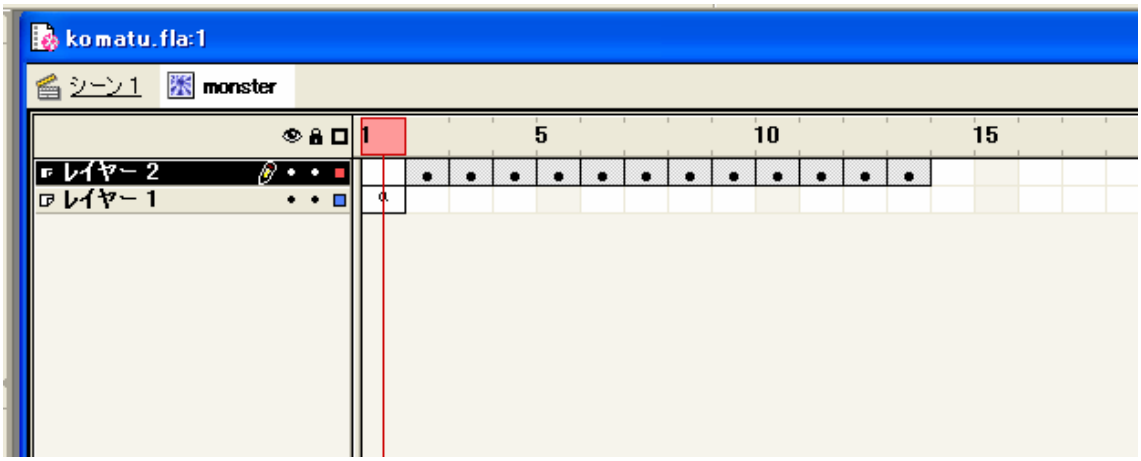

図 4−43　レイヤー１に ActionScript、レイヤー2 の 2〜13 フレームに敵の絵を配置しました

マイキャラ…マップを移動する際に表示する主人公のムービークリップを配置したレイヤー。各マップに対応した ActionScript を配置するためにマップごとに配置している。

マップ…　全体マップや建物内部マップだけでなくゲームに使用した背景を配置したレイヤー。

隠しボタン…ゲームの画面外に配置し、ゲームの操作性をよくするため、主にキーボードのキー入力に対応した ActionScript を書いてあるレイヤー。主に開発中に便利なので配置した。

・ フレームの構成

2 フレーム…主な関数定義と主人公と敵の能力の定義、道具の定義を行っているフレーム。

3 フレーム…ラベル“ title ”。タイトル画面の表示を行うフレーム。

4 フレーム…ラベル“ init ”。さまざまな初期値の代入を行うと共に menu に表示する配列を定義している。

5〜7 フレーム…ラベル“ intro ”。このゲームの設定の紹介を行うとともに自由に名前を入力できるようにしたフレーム。

8 フレーム…ラベル“ 寮 ”。寮の主人公の部屋を配置したフレーム。

9 フレーム…ラベル“ 全体 ”。本学の全体マップを配置したフレーム。

10〜28 フレーム…上二つと同じように学校の内部の絵やムービークリップを配置したフレーム。（構想段階で設定した構成なので実際には何もしていないフレームも存在します。）

31〜33 フレーム…ラベル“ ねる ”。寮内で“ ひとやすみ ”を選んだ時に表示する睡眠の絵を配置したフレーム。

41 フレーム…ラベル“ battle ”。ここでバトルシーンに現れる敵を表示させているフレーム。また、前回のバトルで得た経験値を初期化している。

42 フレーム…ラベル“ AttackMenu ”。主人公のバトルメニューを表示させているフレーム。

43、44 フレーム…ラベル“ AttackLoop ”。もし、トーク部分が表示されていたらここでループさせているフレーム。ここでプレイヤーが道具を使った場合のテキスト読みを待つ。

45 フレーム…ラベル“ AttackExec ”。ここで主人公が攻撃した場合のダメージや発言を決めているフレーム。また、主人公が勝利した場合ここでバトルが終わるためレベルアップの処理を行う。

46〜55 フレーム…“ monAttackInit ”。モンスターの攻撃順番の初期化、ここで主人公がガード行動の初期化を行っているフレーム。またバトルで起こるエフェクトを待つ役目もある。

56 フレーム…“ monAttack ”。モンスターの攻撃ダメージと発言を決定しているフレーム。また、主人公が倒れた場合ここで戦闘が終わるのでその処理も行っている。

57 フレーム…“ monAttackLoop ”。実際にはここでは何も行っていないがもし 58 フレーム目に行ってもトーク部分が表示されていたらここで消されるのを待つフレーム。

58 フレーム…ここでモンスターのすべての攻撃が終わっているのかチェックをし、終わっていないなら“ monAttack ”に飛ばしている。もし終わっているならば“ AttackMenu ”に飛ばしている。

60 フレーム…“ LenelUp ”。もし戦闘が終わった時にレベルアップした時にここに飛ばされるフレーム。ここでトーク部分にテキストを入力し表示している。

61、62 フレーム…“ LevelUpLoop ”。テキスト表示が消えるのを待つフレーム。

63 フレーム…戦闘が発生したフレームに戻す役割を担うフレーム。

4－4 ゲーム中に使用した ActionScript の説明（一部のみ）

前のページまでで説明したように制御するために ActionScript を配置していきます。その例を説明します。

・ フレームアクション（フレームに書き込んである ActionScript）

2 フレーム目

```
function RefreshDispStatus () {   //RefreshDispStatus と言う関数を定義
        _root.DispStatus.Level = _root.myStatus.Level;
            //DispStatus.Level に myStatus.Level を代入
        _root.DispStatus.HP = _root.myStatus.HP;
            //DispStatus.HP に myStatus.HP を代入
        _root.DispStatus.MaxHP = _root.myStatus.MaxHP;
            //DispStatus.MaxHP に myStatasu.MaxHP を代入
        _root.DispStatus.Gold = _root.myStatus.Gold;
            //DispStatus.Gold に myStatu.Gold を代入
}           //インスタンス名 DispStatus の更新を行う

function RefreshTsuyosa ()   //インスタンス名 Tsuyosa を更新する関数

function RefreshItemMenu () {   //RefreshItemMenu という関数を定義
        itemMenu = new Array();   //itemMenu という配列を定義
        for (i=0; i<item.length; i++) {
//i を０から item.length（所持アイテムの個数）まで増やしながら繰り返す
                itemMenu[i] = new Object();
          //itemMenu は新たなオブジェクト
                itemMenu[i].No = item[i].No;
          //itemMenu[i].No に item[i].No を代入
                itemMenu[i].Name = item[i].Name;
          //itemMenu[i].Name に item[i].Name を代入
                itemMenu[i].ExecNo = item[i].ExecNo;
          //itemMenu[i].ExecNo に item[i].ExecNo を代入
        }
        itemMenu[i] = new Object();
        itemMenu[i].Name = "やめる";   //itemMenu の最後にやめるを入れる
        itemMenu[i].ExecNo = 10;
}
```

```
function AddItem (ItemNo)　//道具が増えた時に登録する関数

function RefreshMonsterMenu ()　//戦闘で敵の整理を行う関数

function Dice1 (DiceNo) {　//Dice１という関数を定義
        D = 0;　//変数 D を０とし
        for (i=0; i<DiceNo; i++) {　//i を DiceNo まで増加させながら繰り返す
                D += Math.floor(1+Math.random()*6);
            // D にランダムで 1 から 6 まで足す
        }
}

function Dice2 (DiceNo)　//Dice１と違うのは変数を E にしていること

function fine2()　//D を二倍にする関数

function AddMonster (MonsterNo)　// MonsterNo の敵を追加する関数

LevelData = new Array();　//主人公のレベルデータを定義した配列
for (i=0; i<=10; i++) {
        LevelData[i] = new Object();
}
LevelData[0].Attack = 10;
LevelData[0].DX = 11;
LevelData[0].Defence = 9;
LevelData[0].GP = 5;
LevelData[0].MaxHP = 100;
LevelData[0].AP = 16;
LevelData[0].NextExp = 400;
```

```
itemData = new Array();　//道具のデータを定義した配列
for (i=0; i<=8; i++) {
	itemData[i] = new Object();
}
itemData[0].Name = "パックジュース";
itemData[0].AttackValue = 0;
itemData[0].HPValue = 50;
itemData[0].ExecNo = 12;
itemData[0].Price = 100;
itemData[0].Info = "これでも飲んで一息ついて"+String.fromCharCode(13)+
"HP を 50 回復"+String.fromCharCode(13)+"賞味期限に気をつけて";
monsterData = new Array();　//敵のデータを定義した配列
for (i=0; i<=11; i++) {
	monsterData[i] = new Object();
}
monsterData[0].Name = "基礎物理学";
monsterData[0].HP = 20;
monsterData[0].Attack = 8;
monsterData[0].Defence = 9;
monsterData[0].Gold = 200;
monsterData[0].Exp = 72;
```

4 フレーム目

```
centerX = 275;          //ステージセンター座標に主人公を配置するために定義
centerY = 200;
myX = 280;              //主人公の相対座標
myY = 870;
_root.moveFlag = true;      //主人公の移動の許可を出すフラグを定義
//-----------------------------------
myName = "道夫";  // デフォルトの主人公の名前を道夫にする
MoveDist = myStatus.AP;   //仮に主人公の十字キーによる移動量を定義
maxMenu = 16;        // menu の最大数を 16 に決める
houkou=10;              //主人公歩く方向にムービーを合わせるために変数を定義
//-----------------------------------
myStatus = new Object();   //主人公の能力を新たに独自のオブジェクトとし
myStatus.Level = 1;   //主人公のレベル１の能力値を代入する
myStatus.MaxHP = LevelData[0].MaxHP;
myStatus.HP = myStatus.MaxHP;
myStatus.Attack = LevelData[0].Attack;
myStatus.Defence = LevelData[0].Defence;
myStatus.DX = LevelData[0].DX;
myStatus.GP= LevelData[0].GP;
myStatus.AP = LevelData[0].AP;
myStatus.IQ = LevelData[0].Defence;
myStatus.ST = LevelData[0].Attack;
myStatus.Exp = 0;
myStatus.Arms = 0;
myStatus.Gold = 300;

        eventFlag = new Array();   //イベント制御用のフラグを配列で定義
        for (i = 0; i <= 100; i++) {   //101 個のフラグを false で大量生産
                eventFlag[i] = new Object();
        }
        for (i = 0; i <= 100; i++) {
                eventFlag[i] = false;
        }
```

```
monsterArea = 1;  //戦闘で敵の配置に使う変数を仮に定義
eventA204Flag = false;  //イベント用のフラグ
eventA2Flag = false;
eventA4Flag = false;
_root.monsterMode = false;  //戦闘シーンを判別
_root.MeetMonster = 100 + Math.floor(Math.random()*20);
            //敵との遭遇をランダムに設定
SO=100;  //イベント用変数
//  ExecNo  /この番号でマウスクリックの処理を行っているためこれに合わせてマウスクリック処理があるところに ExecNo を与えていく
// 1:「どうぐ」Menu2 にどうぐメニュー出す
// 2:「じょうたい」Menu2 にじょうたいメニュー出す
// 3:「こうげき」Menu2 に敵リスト出す
// 4:「つよさをみる」つよさを表示
// 5:「どうぐせいり」Item をならべかえ
// 6:「つかう」
// 7:「そうび」
// 8:「みる」
// 9:「すてる」
//10:「やめる」
//11:「アイテム 1 を選択」Menu3 にどうぐメニュー１を出す
//12:「アイテム 2 を選択」Menu3 にどうぐメニュー２を出す
//13:「アイテム 3 を選択」Menu3 にどうぐメニュー３を出す
//14:「こうげき相手選択」こうげき
//15:「やめる」メニューを閉じる
//16:「かいにきた」
//17:「うりにきた」
//18:「やめる」
//21:「アクセスアイテム」
//22:「自販機メニュー」
//23:「にげる」(戦闘から)
//24:「勉強」Menu2 に勉強リストをだす
//25:勉強内容を選択、バトルモードにいく。
//26:買いに来た自動販売機メニュー
//27:「一休み」HP の回復をする
```

```
//-----------------------------------トップメニュー１
	cmd1 = new Array();　//このように配列を定義し
	for (i = 0; i <= 2; i++) {
		cmd1[i] = new Object();　//独自のオブジェクトにして
	}
	cmd1[0].Name = "どうぐ";　//名前と
	cmd1[0].ExecNo = 1;　//ExecNo を与えていく
	cmd1[1].Name = "じょうたい";
	cmd1[1].ExecNo = 2;
	cmd1[2].Name = "やめる";
	cmd1[2].ExecNo = 15;
//-----------------------------------トップメニュー２
	cmd2 = new Array();
	for (i = 0; i <= 3; i++) {
		cmd2[i] = new Object();
	}
	cmd2[0].Name = "こうげき";
	cmd2[0].ExecNo = 3;
	cmd2[1].Name = "どうぐ";
	cmd2[1].ExecNo = 1;
	cmd2[2].Name = "じょうたい";
	cmd2[2].ExecNo = 2;
	cmd2[3].Name = "にげる";
	cmd2[3].ExecNo = 23;

//-----------------------------------アイテム
	item = new Array();　//主人公道具を配列として定義
//-----------------------------------
	_root.NowMenu = "menu0";　//menu の表示状態を見るために定義
```

8 フレーム目

```
myX = 280;  //主人公の相対位置座標の定義
myY = 870;
if(SO eq 1000){  //SO=1000 は卒論が一応書きあがった状態のこと
_root.DispTalk.TalkText = "卒論は一応書きあがったぞ！"+
String.fromCharCode(13);  //テキストを入力して改行を行う
_root.DispTalk.TalkText += "A 棟の 4 階、電子・光システム工学科にいる先生
に見せに行こう。"+ String.fromCharCode(13);
_root.DispTalk.TalkText += "最終決戦の予感！！";
                        _root.DispTalk._visible = true;
}

_root.monsterArea = 4;  //ここで戦闘を行うとエリア４
if(eventFlag[99] eq false){
_root.DispTalk._visible = true;
eventFlag[99] = true;
}
_root.returnFrame =("全体")
  //ここのフレームから全体マップに帰れるように定義する
//------------------------------
for (i=0; i<maxMenu; i++) {
        eval("_root.menu1."+i).gotoAndStop(1);
}  //メニューにつく三角（図 4－37）を消す
_root.moveFlag = false;  //主人公が動けないように定義
_root.menu1.cmdName = "cmd3";
  //menu1 にフレーム 4 で定義した cmd3 を代入
_root.menu1.MenuDisp();  //menu1 を整形する
_root.menu1._visible = true;  //menu1 を見えるようにする
_root.NowMenu = "menu1";  //いま表示しているのは menu1
RefreshDispStatus();  //DispStatu を更新
stop();  //このフレームで再生をとめる
```

9 フレーム目（主人公が移動するマップの代表例）

```
_root.mychar._x=centerX;　//主人公の位置を中央に配置して
_root.mychar._y=centerY;
_root.map_zentai._x=centerX-myX;　//それぞれのマップの座標を決定
_root.map_zentai._y=centerY-myY;

stop();　//このフレームでとめる
```

このように移動マップではマップのインスタンス名の座標を操って主人公を真ん中に表示しています。

24 フレーム目（買い物の代表例）

```
_root.DispTalk.TalkText = "いらっしゃいませ。" + String.fromCharCode(13);
_root.DispTalk.TalkText += "なにをお求めですか。";
_root.DispTalk._visible = true;　//　このようにテキストを表示する
//------------------------------
for (i=0; i<maxMenu; i++) {
        eval("_root.menu1."+i).gotoAndStop(1);
}
_root.moveFlag = false;
_root.menu1.cmdName = "shopmenu1";
  // ショップ用の配列を menu1 に入れる
_root.menu1.MenuDisp();
_root.menu1._visible = true;
_root.NowMenu = "menu1";
RefreshDispStatus();
stop();
```

41 フレーム目（バトルシーンの最初のフレーム）

```
Exp=0;  //戦闘で得られる経験値の初期化
//------------------------------
for (i=0; i<maxMenu; i++) {
        eval("_root.menu1."+i).gotoAndStop(1);
}  //三角印を消して
_root.moveFlag = false;  //主人公を歩けなくする
_root.menu1.cmdName = "cmd2";        //戦闘用のコマンド配列を代入
_root.menu1.MenuDisp();
_root.menu1._visible = true;
_root.NowMenu = "menu1";
RefreshDispStatus();
stop();
//
_root.monsterMode = true;  //敵との戦闘モードにはいる
monster = new Array();  //monster を配列で定義して
rnd = Math.floor (Math.random() * 15);  //遭遇した敵を決定するランダム数
if (monsterArea == 1) {// 1 基礎物理学、2 微積分学 、3 プログラミング
        if (rnd < 4) {
                AddMonster (1);
                AddMonster (1);
        } else if (rnd < 9) {
                AddMonster (1);
                AddMonster (1);
                AddMonster (2);
        }   else if (rnd < 13) {
                AddMonster (1);
                AddMonster (2);
                AddMonster (3);
        }else {
                AddMonster (2);
                AddMonster (2);
                AddMonster (3);
        }
}  //このように敵を配置していく
```

```
else if (monsterArea == 4) {      // 11 卒論
   //これはドミトリーで勉強を行った場合に用意したもの
          if (_root.Selectmon eq 0) {
                    AddMonster (1);
          } else if (_root.Selectmon eq 1) {
          AddMonster (2);
          } else if (_root.Selectmon eq 2) {
          AddMonster (3);
          } else if (_root.Selectmon eq 3) {
          AddMonster (4);
          }else if (_root.Selectmon eq 4) {
          AddMonster (5);
          }else if (_root.Selectmon eq 5) {
          AddMonster (6);
          }else if (_root.Selectmon eq 6) {
          AddMonster (7);
          }else if (_root.Selectmon eq 7) {
          AddMonster (8);
          }else if (_root.Selectmon eq 8) {
          AddMonster (9);
          }else if (_root.Selectmon eq 9) {
          AddMonster (10);
          }else if (_root.Selectmon eq 10) {
          AddMonster (11);
          }
}   //ドミトリー内で勉強した場合敵を選択できるのでどの敵を選択したのかを
_root.Selectmon を参照することで実現している
```

42 フレーム目（主人公の攻撃部分の最初）

```
// ----------------------------------------
for (i=0; i<maxMenu; i++) {
        eval("_root.menu1."+i).gotoAndStop(1);
}
_root.menu1.cmdName = "cmd2";  //戦闘用のコマンド配列を menu1 に代入
_root.menu1.MenuDisp();
_root.menu1._visible = true;
_root.NowMenu = "menu1";  //現在表示しているのは menu1
kaisin=0;  //改心の一撃が出たかどうかを判定する変数を初期化
// ----------------------------------------
stop();
```

44 フレーム目

```
if (_root.DispTalk._visible eq true) {  //テキストが表示されていれば
        gotoAndPlay ("AttackLoop");
}  //43 フレームへ戻りこの仕組みで道具を使ってテキストが表示された場合、
```

プレイヤーが読み終わってテキストを消すのを待っている

45 フレーム目

```
if (_root.AttackMonsterNo == -1) {
  //－1 なのは道具で攻撃した時だが攻撃用道具はないので
} else {
        if (_root.myStatus.Attack+_root.myStatus.Arms eq 8) {
                Dice1(1);
                D += 6;
        } else if (_root.myStatus.Attack+_root.myStatus.Arms eq 9) {
                Dice1(1);
                D += 7;
        } else if (_root.myStatus.Attack+_root.myStatus.Arms eq 10) {
                Dice1(2);
                D += 8;
        } else
  //このような形でダメージの基礎値を決めていく
        if (_root.myStatus.Arms eq 2) {   // myStatus.Arms とは武器の攻撃力
                fine2();   //修正×２
        }   //装備している武器によるダメージの補正をする

Dice2(3);   ランダムで数を決めて
        if (E<_root.myStatus.DX-10) {
  //主人公の myStatus.DX から 10 引いた数より小さければ
 fine3();   //改心の一撃としてダメージ三倍
              kaisin=1;   //改心の一撃が出たので変数を 1 にする
        }

Attack = D-monster[_root.AttackMonsterNo].Defence;
  //このように最終ダメージを計算して Attack に代入する


        if (0<Attack) {   //Attack が 0 以上なら
                monster[_root.AttackMonsterNo].HP -= Attack;
  //敵の HP を減らす
                Mname = "_root.monsterHit"+_root.AttackMonsterNo;
                eval(Mname).gotoAndPlay(2);
 //敵にダメージエフェクトを再生させる
```

```
if (monster[_root.AttackMonsterNo].HP>0){
   //なおかつ敵が倒れていなければ
if (_root.monsterArea == 5) { //なおかつエリアが５なら
                    _root.DispTalk.TalkText = _root.myName+"は文章を直した。
"+String.fromCharCode(13);
if(kaisin eq 1){     //もし会心の一撃が出ていたら
                _root.DispTalk.TalkText  += "久しぶりに脳の回路がつながった。
"+String.fromCharCode(13);
                    }   //と表示する
                    _root.DispTalk.TalkText  += Attack+"のダメージをあたえた。
"+String.fromCharCode(13);
                    _root.DispTalk.TalkText += "卒論は完成に近づいた。";
                    _root.DispTalk._visible = true;
            } else{
   //このように表示しそうでなければ
                  _root.DispTalk.TalkText = _root.myName+"は勉強をしてみた。
"+String.fromCharCode(13);   //と表示する
if(kaisin eq 1){   //もし会心の一撃が出ていたら
                _root.DispTalk.TalkText  += "久しぶりに脳の回路がつながった。
"+String.fromCharCode(13);
                    }   //このように表示し
                    _root.DispTalk.TalkText  += Attack+"のダメージをあたえた。
"+String.fromCharCode(13);
                    _root.DispTalk._visible = true;
                }   //ダメージを表示する
        }
        }else{   //ダメージがなかったら
                    _root.DispTalk.TalkText =_root.myName+"は勉強をしてみた。
"+String.fromCharCode(13);
                    _root.DispTalk.TalkText  += "”ふり”であることがばれた。
"+String.fromCharCode(13);
                    _root.DispTalk._visible = true;
}
```

```
if (monster[_root.AttackMonsterNo].HP<=0) {    //もし敵が倒れたら
        if ((_root.monsterArea == 4) && (_root.Selectmon eq 10)) {
                _root.SO = 1000;
                myStatus.Gold += monster[_root.AttackMonsterNo].Gold;
                        Mname = "_root.monster"+_root.AttackMonsterNo;
                        eval(Mname).gotoAndStop(1);
                        _root.MonsterNum--;
                        Exp += monster[_root.AttackMonsterNo].Exp;
   //Exp に加算していく
                        RefreshDispStatus();
                        _root.DispTalk.TalkText = "なんとなんと
"+monster[_root.AttackMonsterNo].Name+"が一区切りついた。
"+String.fromCharCode(13);
                        _root.DispTalk.TalkText+=
monster[_root.AttackMonsterNo].Gold+"Gold 獲得！";
                        _root.DispTalk.TalkText += "あとは先生に提出しよ
う！";
                        _root.DispTalk._visible = true;
                }
   //このようにフラグたてや、経験値の加算、お金を足していく
if (_root.MonsterNum<=0) { //敵がいなくなったら
        _root.myStatus.Exp += Exp;    //この戦闘で得た経験値を加算
        _root.monsterMode = false;    //戦闘モードを終了
        if (LevelData[myStatus.Level-1].NextExp<=myStatus.Exp) {
myStatus.MaxHP = LevelData[myStatus.Level].MaxHP;
myStatus.Attack = LevelData[myStatus.Level].Attack;
myStatus.Defence = LevelData[myStatus.Level].Defence;
myStatus.DX = LevelData[myStatus.Level].DX;
myStatus.IQ = LevelData[myStatus.Level].IQ;
myStatus.AP = LevelData[myStatus.Level].AP;
myStatus.GP = LevelData[myStatus.Level].GP;
myStatus.Level++;
gotoAndPlay ("LevelUp");
myStatus.HP = myStatus.MaxHP;
RefreshDispStatus();    //レベルアップ処理をしてLevelUpへ飛ばす
```

```
        } else {   //そうでなければ
                gotoAndPlay (_root.returnFrame);
        }   //戦闘が起こる前に戻る
}

    45 フレーム
_root.menu1._visible = false;   //menu1 を消して
AttackMonNo = 0;   //１人目の敵の攻撃から
Ga=0;   //主人公のガードが発動したか見る変数初期化

     56 フレーム
if (0<monster[AttackMonNo].HP) {   //敵の HP が 0 より大きければ
//--------------------------------攻撃ダメージ
if(monster[AttackMonNo].Attack eq 8){
Dice1(1);
D+=6;
}else   //このように主人公と同じように基礎ダメージを決めていく

        Dice2(3);   ランダムで 3～18
        if (E<Math.floor(_root.myStatus.AP*0.4)) {   //この条件を満たせば
                        D-=_root.myStatus.GP;   //GP ぶんダメージ軽減
            Ga=1;   //ガードしたので 1 にする
        }
        Attack = D-_root.myStatus.Defence;   //最終ダメージの決定
        if (0<Attack) {
                _root.myStatus.HP -= Attack;   //HP から引く
                if (_root.myStatus.HP<0) {
                        _root.myStatus.HP = 0;   //0 以下になったら 0 にする
                }
            Exp +=1;   //敵が攻撃するごとに経験値を 1 ずつ足していく
                _root.RefreshDispStatus();
                _root.quake.gotoAndPlay(2);
                _root.DispTalk.TalkText = monster[AttackMonNo].Name+"の
攻撃。"+String.fromCharCode(13);
```

```
//--------------------------------モンスターのしゃべり
                    Dice2(1);   //ランダム変数 E1～6
                if(monsterArea eq 5){
                    if(E eq 1){
                            _root.DispTalk.TalkText += "「ほれほれ、誤字脱字ばかりだぞ。」"+String.fromCharCode(13);
                    }else if(E eq 2){
                            _root.DispTalk.TalkText += "「おつむが足りないんじゃないのか？」"+String.fromCharCode(13);
                    }else if(E eq 3){
                            _root.DispTalk.TalkText += "「君はこんな内容でふざけているんですか？」"+String.fromCharCode(13);
                    }else if(E eq 4){
                            _root.DispTalk.TalkText += "「もう 1 年来たらどうなんだ？」"+String.fromCharCode(13);
                    }else if(E eq 5){
                            _root.DispTalk.TalkText += "「体調が悪いなら、2、3 カ月入院したらどうだ？」"+String.fromCharCode(13);
                    }else{
                            _root.DispTalk.TalkText += "「そんなんじゃ卒論は仕上がらんな。」"+String.fromCharCode(13);
                    }
                }   //というように変数によってせりふを決めていく

if(Ga eq 1){   //ガードに成功していれば
                _root.DispTalk.TalkText  +=  "意地と根性で攻撃を防いだ。"+String.fromCharCode(13);
                }
if (_root.myStatus.HP == 0) {   //主人公の HP が 0 になったら
                    Dice2(3);   ランダム変数 E3～18
                    if (E<_root.myStatus.GP) {   //倒れずにすむ判定
                                            _root.myStatus.HP=1;
                    RefreshDispStatus();
                    _root.DispTalk.TalkText  +="不屈の闘志でかろうじて立ち上がった！";
```

```
                    _root.DispTalk._visible = true;
}else{   //倒れてしまったら
_root.DispTalk.TalkText   =   monster[AttackMonNo].Name+" の 攻 撃 。
"+String.fromCharCode(13);
                if(Ga eq 1){
                _root.DispTalk.TalkText  += "「意地と根性で攻撃を防いだ。」
"+String.fromCharCode(13);
                 }
                                _root.DispTalk.TalkText  += myName+"は鼻血を出し
て気絶した！"+String.fromCharCode(13);
                                _root.DispTalk.TalkText  += "気がつくと自分の部屋だ
ったが、お金が減っている。物騒な世の中だ。";
                                _root.monsterMode = false;
                                _root.myStatus.HP = _root.myStatus.MaxHP;
                    _root.myStatus.Gold = Math.floor(_root.myStatus.Gold/2);
                                _root.sndBattle.stop();
                                gotoAndPlay ("寮");
                    }}   //HP 回復、お金を半分にして寮に戻す
```

58 フレーム

```
if (_root.DispTalk._visible eq true) {   //プレイヤーのテキスト読み待ち
        gotoAndPlay ("monAttackLoop");
} else {
        AttackMonNo++;   //変数を 1 増やして
        if (AttackMonNo<monster.length) {
//まだ敵の攻撃が全部終わっていなければ
                Ga=0;   //ガードの初期化
                    gotoAndPlay ("monAttack");   //もう一度敵の攻撃に飛ばす
        } else {
                    gotoAndPlay ("AttackMenu");   //主人公の攻撃へ飛ばす
        }
}
```

ムービークリップ menu
　　1 フレーム目

```
stop ();
// ------------------------------------------------
function MenuDisp () {
//menu に代入されたコマンド配列にあわせてマークの初期化、整形を行う関数
        this.MenuNumber = -1;
        DispNum = eval("_root."+ cmdName).length;
        for (i = 0; i < DispNum; i++) {
        eval(this+"."+i).MenuItemName = eval("_root."+ cmdName)[i].Name;
                eval(this+"."+i).MarkArm.gotoAndStop(1);
        }
        resize();
}
// ------------------------------------------------
function MenuDisp2 () {
//道具を表示する時に使用する関数。装備中の道具にはマークがつく
        DispNum = eval("_root."+ cmdName).length;
        for (i = 0; i < DispNum; i++) {
        eval(this+"."+i).MenuItemName = eval("_root."+ cmdName)[i].Name;
                if (_root.item[i].Arms) {
                        _root.menu2[i].MarkArm.gotoAndStop(2);
                }
                else {
                        _root.menu2[i].MarkArm.gotoAndStop(1);
                }
        }
        resize();
}
```

```
// ------------------------------------------------
function resize () {
//メニューに表示するものの数にあわせて menu を整形する
	for (i=0; i<_root.maxMenu; i++) {
		eval(this+"."+i)._visible = false;
	}
	for (i=0; i<DispNum; i++) {
		eval(this+"."+i)._visible = true;
	}
	MenuBottom._y = eval(this+"."+DispNum)._y;
}
// ------------------------------------------------
function DispMenuMark (AddNum) {
//十字キーの上、下を押した時にメニューマークが移動するｙ関数
	DispNum = eval("_root."+ cmdName).length;
	this.MenuNumber += AddNum;
	if ( DispNum <= this.MenuNumber ) {
		this.MenuNumber = DispNum-1;
	}
	if ( this.MenuNumber < 0) {
		this.MenuNumber = 0;
	}
	for (i = 0; i < DispNum; i++) {
		if (i == this.MenuNumber) {
			eval("_root."+_name)[i].gotoAndStop(2);
		}
		else {
			eval("_root."+_name)[i].gotoAndStop(1);
		}
	}
}
// ------------------------------------------------
```

2 フレーム目

```
function Namesort (a, b) {    //道具の整理を行うために定義した関数
        if (a.Name<b.Name) {
                return -1;
        } else if (a.Name>b.Name) {
                return 1;
        } else {
                return 0;
        }
}
// ExecNo
// 1:「どうぐ」Menu2 にどうぐメニュー出す
// 2:「じょうたい」Menu2 にじょうたいメニュー出す
// 3:「こうげき」Menu2 に敵リスト出す
// 4:「つよさをみる」つよさを表示
// 5:「どうぐせいり」Item をならべかえ
// 6:「つかう」
// 7:「そうび」
// 8:「みる」
// 9:「すてる」
// 10:「やめる」
// 11:「アイテム 1 を選択」Menu3 にどうぐメニュー１を出す
// 12:「アイテム 2 を選択」Menu3 にどうぐメニュー２を出す
// 13:「アイテム 3 を選択」Menu3 にどうぐメニュー３を出す
// 14:「こうげき相手選択」こうげき
// 15:「やめる」メニューを閉じる
// 16:「かいにきた」
// 17:「うりにきた」
// 18:「やめる」
// 21:「アクセスアイテム」
// 22:「自販機メニュー」
// 23:「にげる」（戦闘から）
// 24:「勉強」Menu2 に勉強リストをだす
// 25:勉強内容を選択、バトルモードにいく。
// 26「かいにきた」自動販売機メニューを出す
// 27:「一休み」HP の回復をする
```

```
//    //前のページの ExecNo に対応した処理を行うように設定していく
ExecNo = eval("_root."+cmdName)[this.MenuValue].ExecNo;
if (ExecNo eq 1) {    //itemMenu を menu2 に表示して menu2 を整形
        for (i=0; i<_root.maxMenu; i++) {
                eval("_root.menu2."+i).gotoAndStop(1);
        }
        _root.RefreshItemMenu();
        _root.menu2.cmdName = "itemMenu";
        _root.menu2.MenuDisp2();
        _root.menu2._visible = true;
        _root.NowMenu = "menu2";
} else if (ExecNo eq 2) {    //state を memu2 に表示後整形
        for (i=0; i<_root.maxMenu; i++) {
                eval("_root.menu2."+i).gotoAndStop(1);
        }
        _root.menu2.cmdName = "state";
        _root.menu2.MenuDisp();
        _root.menu2._visible = true;
        _root.NowMenu = "menu2";
}

else if (ExecNo eq 23) {    //戦闘で逃げるを選んだ時の ExecNo の処理
        _root.Dice2(3);    //ランダム数 E と主人公の AP の半分の大小で判定
        if (Math.floor(_root.myStatus.AP/2)>_root.E) {
                _root.gotoAndPlay(_root.returnFrame);
                _root.DispTalk.TalkText = "つらい現実から一時逃げ出した";
                _root.DispTalk._visible = true;
                _root.menu1._visible = false;
                _root.NowMenu = "menu0";
            _root.monsterMode = false;
        } else {
                _root.gotoAndPlay ("monAttackInit");
                _root.DispTalk.TalkText = "逃げられない。";
                _root.DispTalk._visible = true;
                _root.menu1._visible = false;
                _root.NowMenu = "menu0";
```

```
        }
} else if (ExecNo eq 27) {    //ドミトリーで一休みを選んだ時に行う処理
                _root.gotoAndPlay("ねる");
                _root.DispTalk.TalkText = "おやすみなさい・・・・ＺＺＺＺＺ・・・・";    //シーン 1 のラベルねるへ飛ばし HP の半分を回復
                _root.DispTalk._visible = true;
                _root.menu1._visible = false;
                _root.NowMenu = "menu0";
                _root.myStatus.HP += Math.floor(_root.myStatus.MaxHP/2);
                if (_root.myStatus.HP>_root.myStatus.MaxHP) {
                        _root.myStatus.HP = _root.myStatus.MaxHP;
                }    //回復で最大を超えた場合最大値になるようにした
```

このように ExecNo を設定することで menu にプレイヤーが起こした反応にあわせて対応することが可能になっています。

・オブジェクトアクション（ムービークリップやボタンにかかれたActionScript）

menu3 に書かれたオブジェクトアクション

```
onClipEvent (load) {　//このムービーが読み込まれたら
　　_visible = False;　//このムービークリップを消す。
}
```

このように読み込まれた途端に消されているものは多く、表示・非表示で制御を行ったり、非表示のうちに表示項目の準備をしておいて表示するようにしてあります。

mychar のオブジェクトアクション(A 棟 3 階に配置されたもの)

```
onClipEvent (enterFrame) {
　　var mapname = "_root.map_a3";　//変数にマップの名前を記録させる
　　var vx = 0;
　　var vy = 0;
　　var MoveDist = 9;　//移動量を設定
if (_root.moveFlag eq false) {　//フラグが false なら
　　return;　　//移動をできなくする
　　} //return は値を返す命令だが、さらに実行後それ以降のスクリプトが実施されなくなる特性を持っている
　　if (Key.isDown(Key.RIGHT)) {　//右キーが押されたら
if(_root.houkou ne 1){　//すでに右歩きのムービーが表示されていなければ
this.gotoAndPlay ("右");　//表示する
}
　　vx = MoveDist;　//設定した移動量だけ右に動く
　　} else if (Key.isDown(Key.LEFT)) {
if(_root.houkou ne 2){
this.gotoAndPlay ("左");
}
　　vx = -MoveDist;
　　} else if (Key.isDown(Key.DOWN)) {
if(_root.houkou ne 3){
this.gotoAndPlay("まえ");
}
　　vy = MoveDist;　//Flash の座標は下が y 軸の正なので
　　} else if (Key.isDown(Key.UP)) {
```

```
if(_root.houkou ne 4){
this.gotoAndPlay ("うしろ");
}
                    vy = -MoveDist;
          }
          eval(mapname)._x -= vx;   //主人公はステージの中央なので
          eval(mapname)._y -= vy;   //実はここでマップ自体を動かしている
          var clashFlag = false;   //壁に当たったかをフラグに定義
          for (i=1; i<=600; i++) {
                    if (this.hitTest(mapname+".w"+i)) {
                              clashFlag = true;
                    }
//w＋数字のインスタンス名のムービーとの接触判定をしている
          }


if (this.hitTest( eval(mapname).ups0 )) {
//マップに配置された ups0 に当たっていたら
_root.DispTalk.TalkText = "A 棟の 2 階だ。"+ String.fromCharCode(13);
                              _root.DispTalk._visible = true;
                              _root.moveFlag = false;   //動きを止めて
_root.down._visible=true;   //下り階段のムービーを表示し
_root.down.gotoAndPlay (2);   //再生する
                    _root.myX = -196;   //座標を決めて
                    _root.myY = 571;
                    _root.gotoAndPlay ("A 棟 2");
//階段のちょうどいい位置に表示させる
          }
if (this.hitTest( eval(mapname).ups1 )) { //ups1 に接触したら
if(_root.SO eq 100){   //卒論ができてない場合
_root.DispTalk.TalkText = "ここをのぼると電子・光システム工学科の教員室の
階に行くぞ。"+ String.fromCharCode(13);
_root.DispTalk.TalkText  += "先生に見つかったらどうする!!。"+
String.fromCharCode(13);
_root.DispTalk.TalkText += "卒論をドミトリーで書いて叩きつけてやろう！"+
String.fromCharCode(13);
```

```
if(_root.eventFlag[40] eq false){   //ここにはじめてきた場合
_root.myStatus.Exp += 40;   //経験値をプレゼント
_root.RefreshDispStatus();   //強さの表示を更新
_root.eventFlag[40]=true;   //ここに一度きたことを記録
}
                        _root.DispTalk._visible = true;
                        _root.moveFlag = false;
                        clashFlag = true;   //壁に当たったことにする
}
if(_root.SO eq 1000){   //卒論ができていたら
                        _root.monsterArea = 5;   //ボス戦へ
                        _root.returnFrame = _root._currentframe;
                _root.myX = _root.StageCenterX+275-eval(mapname)._x-vx;
                _root.myY  =  _root.StageCenterY+200-eval(mapname)._y-vy;
//プレイヤーが逃げるを選んだ時のために座標とフレーム名を覚えさせる
                        _root.gotoAndPlay("battle");
}

        }

if (_root.LevelData[_root.myStatus.Level-1].NextExp<=_root.myStatus.Exp)
{   //知識を得てレベルアップした場合のレベル更新
_root.myStatus.MaxHP = _root.LevelData[_root.myStatus.Level].MaxHP;
_root.myStatus.Attack = _root.LevelData[_root.myStatus.Level].Attack;
_root.myStatus.Defence = _root.LevelData[_root.myStatus.Level].Defence;
_root.myStatus.DX = _root.LevelData[_root.myStatus.Level].DX;
_root.myStatus.IQ = _root.LevelData[_root.myStatus.Level].IQ;
_root.myStatus.AP = _root.LevelData[_root.myStatus.Level].AP;
_root.myStatus.GP = _root.LevelData[_root.myStatus.Level].GP;
        _root.myStatus.Level++;
        _root.myStatus.HP = _root.myStatus.MaxHP;
        _root.RefreshDispStatus();
_root.DispTalk.TalkText  +=  _root.myName+"は"+_root.myStatus.Level+"レ
ベルに上がった！"+String.fromCharCode(13);
}
        // ----------------------------------------------------------------
```

```
if (clashFlag eq true) {    //壁に当たったら
                    eval(mapname)._x += vx;
                    eval(mapname)._y += vy;    //座標を元に戻す
          } else if (vx+vy ne 0) {    //移動したら
                    _root.MeetMonster--;    //MeetMonster を減らしていって
                    if (_root.MeetMonster<0) {    //0 より少なくなったら
                    _root.MeetMonster    =    100+Math.floor(Math.random()*20);
//MeetMonster 更新し
                    _root.returnFrame = _root._currentframe;
                    _root.myX = _root.StageCenterX+275-eval(mapname)._x-vx;
                    _root.myY = _root.StageCenterY+200-eval(mapname)._y-vy;
                              _root.monsterArea = 3;
                              _root.gotoAndPlay("battle");
                    }    //バトルへ飛ぶ
          }
}
```

一部ですが代表的に使われている ActionScript の説明を終わります。

4－5　手直し

　一応完成という形になりましたがこの段階で徐々に手直しを行っていきます。
バグを取ったりしていくのは当然としてムービークリップの修正を行います。
ここでは階段の上り下りのムービーの強化を取り上げたいと思います。

当初階段はムービーなしだったのですが、フレームが gotoAndPlay()で飛ぶとすぐに違う階が表示されるために階段を上った感覚が伝わりにくいばかりでなくすぐに動けてしまうので、また違う場所に飛んでいたなどという誤作動まで想定されたことから階段のムービーを作ることにしました。（4－4で説明した ActionScript にはすでに階段ムービーが組み込まれています。）

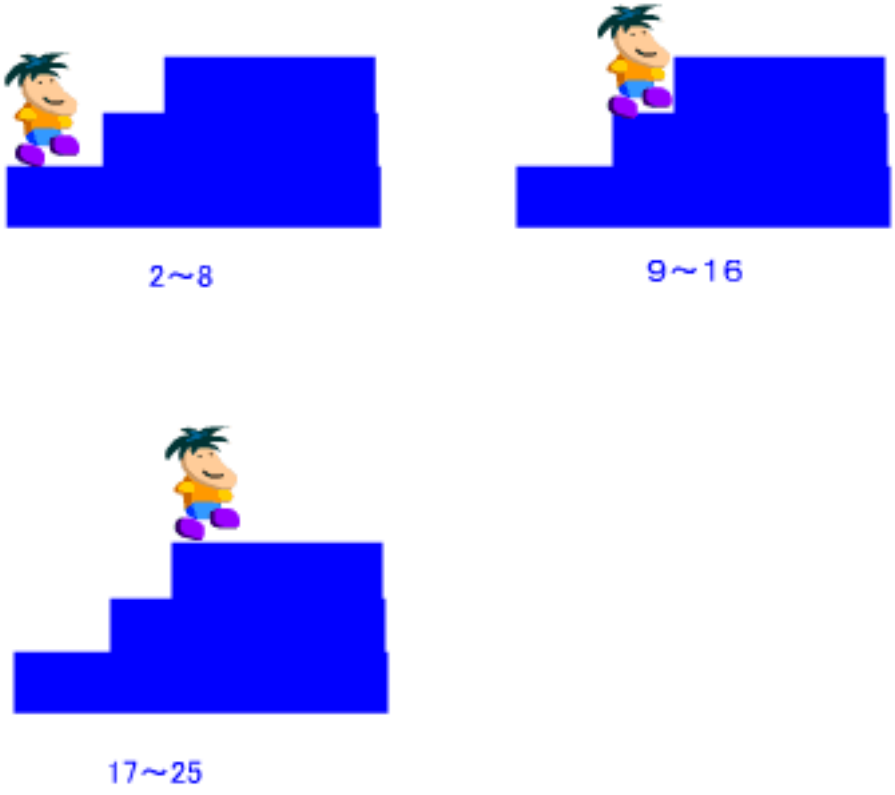

図 4－44　一度作成してみた階段ムービー（下に書いてある数字はフレーム数）

ところが絵が小さかったことや平面的に上っていくムービーのためにパンチのないムービーになってしまい、これを手直しすることになりました。直した結果を図４－４５に示します。

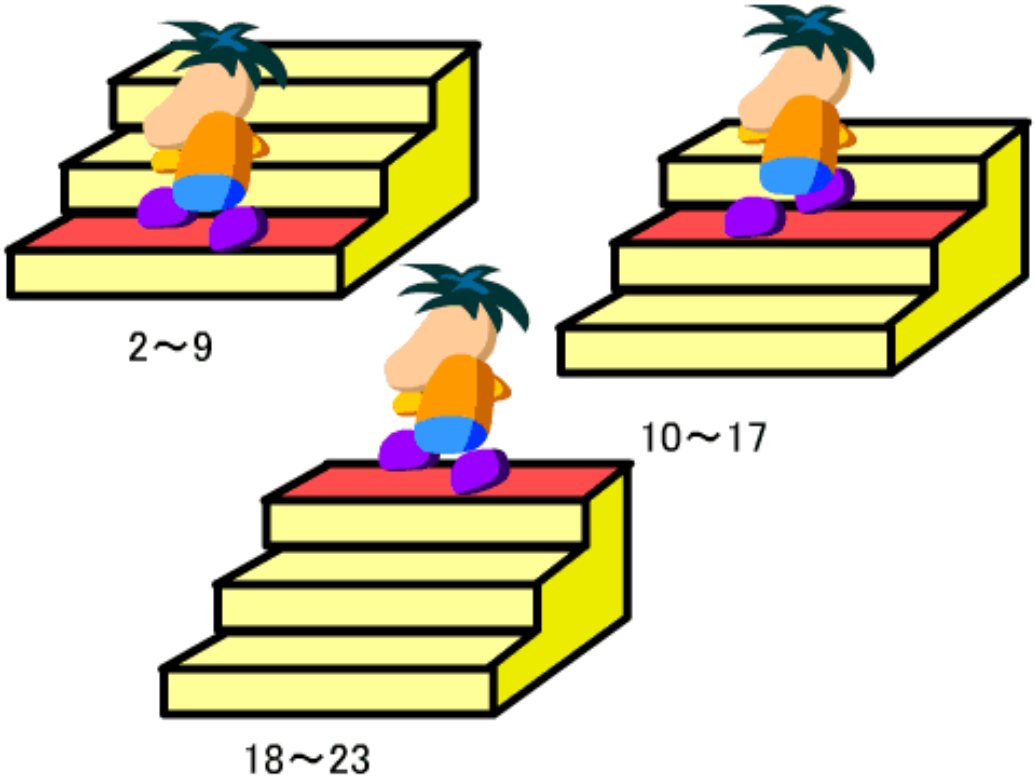

図 4－45　三次元的に直した階段ムービー（下の数字はフレーム数）

三次元的にするだけでなく主人公の手足を動かすことでよりよくなっています。実際のバージョンではこのムービーと共にどこの階に来たのかを表示しています。図４－４６に示します。

図 4－46　実際に採用した階段ムービー（どこに来たのかも表示されている）

ここまででバグ取りは当然見つかり次第行いますが、一応ゲームが完成したとします。

第 5 章　Web 掲載

ゲームはこれで一応完成となりました。しかし、ゲームとして完成することだけが目的ではないのでここで Web 向けに調整を行うことになります。

5－1　ゲームの調整

Flash の機能であるストリーミング機能で実際に配信状態にした時にユーザーにはどう再生されていくか見ることができます。想定環境は 56K の 4.7KB/s ということにします。

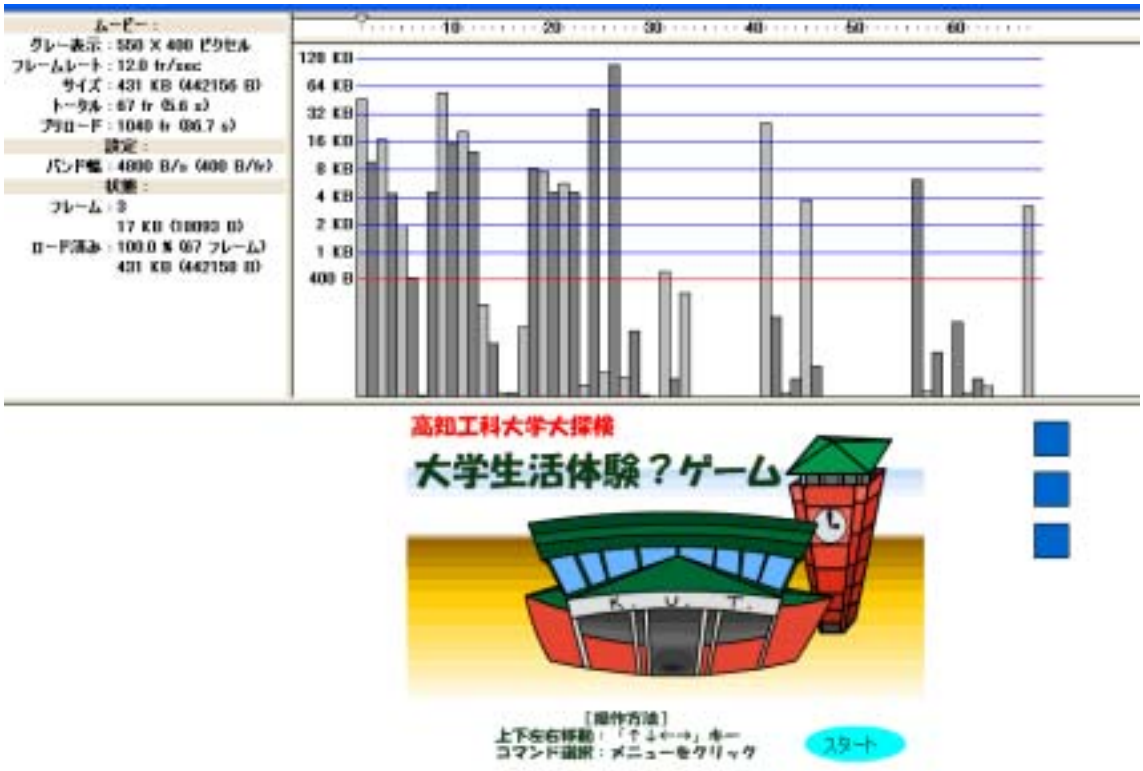

図 5－1　とりあえずできたゲームをストリーミングした図

図５－１に存在する赤い 400B に引かれた線よりバーが上にあると即座には表示できません。このため読み込むタイミングを可能な限り調整しユーザーに真っ白な画面で待たせないようにしたいと思います。できる限り調整したものが図 4－52 です。

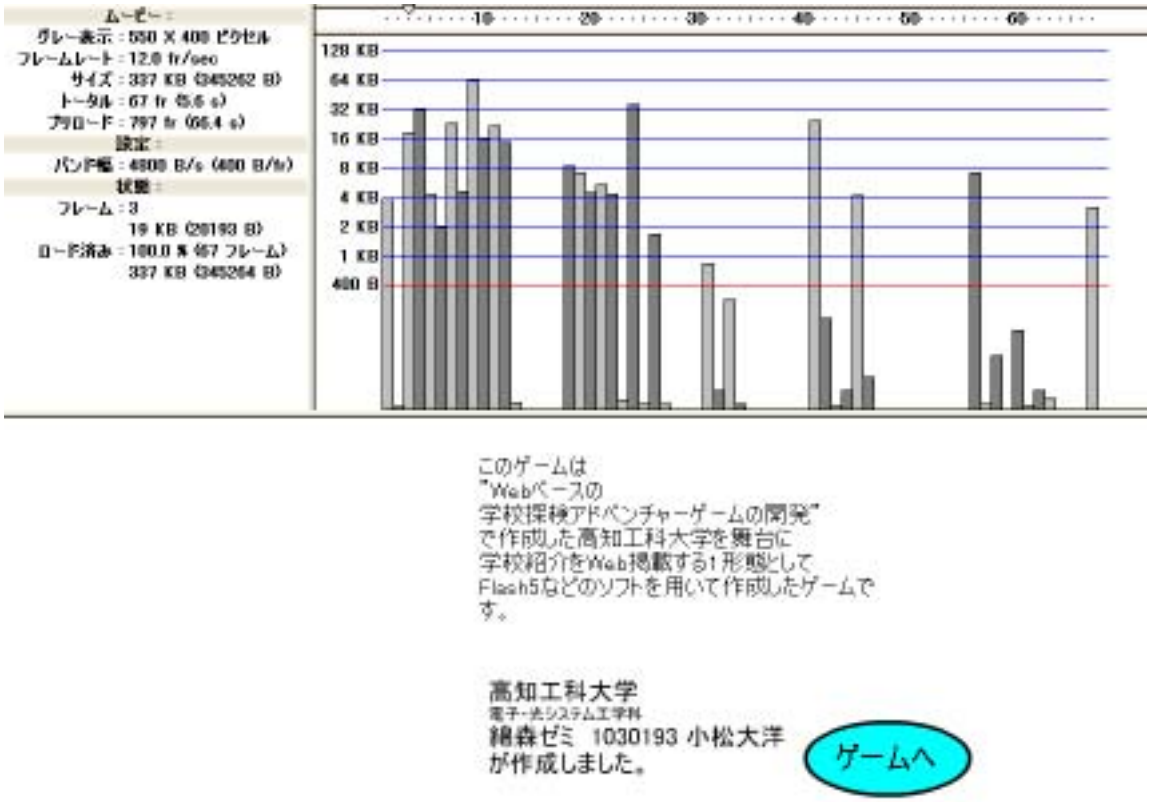

図 5−2　調整したもの

図５－２を見ると 1 フレーム目はすでにデータが赤線を越えていますがテキストデータなので最低限のデータです。このテキストデータを見せて 10 フレーム目を読み込むまで待ってもらうことにします。どうなっているかというと 10 フレームまで読み込んで初めて次に進むボタンが出現する仕組みです。さらにこのゲームはゲームが始まった直後に戦闘を行うことができます。戦闘を行っているフレームは後半にあるので、ゲームに不具合が起きる可能性が出てしまいます。これを防ぐためにまた先ほどの方法で調整しています。また、テキストを読んで、主人公の名前を入力しているうちに時間が経つのでそれほどユーザーに不快感を与えないはずです。ちなみに実際の読み込みは図５－３のようになって行きます。

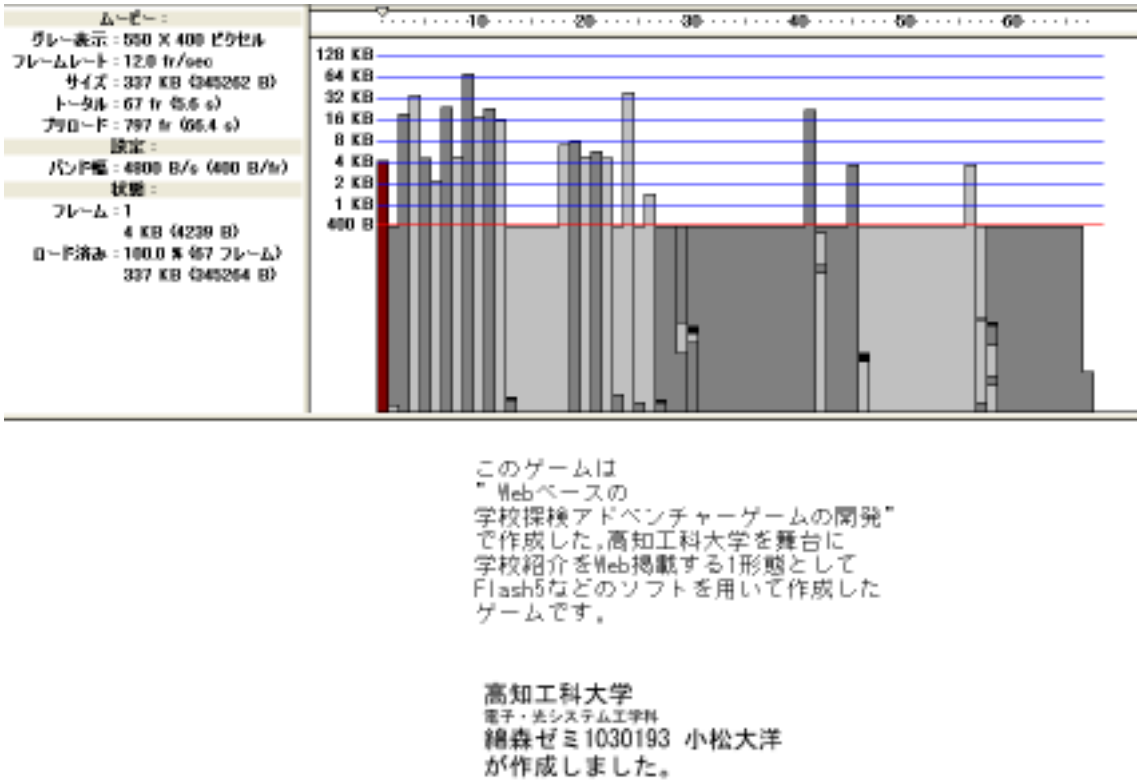

図 5−3　実際に読み込んでいく様子。データがないところには先に次のフレームを読み込んでいくのでこんな形になります

これでゲームを Web 向きに調整することに成功しました。

5－2　ページ作り

ページのポリシーを決めます。

- スクロールしない大きさで作ること
- クリック数が多くないように作ること（階層を深くしない）
- ユーザー側にブラウザを問わない
- 工事中をリンクさせない
- ボタンが大きすぎない

以上のことを念頭において自分のページを作ります。これはユーザーの不快感を少なくする配慮です。
せっかくなので今後ゲームのファイルだけでなく.fla ファイル（Flash）や論文も載せることにします。

Dreamweaver と Fireworks を使って作ります。

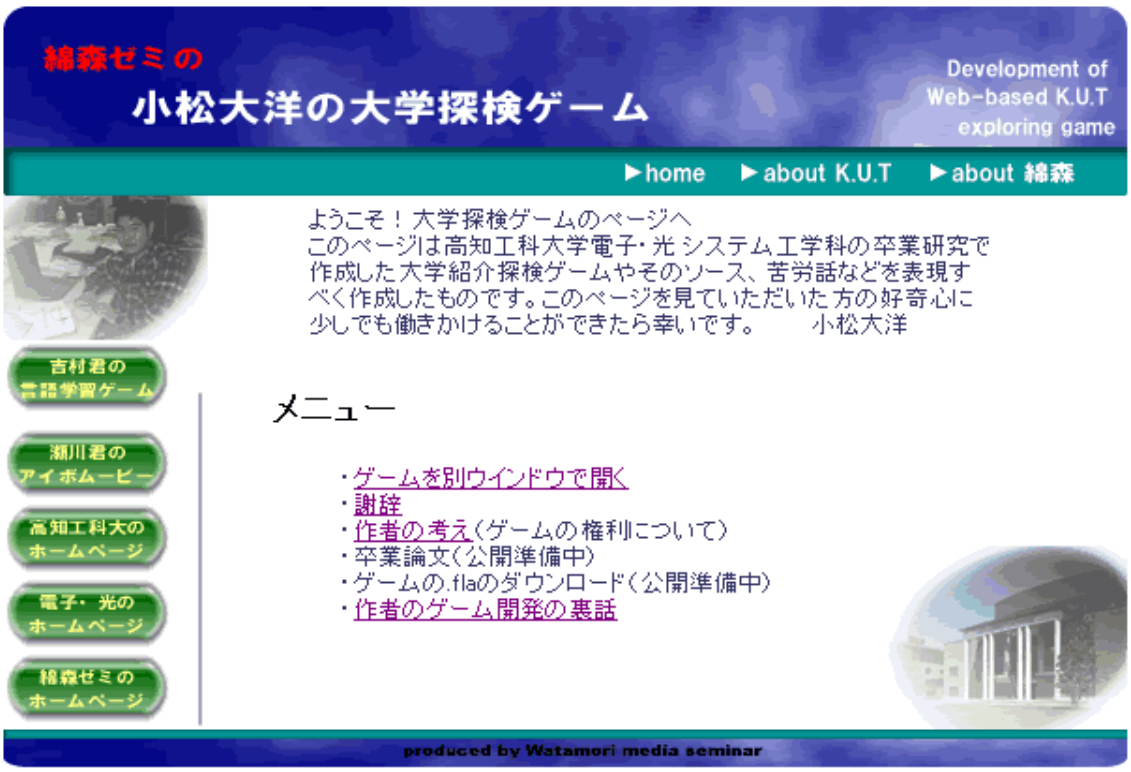

図 5－4　作成した HP

第 6 章　発表 OHP の作成について

本研究で使用した Flash は普段スライドの作成に使用される PowerPoint と同じようにスライドを作成することができます。今回は PowerPoint を使うのはやめて Flash で発表用 OHP を作成します。それは、.swf ファイル（Flash）はホームページに貼り付けることが可能なので PowerPoint とは違いすぐにホームページに掲載することが可能ということも理由のひとつです。

第 7 章　総括

本研究はパクパクモンスターというフリーの Flash ゲームを解析することから始まりました。

その解析したソースを元に自分のテーマにそって新たに自作・構築していく作業は ActionScript の理解をより深いものにし、その根底にあるプログラムはいろいろな仕事をするものの集まりであるという考え方を感じることができました。また、学校紹介の 1 形態を示せたと自負しています。

しかし、問題はそのファイルの大きさだと言わざるを得ません。解析から開発まで時が怒涛のように過ぎてしまい、完全に Web ベースになったとは言い切れない結果となってしまいました。遺憾ながら、今回の想定速度はまだ一般家庭のそれにくらべて高めに設定してあります。今後は、グラフィックの軽減を考えていかないといけないと考えます。そこまで完全な形で開発できなかったことがたいへん残念です。また、部屋の内部を製作する時間的余裕がなく今回は部屋の説明がテキストのみであることは、内部を表示しつつテキストで説明しようという構想が実現できず、ActionScript の部分では可能になっていただけに残念です。

今回 ActionScript の解析に思いのほか時間をとられ、開発に時間的余裕がなくなり、幾度となく挫折しそうになりながら、ここまでやれたことに正直驚いています。

参考文献

1 外間かおり＋なかひらまい　著　Flash5 スーパーリファレンス
2 外間かおり　著　DREAMWEAVER3 スーパーリファレンス
3 Web&HP 研究会編　著　Fireworks3Web 画像作成スーパーテクニック
4 柴田忠浩＋広石里香　著　おしえて!!macromedia　FLASH5　ActionScript
5 Mark Pearrow　著　ログ・インターナショナル　訳
Web サイトユーザビリティハンドブック
6 マーク・クラークソン　著　Flash アニメーターズ・ガイド
7 森巧尚　著　Go!Go!FLASH5[ゲームを作ろう編]
8 リンダ・ワインマン&ガロ・グリーン　共著
Dreamweaver3 Hands-On Training
9 J・スコット・ハムリン/デビット・J・エンバートンほか　著　風工舎　訳
Flash5 ActionScript MAGIC
10 杉山敦　著
ウェブデザイナーのための DREAMWEAVER3&FIREWORKS3
11 上野亨　著 maicromedia FLASH ActionScript バイブル
12　A.c.Suck 著　FLASH アニメーション完全攻略
13　future brain project 著
DREAMWEAVER & FIREWORKS プロフェッショナル・ウェブ・デザイン
14 保坂康介　著　Flash MX 400 Symbol Template Book
15 Robin Williams / John Tellett 著　ノンデザイナーズウェブブック 2001

謝辞

今回の研究と論文の作成にあたり、終始丁寧なご指導とご教示を賜りました高知工科大学電子・光システム工学科綿森道夫助教授に深い感謝の意を表します。綿森道夫助教授の御支援があってこそ本論文が完成したとしか言いようがありません。

高知工科大学電子・光システム工学科在学中にご指導を賜った原央学科長に心から感謝致します。

また高知工科大学電子・光システム工学科在学中、本研究の各過程で終始ご好意、ご協力を頂きました高知工科大学電子・光システム工学科、河津哲教授・矢野政顕教授・成沢忠教授・河東田隆教授・神戸宏教授・畠中兼司教授・平木昭夫教授・平尾孝教授・山本哲也教授・西本俊彦教授・八田章光助教授・野中弘二助教授・橘昌良助教授・高村禎二助教授・井上昌明助教授・関口晃司助教授・笠原泰講師・武田光由実験講師・西田謙助手の方々には重ねて感謝の意を述べさせていただきます。