5. WiiFlash を使ってみよう［ActionScript 編］

デザイン後送

WiiFlashを使うことで、WiiRemoteをActionScriptを用いてAdobe Flash（以下、Flashと表記）とつなげることができます。それにより、Webサイトなど Flashを使ったコンテンツをWiiRemoteで扱うことができるようになります。

この章ではWiiFlash経由でWiiRemoteを扱う方法、そして具体的な作例を紹介しましょう。

5.1 WiiFlashとは

WiiFlashは、WiiRemoteとFlashアプリケーションをつなぐためのライブラリです。Joa Ebert氏とThibault Imbert氏という２人の開発者により開発が進められている無料のプロジェクトです。

WiiFlash : Wiimote and Flash
URL http://wiiflash.bytearray.org/

WiiFlashの仕組み

WiiFlashを触る前に、WiiFlashの仕組みについて理解しましょう。WiiFlashは、「WiiFlash Server」と「WiiFlash ActionScript API」の２つのパートに分けて提供されています。

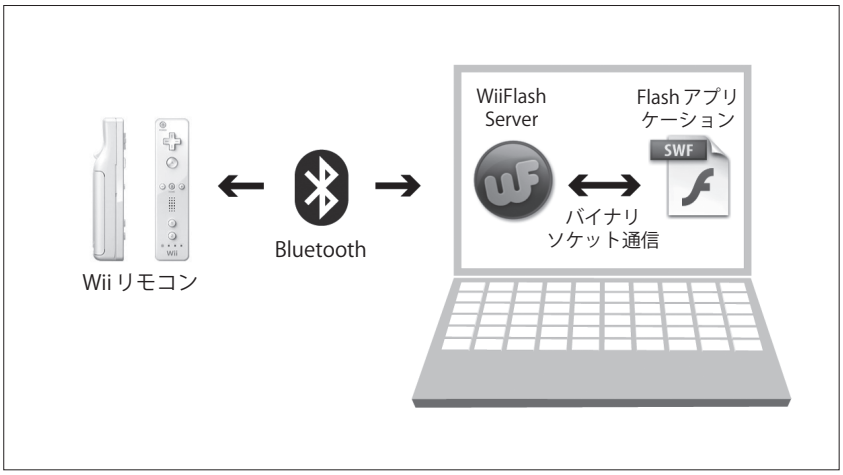
- **WiiFlash Server**

C++もしくは、.NET serverを利用して作られたバイナリソケットサーバーで、WiiRemoteからの情報をFlashが扱える形に変換して通信します。

- **WiiFlash ActionScript API（SWC component）**

WiiFlash Serverからの通信に対して、プログラムしやすいように提供されているのが、このActionScript（Flashのプログラミング言語）APIです。このAPIにより、あまり通信部分を意識せずに、プログラミングすることができます。

図5-1　WiiFlashの仕組み



WiiFlashは、WindowsとMacintosh（以下、Macと表記）両方に対応しています。ただし、WindowsとMacではできることが違うので注意しましょう（WiiFlashの機能については166ページにまとめたので、そちらを参照してください）。

環境設定

Flash CS4のインストール

まずは、一般的なFlashの開発環境であるAdobe Flash CS4をインストールしましょう。ソフトウェアは購入しなくとも、体験版が利用できます。体験版は使用開始日から1ヶ月間のみ有効なので、もし気に入ったら購入しましょう。

Adobe Flash CS4 Professional
URL http://www.adobe.com/jp/products/flash/

● Flash CS4のダウンロード

Adobeサイトのダウンロードページから、「Flash CS4 Professional」→「体験版」を選択します。なお、ダウンロードにはAdobe IDが必要になります。手順に従って、Adobe IDを取得しておきましょう。

図 5-2　ダウンロードページ

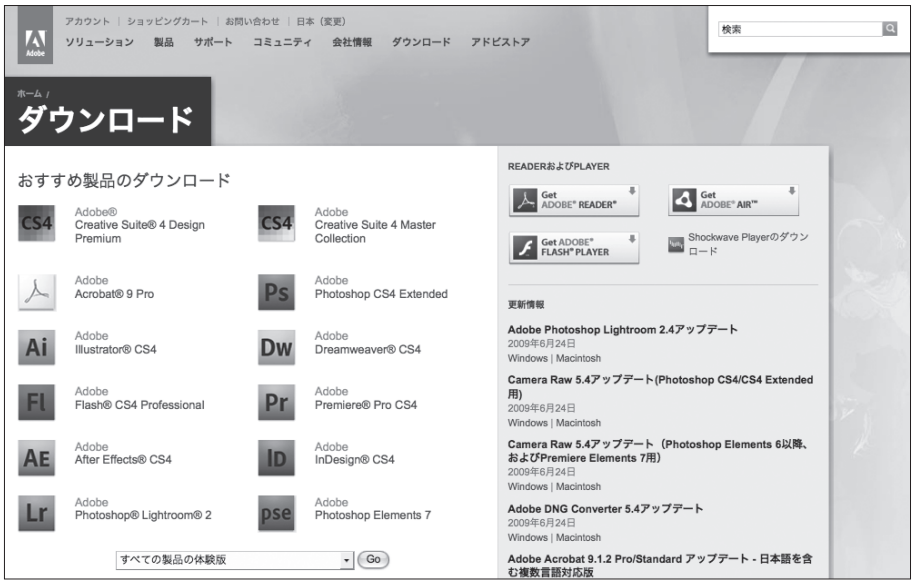


図 5-3　Adobe ID の取得

● Flash CS4 のインストール

ダウンロードしたパッケージ（ZIP ファイル）を解凍し、付属のインストールガイド（Readme.txt）を参照の上インストールしてください。

インストールが完了したらアプリケーションを起動して確かめてください。体験版利用の場合にはソフトウェアのライセンスキー確認画面が出ますが、30日以内であれば「体験版として使用する」をクリックすれば、通常版と同じ機能が使えます。

WiiFlashのインストール

以降では、WiiFlashバージョン0.4.5を使って説明します。

なおWiiFlash Serverは「Microsoft .NET Framework 3.0 Redistributable Package」が必要になります※1。ダウンロードはこちらから。

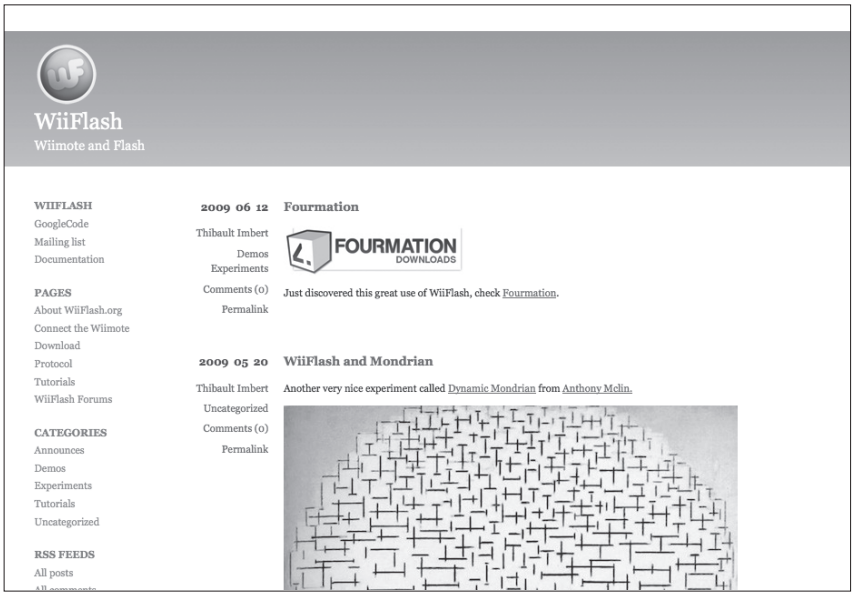
Microsoftダウンロードセンター

URL http://www.microsoft.com/downloads/details.aspx?FamilyID=10cc340b-f857-4a14-83f5-25634c3bf043&DisplayLang=ja

●WiiFlashのダウンロード

WiiFlashサイトを開き、左側のメニューにある「PAGES」→「Download」を選択します。

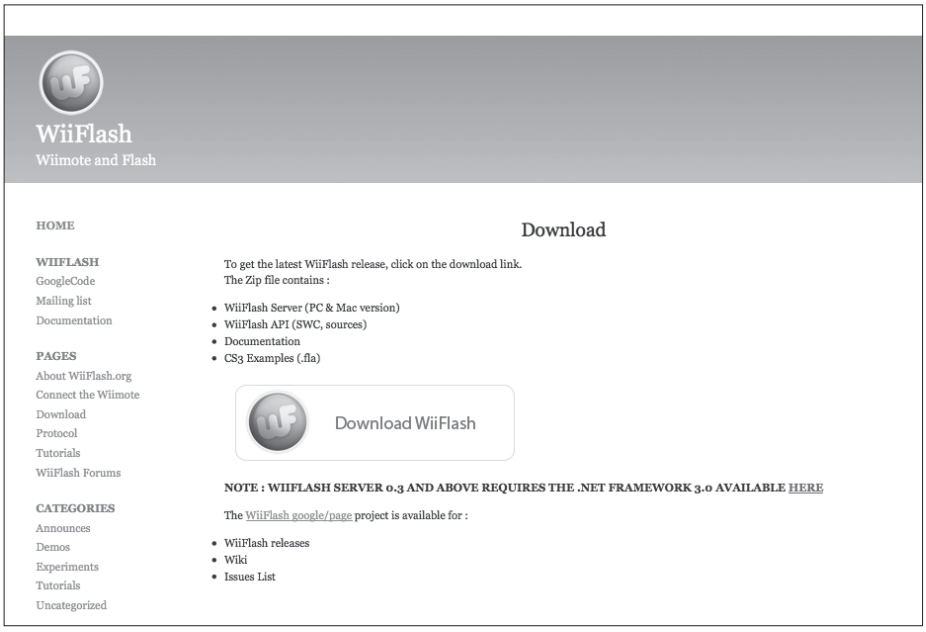
図5-4　WiiFlash（http://wiiflash.bytearray.org/）



※1：第4章ですでに.NET開発環境を導入している場合はインストールする必要はありません。

表示される画面から「Download WiiFlash」をクリックし、アーカイブファイル（ZIP形式）をダウンロードします。

図5-5 「Download WiiFlash」をクリック

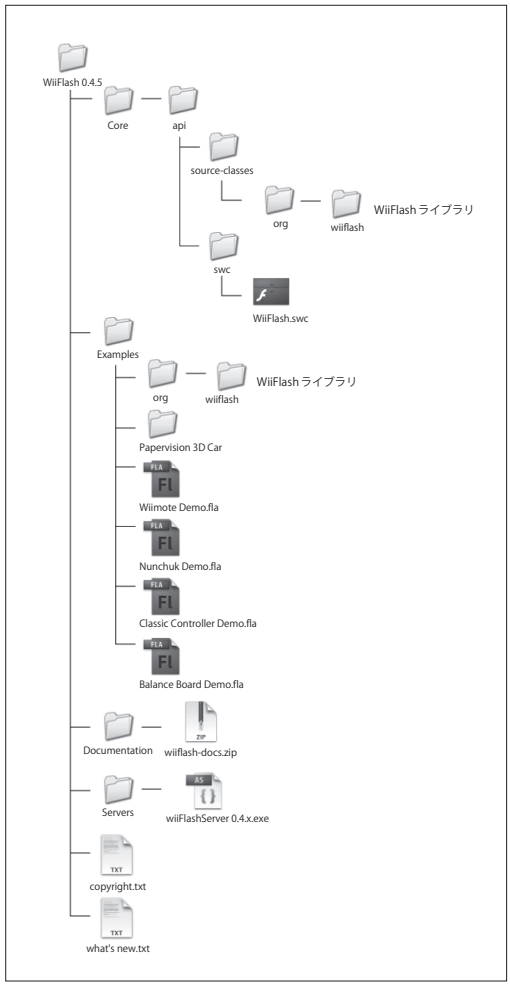


ダウンロードしたWiiFlashのアーカイブを展開すると、次のようなフォルダが現れます。

表5-1 展開したフォルダ（WiiFlash 0.4.5）ファイルの内容

フォルダ名	概要
Servers/WiiFlash Server（latest version）	WiiFlash Serve本体
Core/api	WiiFlash API（SWC, sources）
Examples	デモファイル（.fla）
Documentation	ドキュメント

図 5-6　展開した WiiFlash のファイル・フォルダ構造



WiiFlash の基本機能

前述のように、WiiFlash は Windows と Mac の両方に対応していますが、Windows でできても Mac ではできないということもあるので注意しましょう。

表5-2　WiiFlashの基本機能（バージョン 0.4.5）

リモコン	系統	機能	Windows	Mac
WiiRemote	イベント系	接続ができているか	○	○
		接続ができた	○	○
		接続が切れた	○	○
		赤外線1点目感知	○	
		赤外線1点目消失	○	
		赤外線2点目感知	○	
		赤外線2点目消失	○	
	ボタン系	十字キー（上下左右）	○	○
		Aボタン	○	○
		Bボタン	○	○
		+ボタン	○	○
		ーボタン	○	○
		HOMEボタン	○	○
		①ボタン	○	○
		②ボタン	○	○
		電源ボタン	×	×
	加速度センサー	X軸の加速度	○	○
		Y軸の加速度	○	○
		Z軸の加速度	○	○
		Yaw値	○	○
		Pitch値	○	○
		Roll値	○	○
	赤外線	1点目	○	
		2点目	○	
		1点目のX値	○	
		1点目のY値	○	
		2点目のX値	○	
		2点目のY値	○	
	その他	バッテリーレベル	○	○
		バイブレーション	○	○
ヌンチャク	イベント系	接続ができているか	○	○
		接続ができた	○	○
		接続が切れた	○	○
	ボタン系	StickX	○	○
		StickY	○	○
		Cボタン　○	○	
		Zボタン　○	○	
	加速度センサー	X軸の加速度	○	○
		Y軸の加速度	○	○
		Z軸の加速度	○	○
		Yaw値	○	○
		Pitch値	○	○
		Roll値	○	○
バランスボード		左上の圧力	○	
		右上の圧力	○	
		左下の圧力	○	
		右下の圧力	○	
		重力の中心点	○	

WiiFlash のデモファイルでテストする

まずは WiiRemote でテストしてみます。WiiRemote を PC の Bluetooth スタックに接続して、WiiFlash Server を立ち上げましょう。

図 5-7 のように「1 Wiimote(s) found」と表示されればつながっています。

図 5-7　正常に接続されている

接続ができていない場合は図 5-8 のように丸い「WF」の部分が赤く表示されます。

図 5-8　接続できていない

次に、ダウンロードしたパッケージの中にある「Examples」フォルダから Wiimote Demo.fla を開きましょう。

このファイルでは、電源ボタンを除く、WiiRemote のボタンすべてと「Sensor X」「Sensor Y」「Pitch」「Roll」「Yaw」の加速度センサーの値、さらに、「Battery level」を確認することができます。正常に接続されていれば、ボタンを押すことで、対応する画面上のボタン部分が反応します。

図 5-9　動作テスト（Wiimote Demo.fla）

このように、サンプルのデモファイルがついているので、楽に開発を行うことができます。

確認ができたら、次の項ではサンプルファイルで実際の作り方を見てみましょう。

ボタンアクションの取得を使ったサンプル

まずは、サンプル 1 を開いてください。このサンプルはボタン 1 つで Flash のアニメーションを再生／停止する簡単なサンプルです。

ライブラリを確認すると、「a」と「synchronize」という 2 つのムービークリップが入っています。

図5-10　ライブラリの中を確認する

「a」はボタンのアニメーションが入っているムービークリップです。「synchronize」はWiiFlash Serverとやり取りをするために必要なファイルです。

POINT ▶▶▶ 新規でWiiFlashを使用したアプリケーションを作る際には、必ずAPIが入っている「org」フォルダをflaファイルと同じディレクトリに入れて作成を開始しましょう。また、ライブラリのsynchronizeというムービークリップも必要となるので、忘れずにコピーしましょう。

それでは、さっそくサンプルファイルのスクリプトを見ていきましょう。

ライブラリのインポート

Flashではサードパーティ製のライブラリを使うときには、必ず「インポート」という、その機能を取り込む作業が必要です。

コード5-1　ActionScript　WiiFlashのライブラリをインポートする

```
import org.wiiflash.Wiimote;
import org.wiiflash.events.ButtonEvent;
import org.wiiflash.events.WiimoteEvent;
import flash.events.*;
```

WiiRemoteをインスタンス化して、使えるようにします。

コード5-2 ActionScript WiiRemoteをインスタンス化

```
// create a new Wiimote
var myWiimote:Wiimote = new Wiimote();
```

WiiFlash Serverに接続する処理は次の部分です。

コード5-3 ActionScript WiiFlash Serverに接続

```
// connect wiimote to WiiFlash Server
myWiimote.connect ();
```

エラーや接続状況などを管理する「synchronize」というムービークリップをインスタンス化し、画面に配置します。

コード5-4 ActionScript synchronizeを配置

```
var mySynchronize:Synchronize = new Synchronize();
stage.addEventListener ( MouseEvent.CLICK, onClick );
// makes the wiimote rumble
function onClick ( pEvt:MouseEvent ):void{
 myWiimote.rumbleTimeout = 1000;
}
mySynchronize.x = (stage.stageWidth - mySynchronize.width) / 2;
mySynchronize.y = (stage.stageHeight - mySynchronize.height) / 2
```

エラーなどのイベントハンドラを登録します。

コード5-5 ActionScript イベントハンドラの登録

```
myWiimote.addEventListener( Event.CONNECT, onWiimoteConnect );
myWiimote.addEventListener( IOErrorEvent.IO_ERROR, onWiimoteConnectError );
myWiimote.addEventListener( Event.CLOSE, onCloseConnection );
```

コード5-6 ActionScript イベント管理

```
function onCloseConnection ( pEvent:Event ):void{
 mySynchronize.infos_text = "Connection closed, ↩
                                        WiiFlash Server has been closed";
 addChild( mySynchronize );
```

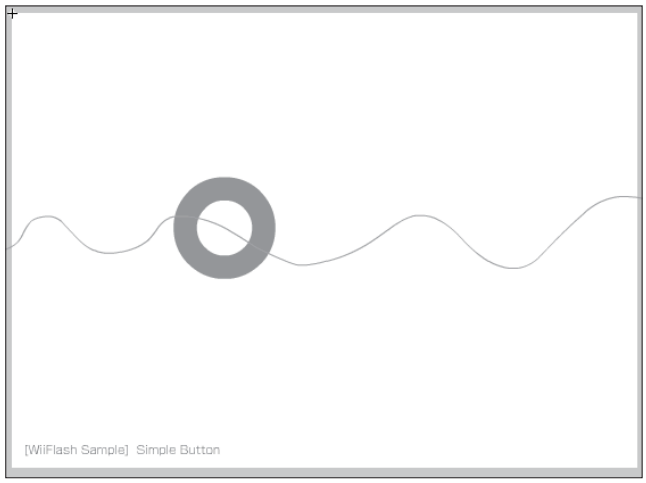
次ページにつづく↗

```
}

function onWiimoteConnectError ( pEvent:IOErrorEvent ):void{
 mySynchronize.infos_txt.text = "Couldn't connect, ↩
                                  make sure WiiFlash Server is running !";
}

function onWiimoteConnect ( pEvent:Event ):void{
 removeChild ( mySynchronize );
 infos_txt.text = "Wiimote successfully connected";
}
```

ボタンイベントを使ったアクション

前述のように、このサンプルはボタン 1 つで Flash のアニメーションを再生／停止する簡単なサンプルです。A ボタンを押すと画面上の円（ムービークリップ：anime_mc）のアニメーションが再生されます。ボタンを離すとアニメーションは停止します。

図 5-11　ボタンで Flash のアニメーションを再生／停止



ボタンを使うにはまず、イベントの登録をします。ボタンのイベント登録は基本的に「on」と「off」を設定します。

・ButtonEvent.A_PRESS

A ボタンが押された際にイベントを発動する

・**ButtonEvent.A_RELEASE**

Aボタンが離された際にイベントを発動する

コード5-7 ActionScript ボタンイベントの登録

```
myWiimote.addEventListener( ButtonEvent.A_PRESS, onAPressed );
myWiimote.addEventListener( ButtonEvent.A_RELEASE, onAReleased);
```

Aボタンが押された際にイベントが発動して、呼ばれるメソッドがこちらです。

画面に配置されたcolor_mcフレームを２フレーム目に変更します。それにより、背景が赤くなります。

コード5-8 ActionScript Aボタンが押された際に呼ばれるメソッド

```
function onAPressed ( pEvt:ButtonEvent ):void {
    anime_mc.play();
}
```

Aボタンが離された際にイベントが発動して、呼ばれるメソッドがこちらです。

画面に配置されたcolor_mcフレームを元の1フレーム目に変更します。それにより、背景が白くなります。

コード5-9 ActionScript Aボタンが押された際に呼ばれるメソッド

```
function onAReleased ( pEvt:ButtonEvent ):void {
    anime_mc.stop();
}
```

このようにWiiFlashでは、簡単に始められるように、リモコンのボタンを使うためのメソッドが用意されています。

このサンプルを使って、アニメーションの再生中身を変更するだけで、簡単にWiiRemoteを使ったFlashのアプリケーションを作れます。変更は、「anime_mc.play();」の部分を変更するだけです。サンプルの処理が理解できれば問題なく使うことができるでしょう。

もっと詳しく勉強したい方は、次節から紹介する作例を参考にしてください。

ここで紹介しているのサンプルの応用やバランスボードを利用したものになっているので、ぜひ理解して自分なりの作品作りに活かしてください。

5.2 作例1：座布団（ザプトン）

制作／尾崎俊介

この節では、WiiBalanceBoard と Flash を使ったサンプルを紹介します。

図5-12　座布団（ザプトン）

Wiiコントローラーの中でも特殊なWiiBoardは、4点の圧力センサーの各箇所ごとの加重を計ることができます※1。この4点の合計を取って、体重計を作ることもできます。ほかにも、4点の圧力センサーのどこに比重が掛かっているかを判断し、上に乗っている人の重心がわかる機能も備えています。バランスを取るゲームでは、こちらの値が使われているのでしょう。今回は、こ

※1：WiiBoardの詳しい説明は、「WiiFlashの基本機能」を参考にしてください。

の重心の移動を取得する機能を使ってサンプルを作りました。

Flash で姿勢が悪い人を救いたい

姿勢を直すことはとても時間のかかる作業です。姿勢の悪さは、常に気を張って意識をし、その座り方自体を習慣化しないことには解決することはできません。

図 5-13　姿勢を直すことはとてもたいへん

そこで、この姿勢を正すことを強く「意識する」ということをいかに習慣づけるのかと考えた結果、以下のテーマに落ち着きました。

姿勢が悪いことは恥ずかしい

この「恥ずかしい」という意識こそが、人の姿勢を直す際の大いなる助けになると思いました。この恥ずかしいという意識を持ってもらうために開発したサンプルが、姿勢強制「座布団（ザプトン）」です。

図5-14　座布団（ザプトン）

この座布団をBlueToothを使い、PCに接続します。WiiFlashを立ち上げて、認識されたら、Flashアプリを立ち上げます。接続が確認できたら、座布団を普段使っているいすの上にのせ、その上に座ります。姿勢が重心から外れると「ぷ〜！」とおならの音が鳴ります。座布団の中にWiiBoardを仕込んで、重心移動を測定しているのです。

図5-15　座布団（ザプトン）の構成

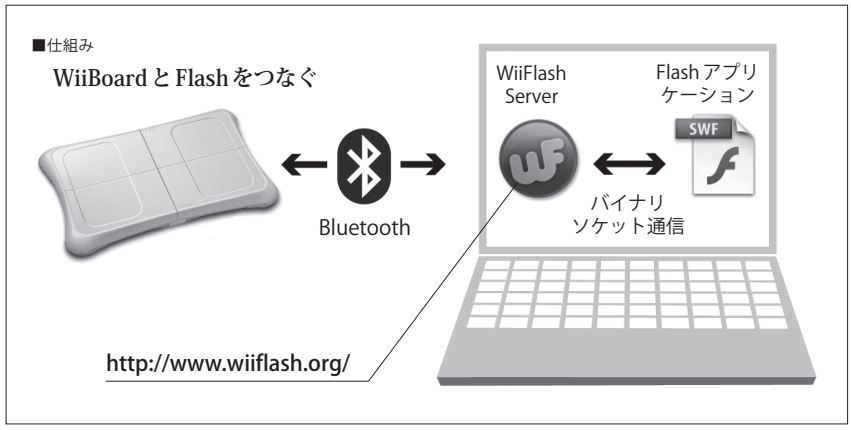
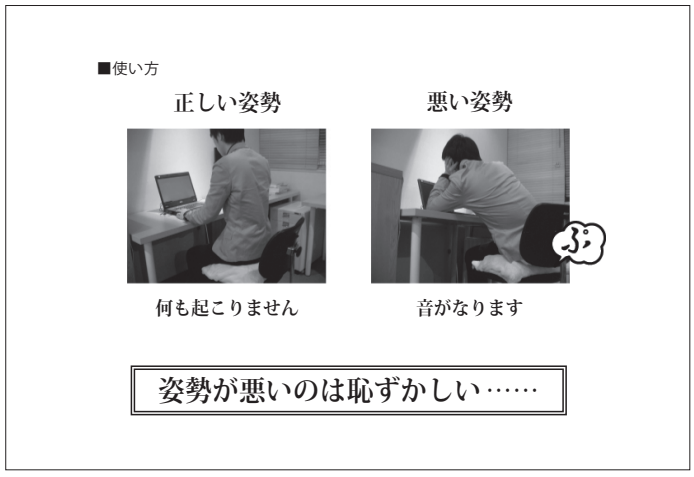


図5-16　姿勢が重心から外れると「ぷ〜!」



重心移動の変化量を使ったアプリ

画面上では、重心の移動を元に黄色の点の位置を変化させます。プログラムを見ていきましょう。

まずは初期設定部分です。前節で解説した通り、WiiFlashライブラリの読み込み、インスタンス化など、WiiFlashを使うために必要な処理です。

コード5-10　ActionScript　WiiFlashの初期設定

```
import org.wiiflash.Wiimote;
import org.wiiflash.events.ButtonEvent;
import org.wiiflash.events.WiimoteEvent;
import flash.events.Event;

// Wiimoteのインスタンス化
var myWiimote:Wiimote = new Wiimote();
// リモコンの接続確認・エラー用にリスナー登録
myWiimote.addEventListener( Event.CONNECT, onWiimoteConnect );
myWiimote.addEventListener( IOErrorEvent.IO_ERROR, onWiimoteConnectError );
//バランスボードへ接続
myWiimote.addEventListener( WiimoteEvent.UPDATE, onUpdated );
myWiimote.addEventListener( WiimoteEvent.BALANCEBOARD_CONNECT,
                                                onBalanceBoardConnected );
```

次ページにつづく

```
myWiimote.addEventListener( WiimoteEvent.BALANCEBOARD_DISCONNECT,
                                                onBalanceBoardDisconnected );

// WiiFlash Serverにリモコンを接続
myWiimote.connect ();
//Synchronizeをインスタンス化
var mySynchronize:Synchronize = new Synchronize();
//Synchronizeをステージに配置
addChild( mySynchronize );
//Synchronizeの位置を画面中心へ
mySynchronize.x = (stage.stageWidth - mySynchronize.width) / 2;
mySynchronize.y = (stage.stageHeight - mySynchronize.height) / 2
```

重心移動の変化量を取得するのは以下の部分です。このUpdated関数がWiiBoardに変化が起こるたびに呼び出されます。

コード5-11　ActionScript Updated関数

```
function onUpdated ( pEvt:WiimoteEvent ):void {
 //バランスボードに掛かる重さ
 var kg:Number = int(pEvt.target.balanceBoard.totalKg);
 //バランスボードに掛かる重さが20kg以上であれば
 if(kg > 20){

 }
 //ターゲットの場所を移動
 pointTarget.x = pEvt.target.balanceBoard.centerOfGravity.x * 10 + 90;
 pointTarget.y = pEvt.target.balanceBoard.centerOfGravity.y * 20 + 115;
 //変化量を計測
 checkDiff(pEvt.target.balanceBoard.centerOfGravity.x, pEvt.target.
                                          balanceBoard.centerOfGravity.y);
}
```

そして、checkDiff関数でポジションを変更します。

コード5-12　ActionScript checkDiff関数

```
function checkDiff(gx, gy):void{
 xPos = gx;
 yPos = gy;
}
```

このサンプルでは、Timer関数を使って定期的に計算をしています。Updated関数でも、定期的に計算ができるのですが、このUpdated関数は呼び出される頻度がとても高いので、処理が重くなってしまうことがあります。そこで今回は、少し間隔をあけて実行するように、次のTimer関数を使用しました。

コード5-13　ActionScript　Timer関数

```
/**重心の移動変化を確認する間隔*/
var delay:uint = 3000;
/**重心の移動変化を確認する間隔*/
var repeat:uint = 0;
/**重心の移動変化を確認する間隔*/
var myTimer:Timer = new Timer(delay, repeat);
myTimer.addEventListener("timer", timerHandler);
myTimer.start();
```

実際に計算している部分はコード5-14に示したtimerHandler関数です。

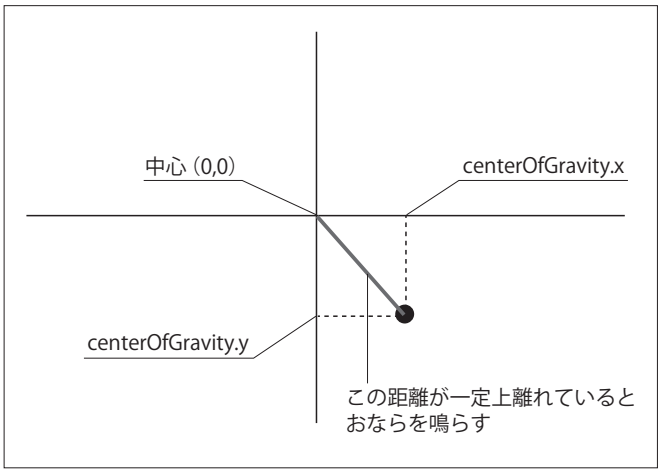
コード5-14　ActionScript　timerHandler関数

```
function timerHandler(e:TimerEvent):void{
 //変化量を計測
 var c:Number = Math.sqrt((xPos*xPos)+(yPos*yPos));
 //変化量を出力
 _txt.text = String(Math.floor(c));
 //変化量が規定値以上であればおならを鳴らす
 if(c > maxAmount){
   playFarts();
 }
}
```

重心はx軸とy軸のポイントからなる位置の値として取得することができます。このx軸とy軸の位置が中心点（0,0）のポイントから直線距離でどのくらい離れているかを計算し、その値を元に姿勢が崩れたかどうかを判断しています。

```
var c:Number = Math.sqrt((xPos*xPos)+(yPos*yPos));
```

図5-17　重心のズレから姿勢を判断



そして、playFarts関数でおならを鳴らしています。

コード5-15　ActionScript　timerHandler関数

```
/**
*  おならを鳴らす
*/
function playFarts():void{
 //鳴らす音を選択
 var soundName = farts_array[Math.floor(Math.random()*farts_array.length)];
 //音のファイルを取り出す
 var req:URLRequest = new URLRequest("se/"+soundName);
 //サウンドオブジェクトを初期化
 snd = new Sound(req);
 //鳴らす
 channel = snd.play();
}
```

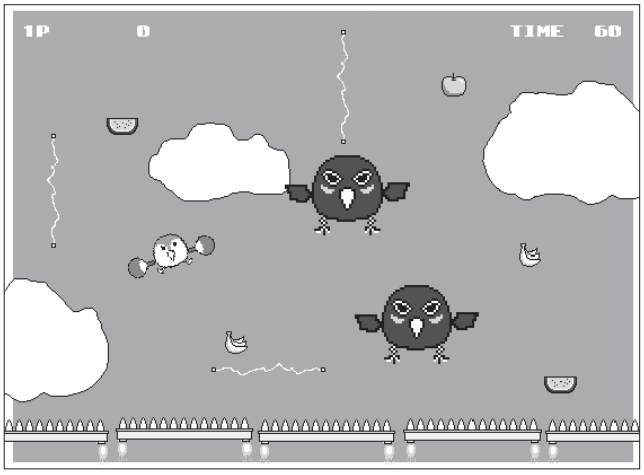
今回は、姿勢を強制するアプリとしていますが、同じように簡単にバランスゲームを作ることができます。Flashゲームの作り方は多くの書籍やWebサイトで公開されているので、そこにWiiRemoteを使った「ひと工夫」を入れることで、より楽しい作品が作れることでしょう。

5.3 作例2：ChunChun
制作／原央樹

WiiRemoteを使ったゲーム

ChunChunは、WiiRemoteを入力装置とするFlashゲームです。WiiRemoteとヌンチャクに内蔵されている加速度センサーを使ってスズメのChunChunを操作し、フルーツを集めるゲームです。

図5-18　ChunChunゲーム画面



ChunChunの遊び方

ChunChunは、でぶっちょなスズメなので、自分の小さな羽根だけでは飛ぶことができません。そこで、両手に団扇を持って飛ぶことを考えました。

プレイヤーはWiiRemoteとヌンチャクを左右の手に持ちます。右手のWiiRemoteを振るとChunChunは右側の団扇を仰ぎ、左上の方向にふわりと浮きます。逆に左手のヌンチャクを振ると左の団扇を仰ぎ、ChunChunは右上の方向にふわりと移動します。プレイヤーがWiiRemoteを操作しなければChunChunは落下してしまうので、団扇の仰ぎ具合を絶妙にコントロールし、一定時間内でできるだけたくさんのフルーツを集めてください。

筆者がこのゲームを作るきっかけになったのは、4歳になる息子が団扇で遊んでいるのをみたときです。彼は両手に団扇を持って羽のようにバタバタと動かし、一生懸命空を飛ぼうとしていました。

もちろん空を飛ぶことはできないのですが、彼は必死に団扇を動かしジャンプして「空を飛んだ」といいます。実は、僕も幼い頃、団扇で空を飛ぼうとしたことがありました。そして、彼と同じようなことを言っていたのだと思います。ということで、親子2代の夢を叶えるべく、このゲームを作成しました。

ゲームのルール

ChunChunは以下のルールに従って動作を行います。

- ChunChunは重力により落下します
- WiiRemoteを振ると、左上方向に力がかかります
- ヌンチャクを振ると、右上方向に力がかかります

WiiRemoteとヌンチャクをうまい具合にパタパタと振って羽ばたかなければ、すぐに落下してしまい、うまく飛行することができません。

それでは、次項からサンプル制作の流れ、ChunChunのプログラムを見ていきましょう。

サンプル制作の流れ

以下では、プログラムの該当のパートを引用して解説していきます。全体に関しては、配布するソースコードを参照してください。また、ここではChunChunの操作や動きに関することを中心に解説を進めます。WiiRemoteとの関連性の薄い箇所（あたり判定やスコアの管理など）は、サンプルプログラムのコメントのほうで確認してください。

WiiRemoteとFlashをつなぐ

WiiRemoteとFlashをつなぐためには、WiiFlashを使用します。前述のように、WiiFlashは、PCにBluetoothを使って接続したWiiRemoteの情報をFlashに受け渡すためのサーバと、その情報をFlashで簡単に扱えるようにしたActionScriptのクラス群で構成されています。

WiiRemoteとPCをBluetoothでつないだら、WiiFlashサーバを立ち上げてください。その後、

次のようにしてWiiRemoteをFlashで扱えるようにします。

コード5-16 ActionScript 初期設定（net.hara3.chunchun.WiimoteControl.as抜粋）

```
import org.wiiflash.Wiimote;
import org.wiiflash.events.*;
```

コード5-17 ActionScript Wiimoteとの接続（net.hara3.chunchun.WiimoteControl.as抜粋）

```
private function _wiimoteConnect():void{
 _wiimote = new Wiimote();//WiiRemoteのインスタンスを作成する
 _wiimote.addEventListener(Event.CONNECT, _onWiimoteConnect);
                   //WiiRemoteとの接続に成功したら_onWiimoteConnectメソッドを呼び出す
 _wiimote.connect();//WiiRemoteとFlashを接続する
}

//Wiimoteに接続成功したらタイトル画面へ
private function _onWiimoteConnect(e:Event):void {
 SceneManager.getInstance().go("title");
}
```

WiiRemoteの入力を受け取る

WiiRemoteからの入力にはさまざまなものがあります。

まずはAボタン、Bボタンなどの簡単なボタン入力の受け付け方法は次の通りです。

コード5-18 ActionScript ボタン入力の受け付け（net.hara3.chunchun.WiimoteControl.as抜粋）

```
public function title():void {
 _title_press_A = false;
 _title_press_B = false;

 _wiimote.addEventListener( ButtonEvent.A_PRESS, _onTitleAPress );
        //WiiRemoteのAボタンをPressしたときに、_onTitleAPress()メソッドを実行する
 _wiimote.addEventListener( ButtonEvent.A_RELEASE, _onTitleARelease );
        //WiiRemoteのAボタンをReleaseしたときに、_onTitleARelease()メソッドを実行する
 _wiimote.addEventListener( ButtonEvent.B_PRESS, _onTitleBPress );
        //WiiRemoteのBボタンをPressしたときに、_onTitleBPress()メソッドを実行する
 _wiimote.addEventListener( ButtonEvent.B_RELEASE, _onTitleBRelease );
        //WiiRemoteのBボタンをReleaseしたときに、_onTitleBPress()メソッドを実行する
}
```

では次に、WiiRemoteからの加速度センサーの値を取得してみます。加速度センサーの値を使って、WiiRemoteを振ったときにどれくらいの速さで振ったのかを調べることができます。WiiRemoteが激しく振られたかどうか、マグニチュードを計算して判定します。その計算結果から、WiiRemoteが激しく振られたことがわかったら、ChunChunに浮力を与えます。

コード5-19　ActionScript　加速度センサーの値からChunChunに浮力を与える（net.hara3.chunchun.WiimoteControl.as抜粋）

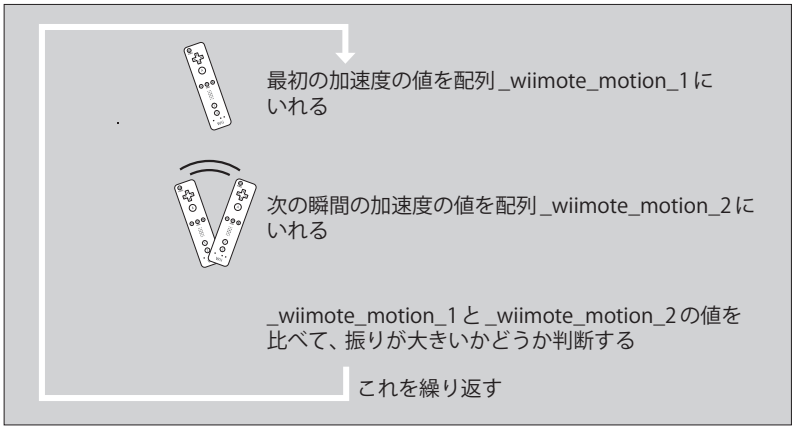
```
public function game(chara_mc:MovieClip):void {

 _chara_mc = chara_mc;//キャラクターを初期化

 _wiimote_motion_1 = new Array(0,0,0);
                //WiiRemoteの加速度を取得するための配列を初期化
 _wiimote_motion_2 = new Array(0, 0, 0);
                //WiiRemoteの加速度を取得するための配列を初期化

 _nunchuk_motion_1 = new Array(0,0,0);
                //WiiRemoteの加速度を取得するための配列を初期化
 _nunchuk_motion_2 = new Array(0, 0, 0);
                //WiiRemoteの加速度を取得するための配列を初期化

 _wiimote.addEventListener(WiimoteEvent.UPDATE,_gameUpdate);
                //WiiRemoteの状態が変化したら、_gameUpdate()メソッドを実行(これを
                                                    使って加速度センサーからの情報を監視)
}

//WiiRemoteの状態が変化するたびに呼ばれる
private function _gameUpdate(e:WiimoteEvent):void {
 //リモコン
 _wiimote_motion_1 = _wiimote_motion_2;
                //以前取得した加速度の値を保存しておく
 _wiimote_motion_2 = [e.target.sensorX, e.target.sensorY, e.target.sensorZ];
                //WiiRemoteの加速度の値を配列で取得。X軸・Y軸・Z軸の加速度が取得できる。

 //以前取得した加速度の値と現在取得した加速度の値からマグニチュードを計算(*1)
 var mag:Number = _magnitude(_wiimote_motion_1, _wiimote_motion_2);
 //マグニチュードの値から羽ばたきが十分かどうか判定(*2)
 if (mag > 0.5) {
   _chara_mc.followR();//マグニチュードが0.5より大きければ、右の羽を振る
 }else {
   _chara_mc.stopR();//マグニチュードが0.5以下であれば、右の羽を止める
 }

 //ヌンチャク
```

次ページにつづく

```
_nunchuk_motion_1 = _nunchuk_motion_2;//以前取得した加速度の値を保存しておく
_nunchuk_motion_2 = [e.target.nunchuk.sensorX, e.target.nunchuk.sensorY,
                                                    e.target.nunchuk.sensorZ];
                  //ヌンチャクの加速度の値を配列で取得。X軸・Y軸・Z軸の加速度が取得できる。
//以前取得した加速度の値と現在取得した加速度の値からマグニチュードを計算(*1)
mag = _magnitude(_nunchuk_motion_1, _nunchuk_motion_2);
//マグニチュードの値から羽ばたきが十分かどうか判定(*2)
if (mag > .75) {
   _chara_mc.followL();//マグニチュードが0.75より大きければ、左の羽を振る
}else {
   _chara_mc.stopL();//マグニチュードが0.75以下であれば、左の羽を止める
}
}

// マグニチュード計算(1*1)
private function _magnitude(list1:Array, list2:Array):Number {
 var d_x:Number = Math.pow((list1[0] - list2[0]),2);
 var d_y:Number = Math.pow((list1[1] - list2[1]),2);
 var d_z:Number = Math.pow( (list1[2] - list2[2]),2);
 return (d_x + d_y + d_z);
}
```

マグニチュード計算では、WiiRemoteがどれだけ激しく振られたかを計算します。
計算方法は次の図の通りです。

図5-19　マグニチュードの計算方法



・X軸に関して

dx = √(加速度センサーのX方向に現在かかっている速度 - 加速度センサーのX方向に少し前にかかっていた速度)^2

・Y軸に関して

dy = √(加速度センサーのY方向に現在かかっている速度－加速度センサーのY方向に少し前にかかっていた速度)^2

・Z軸に関して

dz = √(加速度センサーのZ方向に現在かかっている速度－加速度センサーのZ方向に少し前にかかっていた速度)^2

・XYZ全方向に関して

dxyz = dx + dy + dz

ただし、WiiRemoteとヌンチャクでは中に入っている加速度センサーの種類が異なるようです。そのため、このサンプルではマグニチュードの判定基準をWiiRemoteでは0.5、ヌンチャクは0.75としています。これは、実際に操作して違和感があるようなら値を調整してみてください。

ChunChunの動作

ChunChunは基本的に自由落下します。WiiRemoteからの羽ばたきを受け付けたときに、浮力を得て移動することができます。

まずは、ChunChunを自由落下する部分を見ていきます。自由落下の計算式は用途によってさまざまな形がありますが、今回は最も簡単なオイラー法を用いた計算を行いたいと思います。

たとえば、

_G：重力加速度
_ax：X軸方向に常にかかる加速度
_vy：物体のY軸方向にかかる力
_vx：物体のX軸方向にかかる力
_dT：単位時間（どれくらいの時間の精度で計算するか）

としたときの、物体の座標（x,y）は、

```
_vy += _G * _dT;
_vx += _ax * _dT;
x += _vx * _dT;
```

y += _vy * _dT;

となります。

図5-20　ChunChunの自由落下

これをそのままプログラムに反映します。

コード5-20　ActionScript　ChunChunの自由落下（net.hara3.chunchun.WiimoteControl.as抜粋）

```
//重力計算用のプロパティ
private var _G:Number = 200;
                    //重力加速度(Y軸方向に常にかかる加速度)
private var _ax:Number = 0;
                    //X軸方向に常にかかる加速度(今回はX軸方向に余計な力はかからないので0)
private var _vx:Number = 0;//ChunChunのX方向にかかる力
private var _vy:Number = 0;//ChunChunのY方向にかかる力
private var _dT:Number = 1 / 20;//単位時間(1/フレームレート)

157-169行目
//オイラー法により自由落下運動させる
_vy += _G * _dT;//Y軸の加速度の計算
_vx += _ax * _dT;//X軸の加速度の計算

x += _vx * _dT;//加速度からChunChunのX座標の計算
y += _vy * _dT;//加速度からChunChunのY座標の計算
```

次に、羽ばたいたときに任意の方向に力を加えてみたいと思います。

WiiRemoteを激しく振ったときには、ChunChunの左上方向（たとえば[x:-5, y:-15]）に力が加わるので、その値を羽ばたきの力として_vx、_vyに加算します。

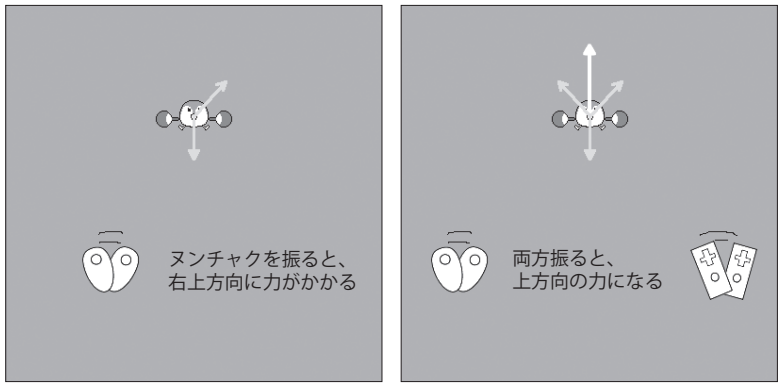
図5-21　右の羽ばたきの動き

コード5-21　ActionScript 右の羽ばたきの動き（net.hara3.chunchun.WiimoteControl.as抜粋）

```
public function followR():void {
 _wing2_mc.followStart();
 //画面の高い位置の場合は羽ばたき無視
 if (y <= 20) {
    return;
 }

 _vy += -15;
 _vx += -5;
}
```

同様にヌンチャクを激しく振ったときには、ChunChunの右上方向（たとえば[x:5, y:-15]）に力が加わるので、その値を羽ばたきの力として_vx、_vyに加算します。

図5-22　左の羽ばたきの動き



コード5-22 ActionScript 左の羽ばたきの動き（net.hara3.chunchun.WiimoteControl.as 抜粋）

```
public function followL():void {
 _wing1_mc.followStart();
 //_wing1_mc.play();

 //画面の高い位置の場合は羽ばたき無視
 if (y <= 20) {
   return;
 }

 _vy += -15;
 _vx += 5;
}
```

もし、ChunChunに加える力が大きすぎたり小さすぎたりして、思ったような操作感が得られない場合は、_vx、_vyに加える力を少し調整してみてください。

いかがでしょうか。

ここまでのプログラムにより、FlashがWiiRemoteの羽ばたきを感知し、落下しているChunChunの_vx、_vyに力を加え、ChunChunがふわりと浮く感じが表現できたと思います。

あとはさまざまなオブジェクトを配置し、たとえばフルーツを取るとスコアがアップする、というようなゲームとしての仕様を追加実装してみてください。