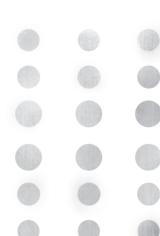
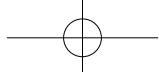


# 9

## アイディアストック・ 演習問題集





この章は「アイディアストック・演習問題集」として、今まで学んだ技術を応用することで、WiiRemoteを使って実現できるさまざまなプロジェクトの実例を紹介します。

アイディアをためておく「棚」のようなものをイメージして「アイディア・ストック」と名付きました。イメージしやすいようにプログラミング編、モノ編、ゲーム応用編、サービス編、作品編、研究編に分けていますが、このアイディアの活用は読者のみなさん次第でさまざまな方向に混ざったり、生み出されたりしていくことでしょう。

本書をここまで読み進めてきた読書であれば、「不可能なほど難しい」という内容ではないはずです。本章では細かいステップバイステップの解説をあえて割愛し、少ない紙面で幅広く WiiRemoteの可能性を伝えることに力点を置きます。

各節の終わりに「演習問題」として、そのテーマの研究や作品作りに役立つ問題集を用意しておきました。難易度が5段階の☆で表現されていますので、難易度に合わせて授業や課題の制作、論文のリファレンスなどにご活用ください。

## 9.1

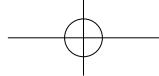
# プログラミング編:XNAを使った リアルタイム3DCGでの利用

第8章までの知識を一步進めて、WiiRemoteを3Dグラフィックスプログラミングの世界で利用できるようになります。マイクロソフトの本格的なゲーム用3DCGプログラム開発環境「XNA」と「WiimoteLib」を使います。

ここでは「XNA Game Studio 3.1」とWiimoteLibを使って、C#.NETによるゲーム開発環境をベースにしたリアルタイム3DCGによるプログラミングを解説します。

XNAとは、マイクロソフトが推進している「DirectX」の流れをくむ最新の.NETによるゲーム開発統合環境です。XNAのコーディングスタイルは、旧来のリアルタイム3DCG開発環境の本流であった DirectX や Managed DirectX とは異なり、XNA FrameworkにおけるC#言語による開発になります。DirectX時代よりもさらにゲーム開発に便利なツールやAPIが統合されており、簡単に効率よくゲームプログラムを作成できるようになっています。

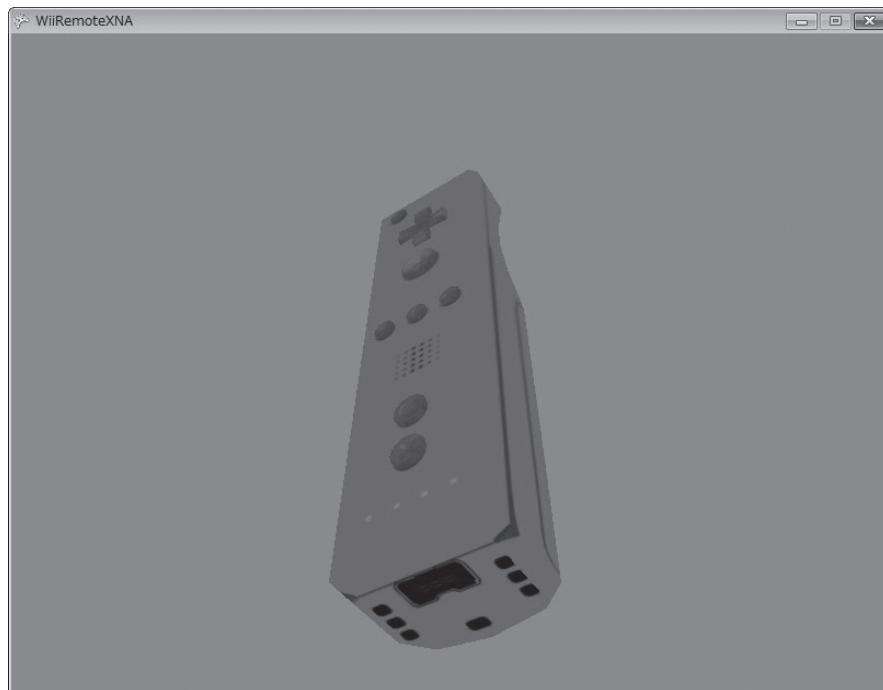
プロのゲーム開発者に限らず、学生などにも親しみやすい環境でもあります。WindowsPC用のゲーム開発に加え、最新のコンシューマー（家庭用）ゲーム機である「Xbox 360」の両方のプ



ラットフォームで、非常に効率的かつ先進的な開発ができるため、プロのゲームスタジオだけでなく、今後ホビープログラマを中心に大きな流れを作り出す可能性があるでしょう。

この節では、WiiRemoteの加速度センサーの傾きによって、3Dで描画されたWiiRemoteがリアルタイムで変化するプログラム「WiiRemoteXNA」を作成します。なかなか派手な感じがするデモですが、XNA Game Studio 3.1を使って、驚くほど短いコードで作成することができます。

図9-1 「WiiRemoteXNA」完成版



## XNAのインストール

まずは、開発環境のセットアップを行いましょう。最新のMicrosoft XNA Game Studio 3.1(無償)をダウンロードしてインストールします。

### Microsoft XNA Game Studio 3.1

URL <http://www.microsoft.com/downloads/details.aspx?familyid=80782277-D584-42D2-8024-893FCD9D3E82>

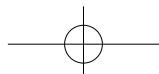
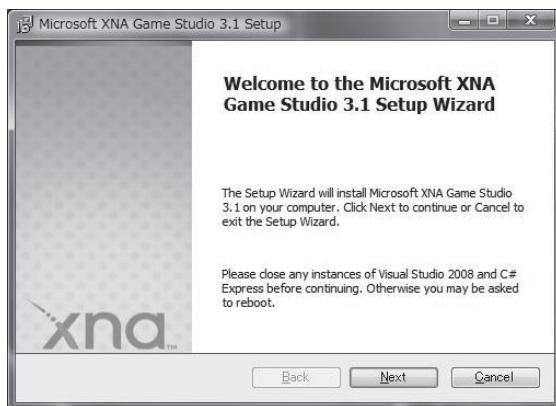


図9-2 XNA Game Studio 3.1のインストール



Microsoft XNA Game Studio 3.1のインストールは、ダウンロードしたインストーラーのウィザードに従うだけで問題なく行えるでしょう。ウィザードの最後に「Xbox 360用のサービスを起動するか?」という質問がありますが、これはXbox 360用のプログラムを開発したときに、ローカルネットワーク経由でXbox 360に送信するためで、特に利用する予定がなければチェックは入れなくても問題ありません。

#### Xbox 360用ゲームの開発

XNAは無料で開発環境を手に入れることができます。ただし、PCをターゲットプラットフォームとして利用する上ではライセンスに従い無料で利用することができますが、Xbox プラットフォームで開発するためには、年間ライセンス料(9,800円<sup>※1</sup>)を払う必要があります。

本書では、Xbox プラットフォームについては扱いませんが、高価なゲーム用PCに比べて、Xbox 360のような「安定して安価で入手できるコンシューマゲーム機」の開発環境が、それほど高額ではないライセンス料で入手できるのは大変な魅力です。開発したゲームプログラムを、世界中に1200万人以上いるXbox 360のユーザーに遊んでもらえることも、モチベーションになるでしょう(Xbox360でWiiRemoteが公式に使える、という話は聞きませんが…)。

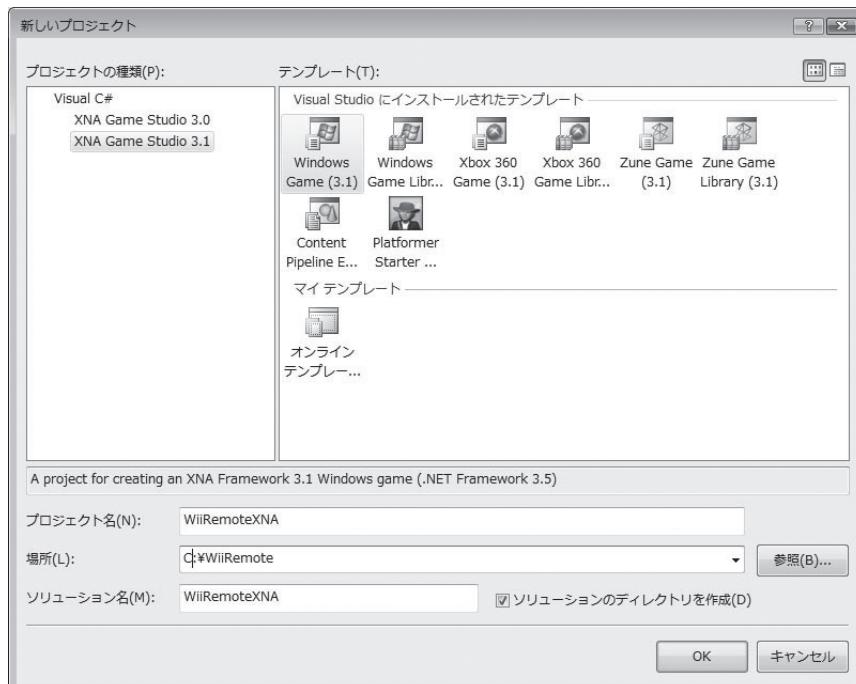
## ゲームプロジェクトの作成

次はゲームプロジェクトの作成です。Microsoft Visual C# 2008を起動してください(無料の

「Express Edition」でも問題なく利用できます)。「新しいプロジェクト」を選ぶと、いつも見慣れた新規プロジェクト作成のダイアログに「XNA Game Studio 3.1」という項目が現れているはずです。ない場合はGame Studioのインストールを再度確認してください。

「テンプレート」から「Windows Game (3.1)」をクリックして、プロジェクト名に「WiiRemoteXNA」という名前をつけて、場所を「C:\WiiRemote」として「OK」をクリックします。

図9-3 新規プロジェクトの作成にGame Studio 3.1のテンプレートが現れる



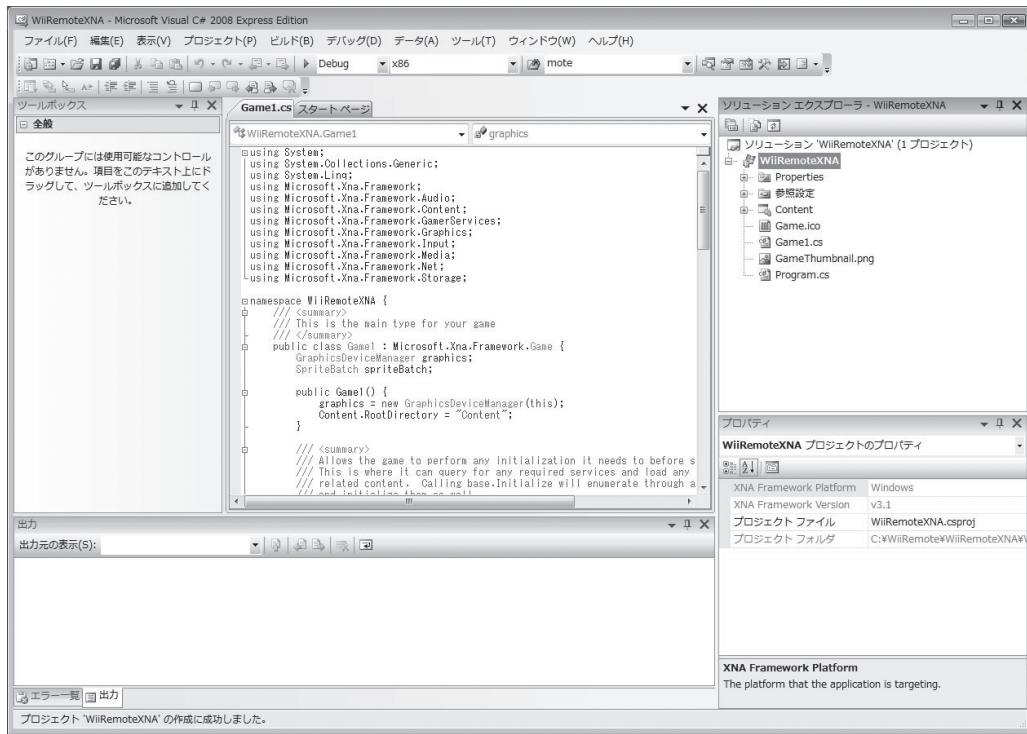
数秒間待つと、新しいプロジェクトが作成されます。「F5」キーを押して、試しにプロジェクトを実行してみましょう。水色の背景に、何かウィンドウが表示されれば成功です。

※1：本書とは直接関係ありませんが、Xbox 360用ゲームを実行するためには年額9,800円の「XNAクリエイターズクラブ」に入会する必要があります。XNAクリエイターズクラブはXbox 360用ネットワーク・サービス「Xbox Live」から加入できます。

<http://www.xbox.com/ja-JP/live/>

URL [http://creators.xna.com/en-US/tour\\_detail](http://creators.xna.com/en-US/tour_detail)

図 9-4 Visual Studio に作成された新しいゲームプロジェクト



## WiimoteLib の組み込み

次は WiimoteLib を組み込みます。XNA 環境でも、第 4 章や第 8 章での .NET 環境における WiimoteLib の組み込み作業の流れと同じです。

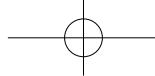
ソリューションエクスプローラの「参照設定」を右クリックし、「参照の追加」を選択します。参照の追加から「参照」もしくは「最近使用したファイル」から「WiimoteLib.dll」を選択します。

今までのプロジェクトと同様、プログラム冒頭の using に WiimoteLib を追加し、クラスの初期化時に Wiimote オブジェクトの新規作成「Wiimote wm = new Wiimote();」を挿入します。

### コード 9-1 C# Wiimote オブジェクトの作成 (Game1.cs)

```
using WiimoteLib;
<略>
public class Game1 : Microsoft.Xna.Framework.Game {
```

[次ページにつづく](#)



```
GraphicsDeviceManager graphics;
SpriteBatch spriteBatch;
Wiimote wm = new Wiimote(); //Wiimoteオブジェクトの作成
<略>
```

初期化時にWiiRemoteに接続しましょう。XNAフレームワークでは「Initialize()」という関数がすでに用意されていますので、そこにいつものWiiRemote接続処理を追加します。

#### コード9-2 C# WiiRemoteの初期化と接続 (Game1.cs)

```
protected override void Initialize() {
    base.Initialize();
    wm.Connect(); //WiiRemote接続
    wm.SetReportType(InputReport.ButtonsAccel, false); //ボタンと加速度
    wm.SetLEDs(15); //LED全点灯
}
```

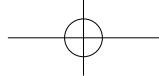
「SetReportType()」の第2引数に「false」を設定して非連続データ取得モードにしています。ここで、いつものようにtrueにして、コールバック関数を設定してもよいのですが、今回のサンプルでは簡単に加速度の値を取れればよいので、値のばたつきが少ない、よりシンプルな方法をとることにします。

この状態でもビルド処理を試すことはできますが、ButtonStateが「あいまいな参照」というエラーを出し停止するはずです(XNAとWiimoteLibに同じ名前のプロパティがあるため)。エラーの出る行をコメントアウトすればよいのですが、オリジナルのソースでは、ここで「ゲームパッドのボタンが押されたら終了」となっているようですので、ここを「WiiRemoteのHomeボタンが押されたら終了」と変更してみましょう。

#### コード9-3 C# Homeボタンで終了 (Game1.cs)

```
protected override void Update(GameTime gameTime) {
    // Allows the game to exit
    // if (GamePad.GetState(PlayerIndex.One).Buttons.Back
        == ButtonState.Pressed)
        if(wm.WiimoteState.ButtonState.Home) {
            this.Exit();
        }
    base.Update(gameTime);
}
```

もともとのコードを  
コメントアウト

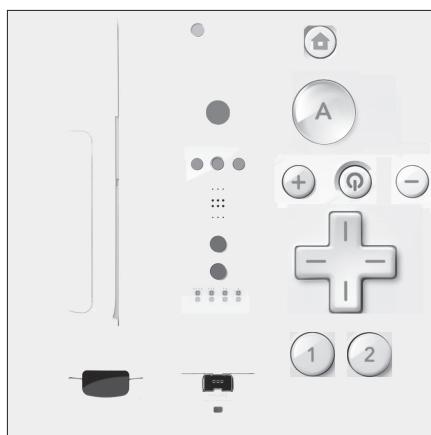


この段階でWiiRemoteをBluetooth接続し、「F6」キーで実行してみてください。先ほどと同様、水色の画面が表示されますが、WiiRemoteのLEDが4つとも点灯し、Homeボタンを押すことでプログラムを終了できるはずです。

## 3Dモデルファイルの準備

次に、読み込む3Dモデルファイルを準備しましょう。3Dモデルデータをゼロから作ると時間がかかりますので、ここでは小坂研究室のホームページからWiiRemoteによく似た「.x形式」のモデルファイル「wiimodoki.x」と、その表面を飾るテクスチャファイル「texture\_wii.jpg」をダウンロードで入手します。

図9-5 特製テクスチャファイル(texture\_wii.jpg)

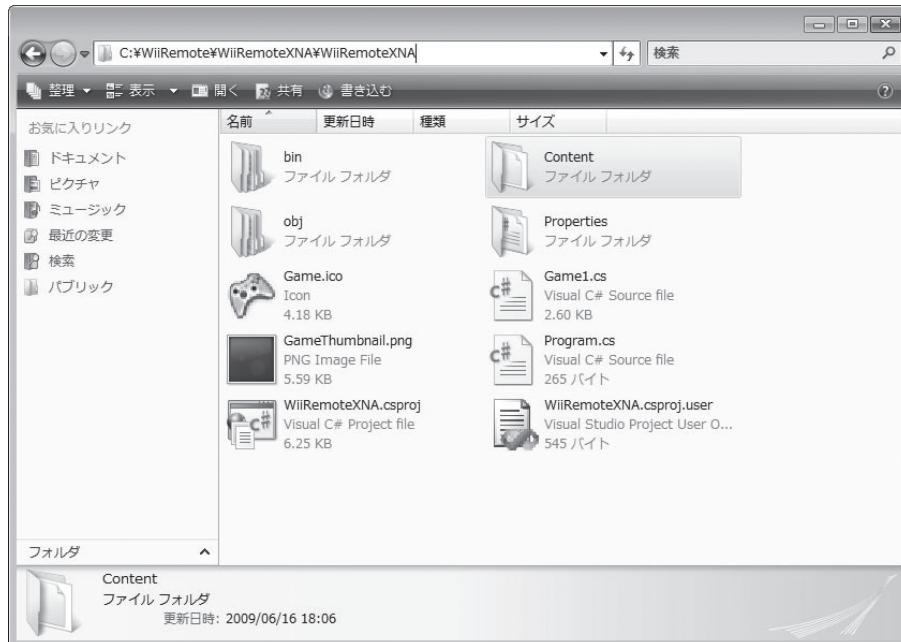


小坂研究室「XNAとWiimoteLibで3Dオブジェクトを操作

URL <http://www.kosaka-lab.com/tips/2009/06/xnawiimotelib3d.php>

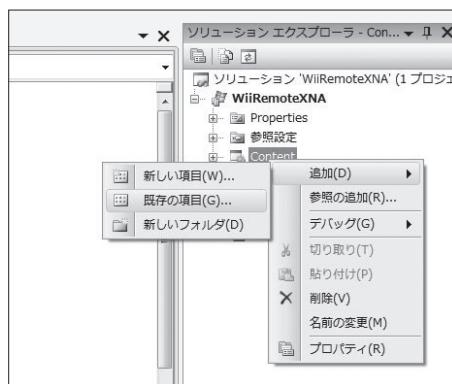
ファイルをダウンロードしたら、XNAのプロジェクトのコンテンツフォルダ「C:\WiiRemote\WiiRemoteXNA\WiiRemoteXNA\Content」に置きます。

図9-6 WiiRemoteXNAの「Content」に.xファイルを配置する

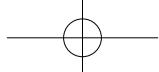


ファイルを置いた後に、Visual Studio内に取り込みます。ソリューションエクスプローラーの「Content」を右クリックして「追加」→「既存の項目」として、先ほど置いた「wiimodoki.x」と「texture\_wii.jpg」を読み込んでください。

図9-7 ソリューションエクスプローラーの「Content」に追加する



なお、モデルデータである「wiimodoki.x」と「texture\_wii.jpg」は、3DCGコンテンツ制作ソフトウェア「Maya2008」を使って作成した後、「cvXporter」を使って.xファイルにコンバートしています。

**cvXporter**

URL <http://www.chadvernon.com/blog/downloads/cvxporter/>

Maya2008のような比較的高価な3DCGソフトウェアが無ければ、日本人が開発している歴史ある3Dポリゴンモデルer「Metasequoia」(メタセコイヤ)でも可能でしょう。無料版と有料(シェアウェア)版があり、価格は1ライセンスにつき5,000円です。「Metasequoia LE R2.4」では.xファイルを直接書き出せますので、XNAやDirectX環境で簡単に利用することができます。作者のO.Mizno氏に感謝です。

**Metasequoia**

URL <http://www.metaseq.net/>

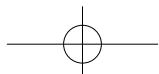
## .xファイルの読み込みと表示

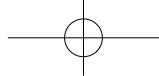
さて、コーディングに戻りましょう。モデルファイルの読み込みと表示について、「LoadContent()」と「Draw()」に加筆をします。

コード9-4 C# モデルファイルの読み込みと表示 (Game1.cs)

```
<略>
Wiimote wm = new Wiimote(); //Wiimoteオブジェクトの作成
private Model xfile;           //Xファイル読み込み用
<略>
protected override void LoadContent() {
    spriteBatch = new SpriteBatch(GraphicsDevice);
    this.xfile = this.Content.Load<Model>("wii"); // .xファイルの読み込み
    foreach (ModelMesh mesh in this.xfile.Meshes) {
        foreach (BasicEffect effect in mesh.Effects) {
            //ビュー行列 カメラの視点を設定(0.0f,0.0f,10.0f)の位置から原点を見る
            effect.View =
                Matrix.CreateLookAt(new Vector3(0.0f, 0.0f, 10.0f),
                Vector3.Zero, Vector3.Up);
            //プロジェクション行列 視野角などの設定
            effect.Projection = Matrix.CreatePerspectiveFieldOfView(
                MathHelper.ToRadians(45.0f),
                (float)this.GraphicsDevice.Viewport.Width /
                (float)this.GraphicsDevice.Viewport.Height,
                1.0f, 50.0f );
```

次ページにつづく↗





```
        }
    }
}

<略>
protected override void Draw(GameTime gameTime) {
    GraphicsDevice.Clear(Color.CornflowerBlue);
    //画面に描画する
    foreach (ModelMesh mesh in this.xfile.Meshes) {
        foreach (BasicEffect effect in mesh.Effects) {
            //WiiRemoteの加速度に合わせて回転角度を設定
            effect.World = Matrix.CreateFromYawPitchRoll(0,
                -wm.WiimoteState.AccelState.Values.Y,
                -wm.WiimoteState.AccelState.Values.X);
        }
        mesh.Draw(); //meshを描画
    }
    base.Draw(gameTime);
}
<略>
```

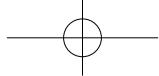
ここでは個々のAPIについて解説はしませんが、いずれも3DCGにおけるお作法的な手続きと「見え方」を設定しているものです。興味がある人は、MSDNなどのマニュアルを調べたり、パラメーターを変更してみたりして、探求してみてください。

実行すると、WiiRemoteの動きに合わせて回転する「WiiRemoteもどき」が表示されます。Homeボタンを押すと終了します。

図9-8 WiiRemoteの動きに合わせて動く「WiiRemoteもどき」



ファイルの読み込みなどでエラーが起きるときは、ソリューションエクスプローラーで正し



くContentにファイルが取り込まれているか確認してください。なお、テクスチャファイルは「wiimodoki.x」の中で記述されていますので、実はVisual Studioに取り込まなくても自動で読み込まれます。また「wiimodoki.x」ファイルはテキストで記述されていますので、テキストエディタで編集することで、マテリアル（表面材質の特性）やテクスチャファイル名を書き換えたりすることができます。

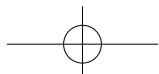
## 完成版「WiiRemoteXNA」

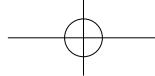
以上で「WiiRemoteXNA」は完成です。usingの整理など行った完成版のコードを紹介します。

コード9-5 C# 完成版「WiiRemoteXNA」(Game1.cs)

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using WiimoteLib;
namespace WiiRemoteXNA {
    public class Game1 : Microsoft.Xna.Framework.Game {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        Wiimote wm = new Wiimote(); //Wiimoteオブジェクトの作成
        private Model xfile; //Xファイル読み込み用
        public Game1() {
            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
        }
        protected override void Initialize() {
            base.Initialize();
            wm.Connect(); //Wiimote接続
            wm.SetReportType(InputReport.ButtonsAccel, false);
            wm.SetLEDs(15);
        }
        protected override void LoadContent() {
            spriteBatch = new SpriteBatch(GraphicsDevice);
            xfile = Content.Load<Model>("wii");
            foreach (ModelMesh mesh in this.xfile.Meshes) {
                foreach (BasicEffect effect in mesh.Effects) {
                    effect.View =
                        Matrix.CreateLookAt(new Vector3(0.0f, 0.0f, 10.0f),
                        Vector3.Zero, Vector3.Up);
                    effect.Projection = Matrix.CreatePerspectiveFieldOfView(
```

次ページにつづく↗





```
        MathHelper.ToRadians(45.0f),
        (float)this.GraphicsDevice.Viewport.Width
        / (float)this.GraphicsDevice.Viewport.Height,
        1.0f, 50.0f );
    }
}
}
protected override void UnloadContent() { ; }
protected override void Update(GameTime gameTime) {
    if(wm.WiimoteState.ButtonState.Home) { this.Exit(); }
    base.Update(gameTime);
}
protected override void Draw(GameTime gameTime) {
    GraphicsDevice.Clear(Color.CornflowerBlue);
    foreach (ModelMesh mesh in this.xfile.Meshes) {
        foreach (BasicEffect effect in mesh.Effects) {
            //WiiRemoteの加速度に合わせて回転角度を設定
            effect.World = Matrix.CreateFromYawPitchRoll(0,
                -wm.WiimoteState.AccelState.Values.Y,
                -wm.WiimoteState.AccelState.Values.X);
        }
        mesh.Draw(); //meshを描画
    }
    base.Draw(gameTime);
}
}
}
```

## XNAの利用

## PRACTICE

### 【演習】☆

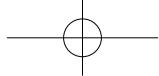
上記のプログラムをコールバックを使う方法に書き換えてみよ。

余裕があれば、「リスト」を使って最近50回の加速度センサーのデータの平均を取得し、よりなめらかに回転する WiiRemoteXNA を作成してみましょう（解答例は小坂研究室のホームページで紹介されています）。

### 【演習】☆☆

自作の.xファイルを読み込んでみよ。

Metasequoiaを使って、自分で好きな.xファイルをテクスチャとともに作成し、読み込めるようにしてみましょう。余裕があれば+ボタン、-ボタンでモデルデータを切り替えたり、十字ボタンでカメラのズームをしたりといった「3Dモデルビューア」としての機能を加えてみましょう。



リアルタイム3DCGという見た目の派手さに対して、とてもコードが短いことに驚かれたのではないかでしょうか(XNAのおかげです)。

なお、今回はコールバックを使わずに、描画ループで直接WiiRemoteの加速度センサーの値をそのまま回転の角度に利用するという、ちょっと荒っぽい方法をとっています。実際のゲームに使う場合には、このような使い方をすることは稀で、コールバックと動作認識関数などを作るべきでしょう。

## 9.2

### ゲーム応用編(1)： レースゲームへの応用

この節では、筆者が開発に協力した反重力レーシングゲーム製品「AceSpeeder2」をWiiRemoteでプレイできるように実験した例を紹介します。WiiRemoteは任天堂自身「マリオカート」などを発売していることもあり、レースゲームへの利用は想定されているようで、親和性高く利用できます。

#### AceSpeeder2

「AceSpeeder2」は2007年に発表された「RAINGRAPHスタジオ」ナカタニタカヒロ氏によるゲーム作品で、シェアウェアとして非常に人気が高かった初代「AceSpeeder」(2000年)の続編となる超高速SFレーシングゲームです。

図9-9 反重力レーシングゲーム「AceSpeeder2」



RAINGRAPHスタジオ

URL <http://www.raingraph.com/>

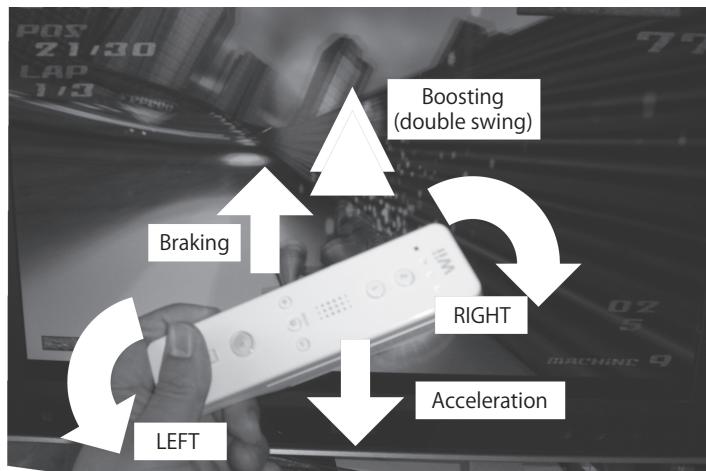
ゲームシステムはDirectX8ベースで開発されており、DirectXのジョイスティック機能であるDirectInputをベースに自機の制御を行っていました。

筆者はナカタニタカヒロ氏の協力により、すでにGPUを使った高速な全身画像認識「GPUVision」や、「OpenCV」を使った顔画像入力による、マルチモーダルなレーシングゲームコントロールの研究をAceSpeeder2のソースコードを用いて行っていました。その流れで、WiiRemoteによる新しい操作方法を実験してみました。

## レースゲーム用モーション認識

ゲームのソースコードに関わる部分ですので、プログラミングの詳細を解説しませんが、プレイヤーインタラクションとしては「左右は傾き、前後はアクセル/ブレーキ、2回振るとブースト発動」という操作体系にしてあります。

図9-10 WiiRemoteによるコントロール



すべて加速度センサーによる重力検出だけで実現しており、ボタンはメニュー選択を含め、全く使用しません。またLEDを「自機シールドの残量」として表示したり、バイブレーターを演出に使ったりと、WiiRemoteを最大限に活用しています。

AceSpeeder2では「ブースト」という機能があり、エネルギーをためることで大きな加速力を得ることができます。また、コースアウト時にもブーストで復旧できる技があるので、いつでもプレイヤーの意志で「ブースト発動！」できることが爽快感につながります。このブーストを「2回振る」というアクションは加速度センサーのマグニチュードを取ることで実現しています。

#### AceSpeeder2 WiiMedia Edition (動画)

URL <http://www.youtube.com/watch?v=KowXAXdfO8E>

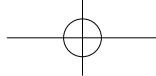
#### 「SIGGRAPH ビデオゲームシンポジウム：Sandbox」での受賞

筆者は2007年に、この実験とWiiRemoteの赤外線センサーの特性実験などを「WiiMedia: motion analysis methods and applications using a consumer video game controller」という論文にまとめています。

この論文は反響が高く、アメリカで毎年開催されるCGとインタラクティブ技術の世界最高の国際会議「SIGGRAPH」におけるビデオゲームシンポジウム「Sandbox」において、最優秀論文賞をいただきました。ソースコードは公開し、研究は発表しておくものだなど、つくづく思います。

#### SIGGRAPH

URL <http://sandbox.siggraph.org/about.html>。



WiiRemoteのおかげで、ボタンを全く使わない操作体系になり、小さな子どもでも体験することができます。

また、実際の展示を通したユーザーテストでの観察によると、おもしろいことがわかりました。ブーストは開発者の意図通り、振って発動する場合のほかに、コースの切れ目を避けるために無意識に「ジャンプ」したときに発動したり、コースアウトして「うわっ！落ちる!!」とのけぞった瞬間、無意識で振ったきっかけで発動したりと、より全身で直感的に楽しめるゲームになりました。

残念ながら当時の WiiRemote の Bluetooth 接続はそれほど安定した環境ではありませんでしたし、本書のような解説書を出版するということを考えていませんでしたので、このバージョンは製品としては公開されませんが、それほど難しいことをしたわけではありません。これからは、もっと多くの PC レーシングゲームで WiiRemote を活用してほしいと思います。

### レースゲームへの応用

### PRACTICE

加速度センサーによる傾きの検出は Atan20 を使えば簡単に求めることができます。マグニチュードの算出などもすでに第7章、第8章などで扱っていますので参照してください。さっそく課題の設定に入りましょう。

#### 【演習】☆☆

GlovePIE を使って傾きをアナログ入力に利用せよ。

誰もが「AceSpeeder2」のようなすばらしいゲーム作品のソースコードにアクセスできるわけではありません。しかしフリーでソースコードを公開されているゲームプロジェクトは SDL 関係では意外に多く、有名なところでは「Tux Racer」という Linux ペンギンのレースゲームなどもソースが公開されているプロジェクトです。

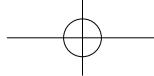
ソースコードがどうしても手に入らない、ということは逆転の発想で、第3章で学んだ GlovePIE を使いましょう。プログラミングが不要ということはソースコードの入手も不要なのです。うまくジョイパッドをエミュレーションするコードを書きましょう。アナログ→デジタル入力でも、うまく作ると自然な体験を作ることもできます。

#### 【演習】☆☆☆☆

自分のゲームプロジェクトに WiiRemote による操作を組み込み活用せよ。

参考までに、既存のレースゲームプロジェクトを WiiRemote 対応させる上で、難しかった点をメモしておきます。

多くのゲームの場合は DirectX のジョイスティック API である DirectInput に従った仕様に



なっているはずです（便利なので）。WiiRemote専用のゲームならいいのですが、PCゲーム開発としては、ゲームバランスや他のコントローラーでの操作感を壊さずに、他のゲームコントローラーと同様の感覚で、うまくなじませることが必要になります。

「AceSpeeder2」の場合も、元のゲームが細かい操作のためにデジタルジョイパッドを想定して設計されていたので、WiiRemoteの傾きをアナログジョイスティックとして当てはめると、大きく動きすぎたり、細かい動きができなかったりとより難易度が上がってしまいました。

難易度が上がるだけなら調整すればよいのですが、「傾ける」というプレイヤーの物理的な行動には制約がありませんので、ゲーム内に働く物理（速度や舵など）といいかに親和性を保ちながら、インターフェクションを向上させるかといった点も大きな課題となりました。

ちょっとしたアイディアとしては、「ゲーム内の物理」と「実際のユーザーの姿勢」を対応させるための「理想の姿勢」というものを考え、係数や式といった数値で扱う方法があります。ここをチューニングしていくことで、元のゲームプログラムを壊さずに、より操作感の向上に注力できるはずです。

#### **Wii Sports Resort「ウェイクボード」**

任天堂からリリースされた最新のレースゲームでの例としては「Wii Sports Resort」の「ウェイクボード」が良くできています。物理シミュレーションを使った美しいCGもさることながら、SFレーシングとは異なり「上に振ってジャンプ」という動作に加えて、「着地のときはWiiRemoteを水平に保つ」というルールを加えることで、WiiRemoteを使った操作とゲーム性を爽快感とともに見事に昇華しています。

#### **Wii Sports Resort「ウェイクボード」(動画あり)**

**URL** [http://www.nintendo.co.jp/wii/rztj/resort/02\\_sports/index.html](http://www.nintendo.co.jp/wii/rztj/resort/02_sports/index.html)

## 9.3

## ゲーム応用編(2)： 「振る」の認識・剣術アクションへの応用

ゲームでのWiiRemote応用(といっても、もともとゲーム用コントローラーですが)を考える上で、レースゲームのようなダイレクトな方向入力として使う以外、もっとも期待される使用方法が「認識」ではないでしょうか。

### 秘宝館「剣神ドラゴンクエスト甦りし伝説の剣」

赤外線センサーと全身を使ったゲーム製品は、任天堂Wiiが世界初というわけではありません。「剣神ドラゴンクエスト甦りし伝説の剣」(スクウェア・エニックス／2003年)で利用されていたのが国内メジャー作品では最初といえるかもしれません。

図9-11 「剣神ドラゴンクエスト」

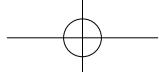


フォトダイオードと赤外線LED光源が受光部(ロトの証)に組み込まれ、「ロトの剣」には再帰性反射剤(反射板や銀スプレー)が処理されています。この「電池の要らない剣」を振り回して、8種類の切る方向に加え、正面に構えて魔法を使うモーションを入力することができました。

この製品は滋賀県草津市にあるインタラクティブ技術の研究開発企業「新世代株式会社」の技術によって実現しています。この会社のHPにいくと、技術の高さとさまざまな産業へのインタラクティブ技術のインパクトをうかがい知ることができます。

**新世代株式会社**

URL <http://www.xavix.jp/>



この節では、筆者が実際に開発した剣術アクションゲーム「JaWii's Virtual Fencing」(ジャウイのバーチャルフェンシング)の開発をベースに WiiRemoteを使ったモーション認識の基本テクニックと、剣術、特にフェンシングゲームへの応用を簡単に紹介します。

## フェンシングゲーム「JaWii's Virtual Fencing」

この作品「ジャウイのバーチャルフェンシング」は筆者がフランスの西部ラヴァル市(Laval)にて、テーマパーク開発のためのエンタテイメントシステムの開発に従事してたころに、市の観光振興企画として開発したものです。

ラヴァル市は人口10万人程度、世界遺産モンサンミッシェルから車で2時間程度の場所にある、中世の雰囲気を残す美しい中規模都市です。バーチャルリアリティ応用で有名な学術・産業研究都市でもあります。年に一度ヨーロッパでもっとも大規模なバーチャルリアリティのイベント「Laval Virtual」(ラヴァル・バーチャル)が開催されます。

図9-12 「ジャウイのバーチャルフェンシング」研究室でのユーザーテスト



Laval Mayenne (Wikipedia 英語)

URL [http://en.wikipedia.org/wiki/Laval,\\_Mayenne](http://en.wikipedia.org/wiki/Laval,_Mayenne)

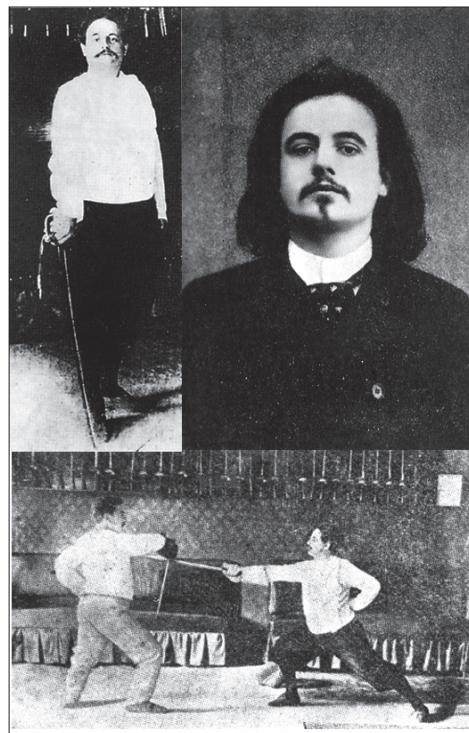
Laval Virtual (日本語ページあり)

URL <http://laval-virtual.org/>

フェンシングゲーム「JaWii's Virtual Fencing」はそのLaval Virtualでラヴァル市のブースで展

示されるゲーム企画でした。ちょうどラヴァル市は2007年に、同市出身の画家アンリ・ルソー(Henri Rousseau, 1844～1910年)と同じ時代を生きた、ラヴァル市出身の劇作家アルフレッド・ジャリイ(Alfred JARRY, 1873～1907年)の没後100年祭を祝っていました。

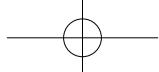
図9-13 アルフレッド・ジャリイと古式フェンシング



ジャリイは作家としては「ウビュ王」(Ubu Roi)シリーズが有名ですが、古式フェンシングの名手でもありました。そこで、当時発売されたばかりで話題だったWiiRemoteを使って「ジャリイに古式フェンシングの指南を受ける」というゲームアイディアが、市民にジャリイの人物を伝える良い企画として持ち上がったのです。

## 古式フェンシング指南ゲームの開発

まずは解説に入る前に、完成版の動画をYouTubeにアップロードしておりますので、ご覧ください。



ジャウイのバーチャルフェンシング (動画) : WiiMedia:Sword Fighting "JaWii's Virtual Fencing"

URL [http://www.youtube.com/watch?v=Kl\\_-KoVLtx4](http://www.youtube.com/watch?v=Kl_-KoVLtx4)

フェンシングの基本として、「突き」「正面切りつけ」「右切りつけ」「左切りつけ」といった攻撃、それから各攻撃に対応した防御法があります。古式のフェンシングはより複雑ですが、現代のように電気を使った接触検出はありませんし、防具も付けません。

このような複雑で形式ばった古式フェンシングのモーションを、子どもも交えた一般のお客さんに伝えるのは企画の趣旨ではありませんが、ジャリィが嗜んだという古式フェンシングはちゃんと表現したいところですので、フランスフェンシング協会に依頼して、Gypsy社製の機械式モーションキャプチャで古式演舞を収録しました。

3D Studio MaxとVirtoolsを使って、ジャリィを模した3Dキャラクター「JaWii」にこのモーションを元したアニメーション割り当てます。

図9-14 プロフェンシング選手 Benoit Pincemaille 氏によるモーションキャプチャ収録



ゲームは背景投影の大スクリーンに表示され、プレイヤーの視点で遊びます。プレイヤーの3Dモデルは必要ありませんが、フェンシングのサーベルを振るアニメーションだけは事前に複数通り作成しておきます。

図9-15 リアルタイムアニメーションはVirtoolsで開発



## モーション検出のための評価関数を作る

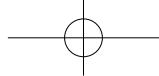
さて、ここからがWiiRemoteプログラミングの話になります。

プロジェクト背面投影という設営の構成上、赤外線マーカーは利用できそうにありません。加速度センサーだけでさまざまなプレイヤーの複数の動作を認識する必要があります。

「振る」を認識するだけであれば、加速度センサー各軸の強度を算出するマグニチュードで十分でしょう。しかし、大人や子どもなど、人によってマグニチュードの強さは異なりますし、「突き」だけならともかく、フェンシングらしく切りつける方向もある程度は検出したいと思います。ここで複数のプレイヤーの「切りつける」という動作について、加速度センサーの値をテキストファイルに保存し、作図して観察してみると、いろいろな問題や解決方法が見えてきます。以下、ポイントをいくつかまとめてみます。

### ●処理ウィンドウ

いつからいつまでの時間を「入力」とするのか。信号処理の用語で、処理の区間を「ウィンドウ」と呼びますが、これを決める方法を作らないと話が前に進みません。



### ●正規化

複数のプレイヤーに対する最大マグニチュードは異なります。正規化処理、つまり最大値を1.0にするような割り算を行うことで、ある程度特徴だけに注目できるようになります。

### ●WiiRemoteがねじれる

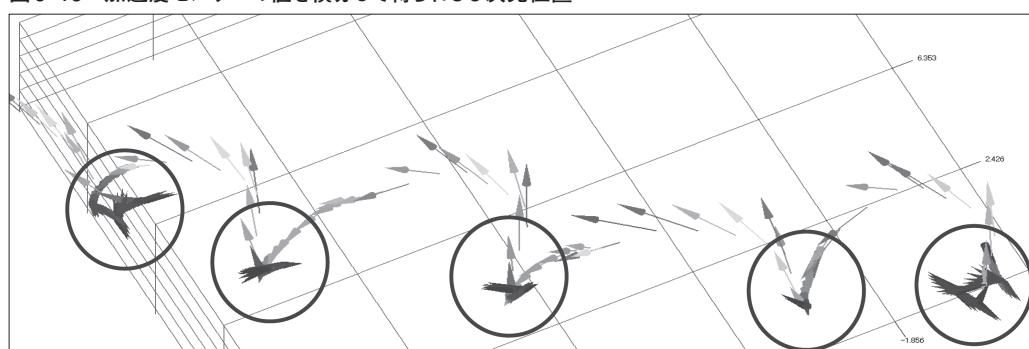
「まっすぐ振り下ろす」といっても、人間の手首は加速度センサーの軸とは関係なく、時間的にねじれることがあります。テニスなどでも同じことが起きます。

### ●検出してからでは遅い

たとえば「突き」のモーションを検出するのは簡単(最も加速度が高く、ベクトルが直線的)ですが、「突き」を表現するアニメーションにも再生時間(duration)があります。最大の加速度を得てから「突き」のアニメーションを開始したのでは、「なんだか遅いな」という操作感になります。

特にさまざまな問題の根底に感じられる原因は、WiiRemoteの検出分解能と回転速度が得られない点でしょう。図9-16は加速度センサーの加速度データ1つひとつを3次元ベクトルにして積分し、得た速度を基に、さらにその積分を取って3次元的な位置に配置した図です。「同じ場所で数回、自然に振る」というアクションで、長い矢印ほど大きいマグニチュードを表していますが、なぜか右から左に移動しているように見えます。

図9-16 加速度センサーの値を積分して得られる3次元位置



加速度センサーの分解能により、3次元座標が再構築できないことは第7章でWiiRemoteを「そっと動かすと検出されない」という実験を行ったので理解できるでしょう。そして、図中の○の部分に注目すると、よりおもしろいことが見えてきます。この部分は「振り」モーションの最後で、伸ばした腕を引いて、後ろに振りかぶった瞬間です。明らかに振りの最高速に比べて直線的

な動きではなく、小さい力で回転しているように見えます。

実際にはこの瞬間、WiiRemoteは頭の後ろで手首を使って回転しています。つまり回転のエネルギーを WiiRemoteが取得できていないため、このような図になるようです。しかし逆転の発想で、この瞬間のマグニチュードは、振りの最大速のマグニチュードとは異なる性質を持っているので、弱いマグニチュードが入力されたときに「処理ウィンドウの最初」として評価を開始することができます。

### 3DVIA Virtools+WiiRemote

「JaWii's Virtual Fencing」の開発で使ったVirtoolsは高価な産業向け製品であることもあり、日本ではそれほど有名ではありませんが、欧米では最も利用されている可視化プラットフォームです。モデルや画像などのリソースに対して部品化されたGUIプログラミングを施すだけではほとんどのインタラクティブデザイン関係が作れてしまう画期的なツールです。

カスタマイズ性も高く、レンダラーやプラグインなどほとんどのソースは公開されています。ゲームの流れやおもしろさの根幹に関わる部分の設計をプランナーがGUIで作成し、最終工程である最適化やプラットフォームの独自部分などをプログラミングで行う、といったゲーム開発手法です。

ソニーPSP用や任天堂Wii用のVirtoolsも存在します。任天堂Wiiの開発ライセンスを持っているゲーム開発企業であれば、WiiRemote関係のプラグイン入手することもできるそうです。またVirtoolsのフリーな開発者コミュニティは活発で、スワップミート(<http://www.theswapmeet-forum.com/>)でさまざまな情報やソース、プラグインが共有されており、PC上で利用できるオープンソースのWiiRemoteのプラグインも多数あります。将来ゲーム制作を目指す学生やフリーのゲーム企画者にとって、これは協力なソリューションです。Virtoolsを使ってPC上でゲームのプロトタイプを制作すれば、すばやくアイディアを形にできますし、資金や技術的なリスクを回避できるからです。

日本ではクレッセントと三徳商事という会社が中心に代理店を行っています。コンテンツの制作サポートや技術サポート、教育支援、学校向けライセンス販売などリセラー各社の得意分野がありますから、興味があったらまず問い合わせてみてください。お試しライセンスを発行してくれるかもしれません。

#### クレッセント

URL <http://www.crescentvideo.co.jp/virtools/>

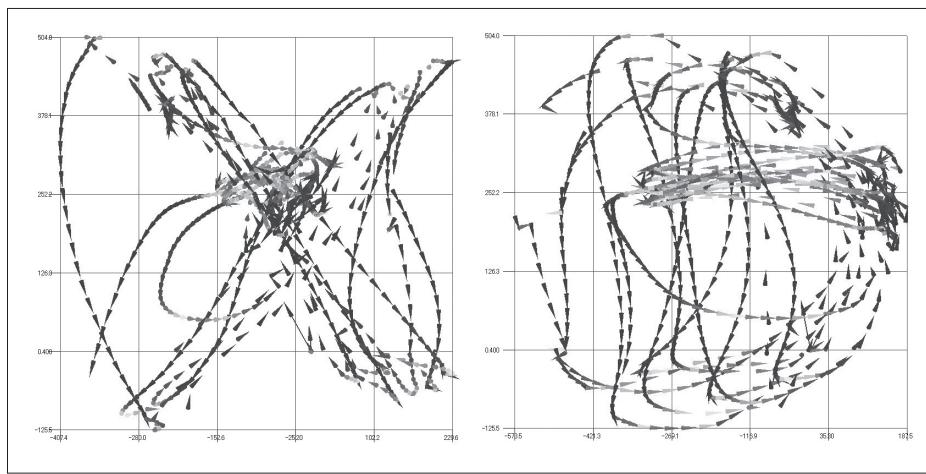
#### 三徳商事

URL <http://virtools.jp/>

以後、このような「評価関数」を作り込んでいくことで、目的のモーションを発見する関数を作っていきます。あとは個々のモーションに対応する評価関数をそれぞれ作っていきます。

しかし、加速度センサーだけの値だけでは観察できるデータが少なすぎます。そこで別の測定方法を使って振る舞いを観察します。図9-17は光学式のモーションキャプチャーを装着した状態で、WiiRemoteを持って「正面、右から、左から」の攻撃モーションを繰り返した様子です。

図9-17 モーションキャプチャーによる「振りの」様子(左:正面図、右:上面図)

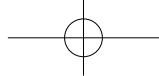


モーションキャプチャは高価な機材なので、そう気軽に使うことはできないでしょう。しかし、このような実験と可視化を一度行っておくことで目的のモーションを検出するために、どのようなベクトルに注目すべきか、またどれぐらいのサンプル時間(図の矢印の個数)が必要かを見極めることができます。理想とされる方向と近いものを1.0とすればよいのです(ベクトルの内積を使いましょう)。

また、評価関数化することで複雑なモーションも簡単に設計できるようになります。評価関数の足し算や掛け算を使って、複数の評価関数が同時に成立する条件を判断させたり、マイナスの評価を使って、誤検出されやすい条件から逆の条件を浮き立たせます。

例えば図9-17では、「突き」と評価されるベクトル(つまりY軸、十字ボタン方向への加速)に對して着色していますが、このベクトルから遠いものが「右切りつけ」や「左切りつけ」に該当します(「振り」モーション中は重力の影響はほとんどないことも読み取れます)。「突き」の最高速モーションは5サンプル程度で認識できているので、他の評価関数では、より多くのサンプルを使って「突きではないモーション」に注目させればよいのです。

この方法を使って「ジャウイのバーチャルフェンシング」では、さまざまな年齢層のプレイヤーでも3方向の攻撃モーションと防御モーションを認識させ、適切なアニメーション再生に割り当



て、ゲーム作品を完成させることができました。

**PRACTICE****振りの検出****【演習】☆☆**

7.5節の「WiiRemote測定器」を参考にして、テニスや「スイング」と「寸止め」「バックハンド」「(左右への)打ち分け」を分類する評価関数を作成せよ。

**【演習】☆☆☆**

上記の評価関数に対し、WiiMotionPlusと最新のWiiYourself!を用いて、回転を積極的に利用した評価関数を作成し「フェイントが検出できる剣道ゲーム」を作成せよ。

WiiMotionPlusは本書執筆の最終段階で発売されましたので、上記のようなグラフを描くことはできませんでした(発刊が遅れてしまっています!)。演習ではWiiYourself!によるC++を意識していますが、グラフを描画やファイル入出力もきっちり実装するなら、C#.NETによるフォームアプリケーションのほうが便利かもしれません(小坂研究室にCSVファイルを保存するサンプルがあります)。

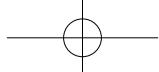
このような「モーション評価関数デザイン」は、今、ゲーム開発の世界ではとても需要がある技術です。HMMやSVMなどの機械学習を用いた方法も、研究としてはおもしろ味がありますが、最後はこの評価関数のデザインセンスが、ゲームのおもしろさを決定付けることもあります。if文のカタマリで開発し、ゲームプログラマの誰かに特化されたインタラクションではなく、エレガントでエキサイティングな評価関数を設計できるよう、探求してみてください。

**9.4**

## 作品編(1)： WiiBoardを用いた「オーラ診断」

次のテーマはWiiBoardを扱います。1.4節で紹介した学生作品「人間椅子」「でもWiiBoardを2つ使い、Windows上の独自APIにより開発を行っていました。

この節ではMac上のProcessingを使った学生の卒業制作作品「オーラ診断」の開発プロセス

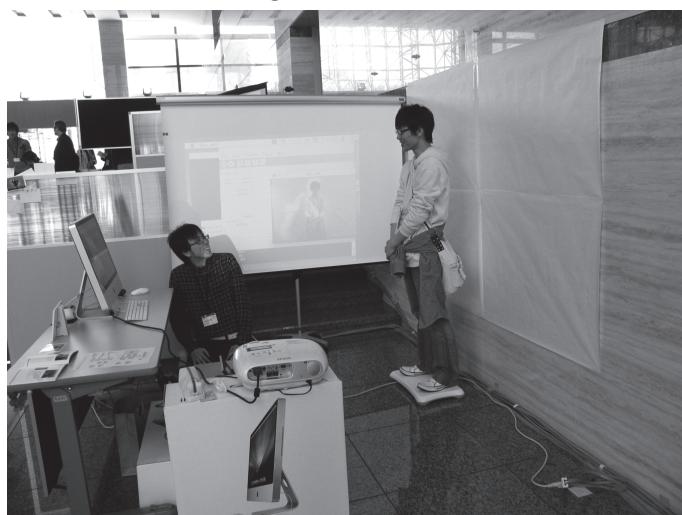


を、実際に本作品を開発した東京工科大学コムメディアデザイン研究室・電王隊（小笠原明日美、平塚宏、平野実花、渕上伸吾）のみなさんのご協力により学生視点で開発資料を紹介することで、WiiBoardによる作品作りに親しんでみたいと思います。

## プログラミング初心者4人が作った「オーラ診断」

「オーラ診断」は東京工科大学メディア学部の卒業制作展「メディアコンテンツ展2009」の「ライフエンタテインメント」として発表された作品です。

図9-16 MacとProcessingとWiiBoardで作った「オーラ診断」



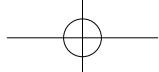
### オーラ診断（動画）

URL <http://www.youtube.com/watch?v=3pL3ObUwoA8>

### プログラミング初心者4人が作った「オーラ診断」という作品

URL <http://gryng.blog87.fc2.com/blog-entry-15.html>

まずは、「オーラ診断」の動画とブログから紹介します（以下、渕上伸吾氏のBlogより文体も含めできる限り、引用してみましょう）。



### ■「オーラ診断」の概要

「オーラ診断」は体験者のオーラを診断する作品です。自分のオーラの色や形を見て、脳内メーカーのように楽しんでもらうことが狙いです。

こんなかんじです。

怪しい「オーラ」診断中



### ■この作品のキモ

いわゆる脳内メーカー系サービスは、名前や誕生日を入力させることで結果を算出しています。そうしないと、分析する手がかりがないので当然です。

しかし、オーラを診断するにあたって、体験者に“入力”をさせたくありませんでした。厳密には、入力したことを気がつかせない。ここがこの作品のキモになっています。

そのために使ったデバイスが、バランス Wii ボード (WiiBoard) です。

### ■バランス Wii ボード

NINTENDO Wii 用の周辺機器であり、Wii Fit でおなじみのバランス Wii ボード。Wii Fit ではバランスゲームや筋トレ、ヨガなどを楽しめます。

このバランス Wii ボードは、乗った人の重心を求めるすることができます。具体的には、左右の足の前後、合計 4 力所にかかる重さを得ており、それぞれを比べることで重心を調べています。

このバランス Wii ボードなら、体験者に意識させずに情報を入力させることができる！と思ったわけです。なぜなら、そこに立たせるだけで重心の情報を得ることができちゃうんですから。

### ■BBOSC

バランス Wii ボードと Mac を接続するために「BBOSC」というソフトを利用させていただきました。

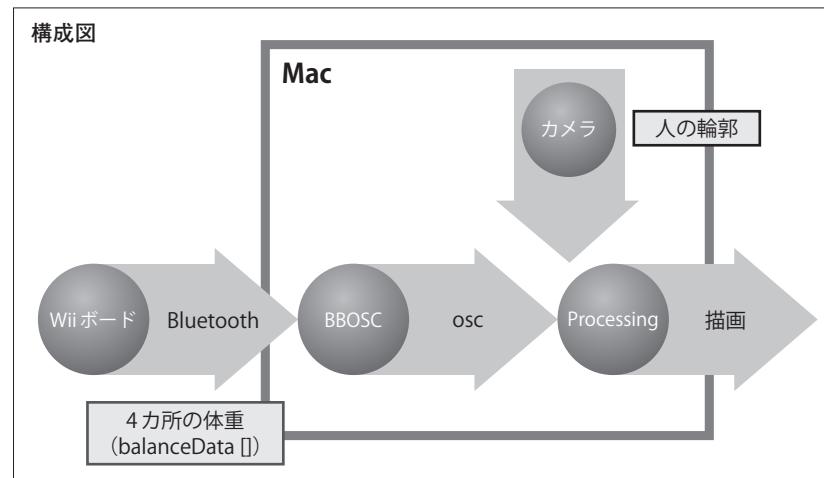
**4nchor5 la6 (「BBOSC」ダウンロード)**

**URL** <http://456.im/wp/download/>

立ち上げて、Wii ボードの電池ボックスの中にある Sync ボタンを押すだけ。びっくりするほど簡単に接続できました。

そうして Mac に送られてくる 4 力所の体重の情報を、Processing に渡すわけです。このあたりは、『Web Designing』(2008年3月号) の記事「Beyond the Browser」を参考にさせていただきました。

大雑把にいえば、BBOSC が体重の情報を OSC という規格で送信し続けてくれるので、Proecssing 側では「oscP5」というライブラリを使ってキャッチする、という感じです。



### ■ Processing

Processing はビジュアル表現が容易なオブジェクト指向のプログラミング言語、らしい。はっきり言って名前すら知りませんでした。

Processing では体重の情報を元にオーラを描画していきます。

人の重心は絶えず動いているもの。それだとちょっと扱いにくいので、その人の平均的な重心位置を求めるために、数秒の判定時間を設けました。その間、体験者には画面に集中しておいてもらいます。そして見つかった重心の位置を使って、その人の基準となる 1 色を設定します。

基準の 1 色だけだと画面が寂しいので、重心の移動に合わせて、ある程度色が変化するようにしました。この変化の幅も重心の位置から設定しています。より「その人だけのオーラ」が診断できるようになりました。

書いてないこともまだたくさんあるんですが、作品のおおまかな仕組みを紹介してみました。

## 補足：「BBOSC」とは？

以上の渕上伸吾氏のブログエントリーだけでは情報が足りないので、以下補足します。

「BBOSC」は石橋素（いしばしもとい）氏が雑誌『Web Designing』連載記事のために開発したものです。WiiBoardの情報をOSCで取得することができます。

WiiRemoteでMacを操作できる「DarwiinRemote」などを開発したHiroaki Kimura氏による、Mac OSにおけるWiiRemoteプログラミングAPI「WiiRemote Framework」を参考して開発されたそうです。

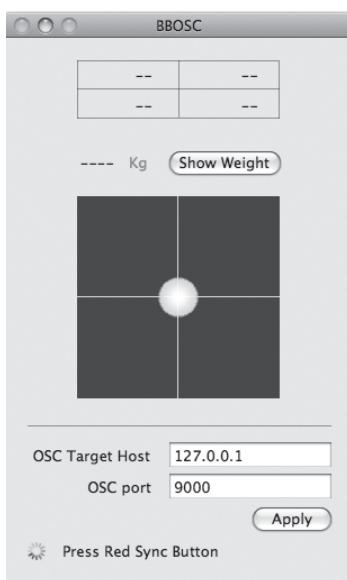
Hirolog（「DarwiinRemote」WiiRemoteでMacを操作できる）

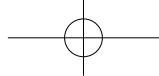
URL <http://blog.hiroaki.jp/2006/12/000433.html>

Hirolog（WiiRemote Framework）

URL <http://blog.hiroaki.jp/2007/05/000456.html>

図9-17 BBOSC





### OSCとは

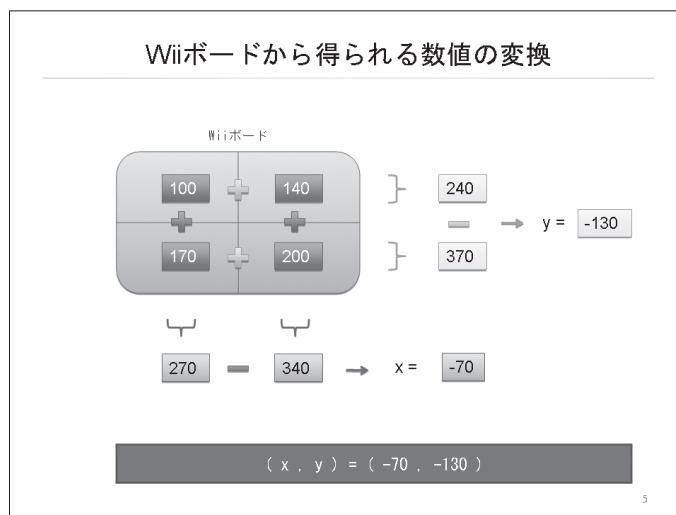
OSCとは「Open Sound Control」の略で、電子楽器やコンピュータの音楽演奏データをネットワーク経由でリアルタイムに共有するための通信プロトコル、つまりMIDIの代替となることを意図して作られたネットワークプロトコルです。カリフォルニア大学バークレー校にあるCNMAT(The Center for New Music and Audio Technologies)を中心にオープンソースで開発されています。

#### OSC

URL <http://opensoundcontrol.org/>

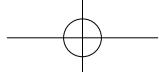
BBOSC実行時に、ターゲットとなるホストのIPアドレスとポートを指定します。4力所にかかる体重を0-10000にスケールしてOSCで送信します。右上、右下、左上、左下と、WiiBoardに内蔵された4つのひずみセンサーの値が得られますが、そのままでは使いにくいので、「オーラ診断」では「X, Y」の値に変換し、平均を利用しています。

図9-18 WiiBoardから得られるセンサー値の利用



5

「オーラ診断」を実現するために、BBOSCのほかに、カメラの利用と処理に「JMyron」、人と背景を分けるために背景差分法を用い、人の輪郭をとるために「blobDetection」というProcessingのライブラリを使っています。



### 画像処理のためのライブラリ

#### JMyron

URL <http://webcamxtra.sourceforge.net/download.shtml>

ちなみに「Myron」とはアメリカのコンピュータアーティストミロン・クルーガー (Myron Krueger, 1942年) に由来する名前と思われます。1980年代にインタラクティブアートやバーチャルリアリティを使った作品を作った人です。

#### blobDetection

URL <http://www.v3ga.net/processing/BlobDetection/>

3 H Processingの画像処理ライブラリです。「Blob」とは「もやもやしたカタマリ」のこと、人物などの検出をするには向いています。なお、このHPには画像処理を利用したさまざまなアートプロジェクトへのリンクがあります。

この作品が発表された東京工科大学「メディアコンテンツ展2009」のHP「ライフエンタテインメント」には「オーラ診断」の他にも、おもしろい作品が数多く発表されています。

このセクションにご協力いただいた渕上伸吾氏も「Earth Surfer」という WiiBoard を使った別の「バックストウザフューチャー感覚で写真を見るプロジェクト」を発表しています。

### ライフエンタテインメント

URL <http://www.teu.ac.jp/mce/2009/work/lifeenter.html>

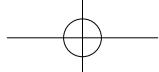
なお、Windows環境ではWiimoteLibでWiiBoardを利用することができます。特にC#.NETで WiiBoardを利用したい方は、小坂研修室でサンプルが公開されていますので活用するとよいでしょう。

### 小坂研修室「WiiBoard」C#.NETでの開発

URL <http://www.kosaka-lab.com/tips/2009/02/wiiwii-fit.html>

この節ではWiiBoardとMac OSでの学生プロジェクトを扱いました。本書ではMac OSでのWiimoteプログラミングをProcessingとActionScript以外は扱っていませんでしたが、Bluetooth接続の安定感もあり「Wiimote Framework」など、Windows XP環境よりも先に日本人開発者によってプログラミング環境が開拓されてきた時期もありました。

またこの節で扱ったようなOSCのような「ネットワーク経由の楽器として扱う」という方法は、VJやアーティスト系に親しまれているインタラクティブなサウンドプログラム環境「Max/MSP」



などでよく使われる方法です。実際にMax/MSPとWiiRemoteを使ったVJ活動などもよく聞きます。Windows環境だけにとらわれる必要はないのです。

「オーラ診断」を使ったような、カメラ画像処理、画像エフェクトを組み合わせ、今後さらに幅広い層でバーチャルリアリティアート、ビデオアート、インタラクションアート作品が生まれることを期待します。

## PRACTICE

### MacとWiiBoardの利用

#### 【演習】☆☆

Mac環境で動く「BBOSC」や「WiiFlash」などWiiRemote利用ツールを探し、Processingを使った「誰も見たことがない」メディアアート作品制作に挑戦せよ。

#### 【演習】☆☆☆

WiiBoardを使って、人間の「足踏み」を検出してVR世界を散歩せよ。

「足踏み動作解析」については、東京農工大学教授藤田欣也氏他、論文が多数ありますので検索して参考するとよいでしょう。

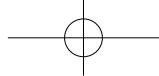
## 9.5

### 作品編(2)： Second Lifeで使う

世界的に有名なバーチャルワールドサービス「Second Life」でWiiRemoteを使えるようにしてみましょう。

Second Lifeは無料で利用できるバーチャルリアリティ空間共有サービスです。リンデンラボという会社が運営しており、リンデンドルという実社会に似た通貨を買ったり、土地の売買や建築、キャラクターの装飾やプログラミングといったユーザーによるコンテンツ作成が行えるのが特徴です。よくわからない人は「ゲームが目的ではない3Dネットゲームのようなもの」を想像するとよいでしょう。

Second Lifeのクライアントソフトのソースコードが公開されているわけではありません



が、第3章で学んだGlovePIEを使えば、プログラミングや改造することなくSecond LifeをWiiRemoteで操作することができます。

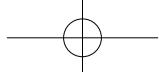
Second Lifeを使ったバーチャルリアリティ空間の建築作品作りで有名な首都大学東京首都大学東京の渡邊英徳氏が、インタラクティブ技術のイベントのための写真アーカイブ作品「Laval VRchive」の展示用スクリプトを作成しています。渡邊氏より、以下のコメントをいただきましたので、紹介します。

図9-19 Second Life作品「Laval VRchive 2009」



まず、Google Earth用のPIEスクリプトをベースに、複数回のユーザビリティ検討を行った結果、まずSecond Lifeにもとからある前進/後進機能をオフにすることにしました。「バーチャルリアリティ空間内で等身大のサイズで過去の体験型イベントの写真を共有する」というコンテンツの設計上、前後に移動することがそれほど重要ではないと判断したのです。このような機能の刈り込みは、ユーザーインターフェースデザインを向上させる上で重要な機能制限といえますし、GlovePIEで入力させなければよいので、比較的簡単に検討することができました。

しかし再検討を重ねていく上で、最終的には「十字キーで前後移動+左右転回、「Bボタンを押しながら↑↓」もしくは「+ボタンで上昇下降」という仕様に落ち着きました。赤外線センサーの値はマウスポインタに割り当ててあります。USB給電できるセンサーバーをプロジェクタースクリーンの下に設置し、Second Life内の「指さし」と同じ感覚で、作品中のオブジェクトにWiiRemoteを向けて、Aボタンを押すことで操作することができます。



Second Lifeではちょっとしたことでカメラアングルがずれてしまうので、Homeボタンでリセットできるようになっています。またSecond Lifeのコンテンツを展示する場合、メニューバーが邪魔になるため、ディベロッパー mode 「Ctrl + Shift + D」に切り替え、「インターフェイスを off」、「Ctrl+Shift+1」というモードにしています。この場合、画面上部のメニューバーは不可視にはなっていますが、メニューバーそのものは存在しているため、画面の上下端でAボタンをクリックすると誤動作する恐れがあります。今回のスクリプトでは実装していませんが、画面の上下端にマウス移動のリミッターを付けることが望ましいかもしれません。

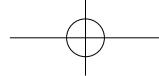
図 9-20 WiiRemote を使って片手で操作できる



#### **WiiRemote から自動切断されないようにしたい**

WiiRemote の更新が長時間なにもないと、いつの間にか切断されてしまいます。赤外線を見るなどして、切断されないリポートモードを使うと確かに切断はされないのですが、今度は電池が切れてしまします。実験するには長い時間がかかりますが、時々 LED 出力などの信号を送ってあげるとよいのかもしれません。なお Wii 本体では、同様に WiiRemote が長時間操作しないとスリープモードに入るのですが、何か WiiRemote のボタンを押すと、本体側から再度 Bluetooth のペアリングを要求するらしく、接続が復旧するようになっています。

Windows 環境においてはまだ Bluetooth 自動接続に成功したソフトウェアはありませんが、試している人がいないわけではありません。原理的には DDK があるので不可能ではないはずです(専用の HID ドライバを作った方が早いのかもしれません...)。そのうちこういった高度な WiiRemote 管理もオープンソースの API で可能になるかもしれませんね。

コード9-6 **GlovePIE** Second Life「Laval VRchive 2009」展示用 PIE スクリプト(抜粋)

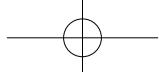
```
Mouse.LeftButton = Wiimote.A
Keyboard.ESC = Wiimote.Home
Keyboard.Up = Wiimote.Up
Keyboard.Down = Wiimote.Down
Keyboard.Left = Wiimote.Left
Keyboard.Right = Wiimote.Right
Keyboard.E = Wiimote.Plus
Keyboard.C = Wiimote.Minus
if Wiimote.B & Wiimote.Up Then
    KeyBoard.Up = False
    Keyboard.E = True
    Wait 600ms
    Keyboard.E = False
endif
if Wiimote.B & Wiimote.Down Then
    KeyBoard.Down = False
    Keyboard.C = True
    Wait 600ms
    Keyboard.C = False
endif
<以下、赤外線センサーの利用や安定感向上のためのスクリプト>
```

**PRACTICE****Second Life + GlovePIE を極めよう****【演習】☆☆☆**

上記PIEスクリプトと第3章を参考にして、GlovePIEとSecond Lifeを使って、自由にバー  
チャルリアリティ空間を散歩できるPIEスクリプトを作成せよ。その際ヌンチャクなども利用  
して、展示向け、デスクトップ向けなどのカスタムバージョンも検討せよ。

**9.6****モノ編(1)：  
センサーバーを自作する**

WiiRemoteの赤外線センサーは非常に多機能で高速で高機能ですが、このままの状態では、セ



ンサーバーをWii本体に接続していなければ使えません。せっかくPCでWiiRemoteが使えるの  
で、Wii本体がなくてもよいように、センサーバーの仕組みを知り、自作に挑戦してみましょう。

WiiRemoteの加速度センサーだけ使う予定の読者の方や、センサーバーをWii本体に接続して  
利用する方は、この節は読み飛ばしていただいてもかまいません。

## センサーバーの仕組み

センサーバーは、名前だけ聞くと「中にセンサーが入っている」ように聞こえますが、実際には赤外線センサーはWiiRemote内に実装されており、センサーバー内部にセンサーは存在しません。

センサーバー内部には、左右にそれぞれ5つの赤外線LEDが実装されています。Wii本体と接続しているケーブルはただの電源ケーブルで、赤外線LEDはプラグを差している間、常に点灯しているようです。つまり、LEDは信号を送って同期したり、変調（周波数を変えて明度や速度を調整すること）したりといった凝ったことはせず、単純に直流電流を使って、同じ明るさで点灯しています。ちなみに「テレビの友チャンネルGガイドfor Wii」でリモコンとして使うときだけは、リモコン信号の規格に合わせて高速に点滅しています。

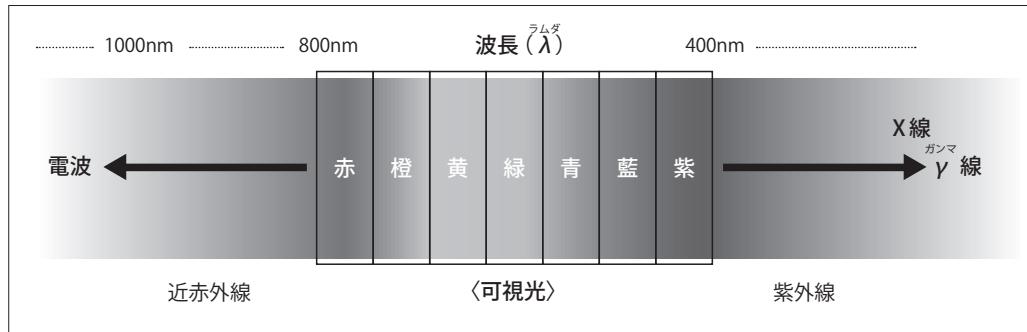
同期や変調といった複雑な電子回路の場合は、自分で作るのは少々大変ですが、LED点灯回路ぐらいであればそれほど難しくはありません（中学生レベルの電子回路です）。このLED点灯回路を赤外線LEDを用いて自作すれば、オリジナルの赤外線マーカーのできあがりです。センサーバーは必要なくなります。PCでWiiRemoteを利用するのに、いちいちWii本体を起動してセンサーバーを点灯させる必要はありませんし、赤外線センサーを使った自作の作品を利用する上で自由度も広がるでしょう。

## 目には見えない「近赤外線」

ここで赤外線について学んでおきましょう。まず、世の中の光にはすべて「波長」があります。波長が変わると色が変わって見えます。虹やプリズムを通して太陽の光を分解してみると「赤橙黄緑青藍紫」という順番に並んで見えます。

赤色に近くなればなるほど長い波長、紫色に近くなればなるほど短い波長です。人間が肉眼で  
見ることができる波長「可視光」には限りがあり、実際にはもっと多くの波長が存在します。

図9-21 目には見えない近赤外線



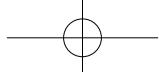
## 近赤外線とは

「赤外線」と一言でいっても、本書で扱う赤外線は波長700nm～2500nm近辺の近赤外線と呼ばれる赤外線です。他にも2500nm～4000nmの中赤外線や、波長 $4\ \mu\text{m}$ ～ $1000\ \mu\text{m}$ の遠赤外線(熱線)があります。いずれも人の目では見えない光で、電波よりも波長の短い電磁波のことです。遠赤外線以上に波長が長くなると、マイクロ波、メートル波といった電波と呼ばれます。

人間が見える可視光は、せいぜい赤の750nmから紫の380nm程度で、それよりも短い波長になると紫外線となり、さらに波長が短くなるとX線やガンマ線と呼ばれ、性質が異なってきます。

人体に吸収されたり、山や建物を通り抜けたり、お湯が沸いたり、無線通信できたり……と波長ごとにいろいろな利用上の特性がありますが、特にWiiRemoteを使う上では波長1000～800nmの近赤外線の光を使います。この近赤外光は、人間の目に見えにくいという以外は、普段我々が目にする光とほとんど変わらない性質を持っています。

近赤外線は目に見ることができませんが、可視光に近いため、目に見える光に似た拡散や反射が観察できます。目に見えないという理由から、自動ドアの接触センサー(フォトインラプタ)やテレビのリモコン、携帯電話同士の赤外線通信(IrDA)などに使われています。こうしてみると、街の中は赤外線センサーだらけなのです！なお、インフルエンザで発熱している人を見分けるときなどにも使われる「熱画像カメラ」や「赤外線サーモグラフィー」とよばれる温度に応じて色をわりあてるカメラがありますが、これは黒体放射による7.5～13 $\ \mu\text{m}$ の波長、つまり遠赤外線です。



### 光の不思議についてもっと知りたかったら

文部科学省が2008年の科学技術週間で配布した「一家に1枚光マップ」が非常に良くできます。波長ごと、ありとあらゆる光について、実際に使われている例が写真入りで紹介されているポスターです。PDF版が理化学研究所のホームページからダウンロードできます。

#### 一家に1枚光マップ

URL [http://www.riken.go.jp/r-world/topics/080404\\_2/lightmap.pdf](http://www.riken.go.jp/r-world/topics/080404_2/lightmap.pdf)

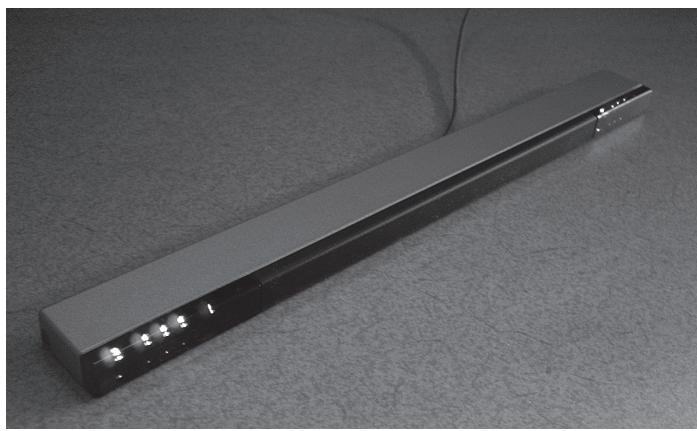
製作著作：文部科学省／監修：河田聰（独立行政法人理化学研究所）

## 見えない赤外線を見るようにするには？

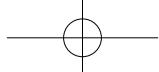
白熱電球なども、目に見える光（可視光線）とともに、熱線と近赤外線を発光しています。「センシング用途」つまり WiiRemoteのようなセンサーとして光を使う場合には、可視光や熱は必要ありません。逆に効率が良くないので、マーカー用光源として赤外線LEDの発光を使う場合が多いようです。

センシング用途では、赤外線LED光源に合わせて、フォトダイオード（PD）やフォトトランジスタといった特定の波長の光に対して反応する半導体とセットで利用されます。TVのリモコンや携帯電話やPCの近距離通信に使うIrDA（Infrared Data Association）規格、自動ドアもこの赤外線LEDと半導体素子のセットで構成されています。

図9-22 センサーバーの赤外線をデジカメで撮影した様子



目には見えない赤外線ですが、デジカメや携帯、Webカメラなどの画像センサーを使うことで、赤外線を画像として見ることができます。デジカメに利用されている画像センサーである「CCD」や「CMOS」は、本来、限定された幅の波長の光しか電子に変換できないのですが、「赤・



緑・青」といった、人間が画像として利用するための受光特性以外にほんの少しだけ、可視光の外側の波長に感度があるデバイスもあります。この受光特性を利用して「見えない赤外線を見る」ことができます。

この知識は非常に有効で、センサーバーを自作したときや動作確認をする上で、赤外線が見えるカメラを手元においておくと、赤外線LEDの点灯状態が見えて非常に便利です。

なお「ノクトビジョン」や「ナイトスコープ」と呼ばれるカメラは、この仕組みを使って目に見えない赤外線という明かりを使って、夜中や暗闇でも撮影できるカメラを実現しています。

#### **WiiRemoteの受光特性は？**

WiiRemoteの赤外線センサーはいったいどのような仕組みでどんな波長の光を感じることができのでしょうか？

実際には型番が公開されているわけではないのでよくわかりません。実験してみるしかないのですが、WiiRemoteに内蔵されているセンサーは、任天堂が台湾のPixArt Imaging (<http://www.pixart.com.tw/>) に特注して開発した、特別な赤外線画像センサーといわれています。推測の範囲を出ませんが、低解像度のCMOSで、デジカメのような「画素値」ではなく赤外線の明かりの「重心の位置」を高速に出力するタイプのデバイスのようです。

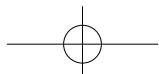
一般的なWebカメラなどの画像処理速度が毎秒30～60フレーム程度なのに対して、このPSDは2次元の重心位置を出力するだけなので、高速です。値段と大きさにもよりますが、毎秒400フレームぐらい出せるデバイスもあります。

なお筆者が大学4年生のときに書いた論文「光学的3次元位置検出法を用いたリアルタイム人間動作入力デバイス」では、浜松ホトニクス製のPSD(Position Sensing Device)カメラを使いました。このカメラもWiiRemoteのCMOSに似た半導体デバイスですが、当時100万円以上で研究室が購入したこと記憶しています！

## 電子部品をそろえる・工作する

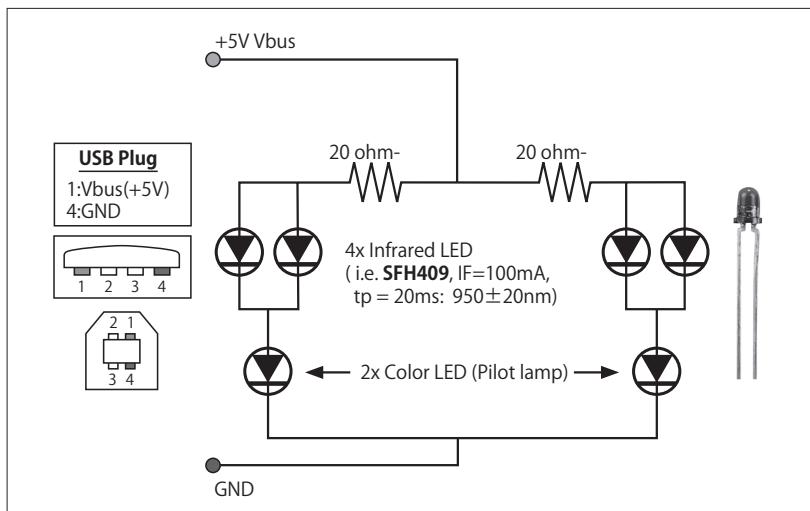
### **ATTENTION!**

これから先は電子回路等の知識がある方、半田ごての扱いなどが可能な方のみ実践に臨んでください。本書を原因とするPCの破損や火傷その他の不利益について、著者や出版社は責任を持ちません。



こちらが自作USBセンサーバーの回路図の例です。

図9-23 自作USBセンサーバーの回路図例



材料としては、赤外線LEDと抵抗、基盤、半田ごと一式、あとは不要なUSBのケーブルを1本、切断して使います。

平型のUSBプラグなら、金属端子面を下にして左から順に1、2、3、4と4つの端子がついています。この1番が電源となるVCC(+5V)で、4番がGND(−)です。台形のUSBの端子の場合は台形の長編を下側にして右上が1番、右下が4番になります。

USBのケーブルを適当な長さで切断して、テスターなどで確認しながら、図中のUSB1番を回路の+5Vに、USB4番を回路のGNDにそれぞれ半田を使ってつなぎます。

赤外線LEDは普通のLEDと同じく、秋葉原や通販で入手できる電子部品のお店で買うことができます。

「秋月電子通商」の赤外線LED売り場の例

型番：OSIR5113A

VF=1.25V(@20mA)、ピーク波長940nm、半減角15度、推奨電流20mA

秋月電子通商

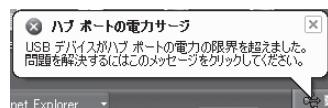
URL <http://akizukidensi.com/catalog/g/gl-00656/>

値段は100個入りで700円と、電子部品としては気軽に買える部類に入ります。購入前に、

仕様書をよく見てください。重要なのは「ピーク波長」、「VF (DC Forward Current)」、「推奨電流」、それに「半減角 (50% Power Angle)」と呼ばれる値です。ピーク波長は保障はできませんが、WiiRemoteには900～1000nmの間ぐらいがよいようです。半減角は、正面を100%としたときに明るさが半分になる角度です(LEDには広角のものと正面に指向性の高いものがあります)。なお、ここで紹介している「OSIR5113A」は小坂研究室でも利用実績があるそうです。

またVF (mA)によって制限抵抗の値が決まりますので、それに合わせた抵抗も一緒に購入してください。制限抵抗を入れないと無制限に電流が流れてしまい、非常に危険な光源になってしまいます。最悪、PC本体を壊すかもしれません。

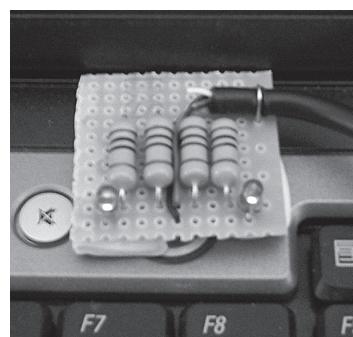
図9-24 USB電力のサージ警告



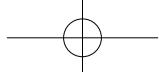
この警告メッセージはUSBポートの電流がUSBの規格で定められた許容量である500mAを超えたときなどに表示されます。配線が甘くてVbusとGNDがショートしているときなども同様に表示されるので、このメッセージが表示されたときは、すばやくUSBポートからプラグを抜き、テスターなどでショートがないか確認してください。

また、LEDはダイオードという電流を一方向にしか流さない性質を持ちます。アノード(足の長いほう)からカソードへ流れますが、逆には流れません。赤外線が見えるデジカメを傍らにおいて、仮組みしたりテストで駆動しながらやらないと、足を切った後では極性がわからなくなりますので注意しましょう。

図9-25 自作の極小USBセンサーバーの制作例



このように非常に小さなUSBセンサーバーも作ることができます。いろいろと応用の幅が出てきます。

**PRACTICE****赤外線センサーバーの自作****【演習】☆☆**

上記の回路図を応用して、LEDに対して適切な制限抵抗値を算出し、USBで給電する赤外線センサーバーを2セット自作せよ。

**【演習】☆☆**

上記2セットのセンサーバー（各LED2点）、合計4点の赤外線マーカーを取得できるプログラムをWiiYourself!かWiimoteLibを使って作成せよ。

**【演習】☆☆☆**

第1章「SoundQuest」にあるような、二等辺三角形の頂点に赤外線を配置し「三角形の向き」を取得できるプログラムを作成せよ。

ヒントは筆者のホームページ「Aki4IRDemo」、解答例としてYoutube動画「Sound Quest V1」があるので、参考にしてください。

**Aki4IRDemo**

URL <http://akihiko.shirai.as/modules/mydownloads/viewcat.php?cid=6>

**Sound Quest V1**

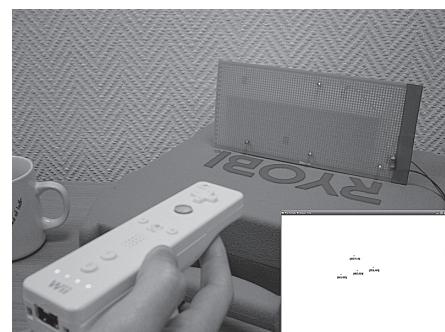
URL <http://www.youtube.com/watch?v=TMK7ULUG7S4>

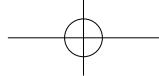
半田ごてが自信を持って握れる方のみお勧めします。半田ごてで火傷したりしても、本書は責任を持ちません。

赤外線は目に見えないので、回路図の赤外線LEDに加えて、通電しているかの確認のために可視波長（赤や緑）のLEDを使ってパイロットランプを作るとよいでしょう。

他の課題は自作センサーバーを作らなくても挑戦できます。三角形の認識でちょっと数学パズルがありますが、楽しんで解いてみてください。WiiRemoteは4点まで検出できるので、三角形の向きが拾えたり、面が推定できたりと、いろいろな応用できます。

図9-26 赤外線LED4点で二等辺三角形を検出させる例





## 9.7

## モノ編(2)： ロボット兵器「WiiRemoteTank」

WiiRemoteを使ってラジコンカーを操作しようというアイディアを実現した人は（世界には）意外にいるようです。

### WiiRemoteを使ってラジコンカーを操作

URL [http://gigazine.net/index.php?/news/comments/20061222\\_wii\\_rc/](http://gigazine.net/index.php?/news/comments/20061222_wii_rc/)

URL <http://www.inside-games.jp/news/329/32904.html>

この2006年12月のニュース(Gigazine)で紹介されている例は、WiiRemoteからの信号をBluetooth経由でPCに受信して、それをラジコンカーのコントローラー(プロポ)などに送信する方法です。

WiiRemote → PC → ラジコンプロポ → ラジコン

もう1つの動画も同様で、WiiRemoteに飽きたらず、ヌンチャクやWiiBoard、さらにiPhoneを使ってラジコンカーを操作する動画を公開しています。

## ロボット兵器「WiiRemoteTank」

人がやったことをただ真似てもおもしろくありません。ここでは、上のような構成ではなく、

Wiiremote → PC → WiiRemote → ラジコン

というプロポすら使わない方法で、ラジコンを操作してみたいと思います。

正確には「WiiRemoteでラジコンを操作」ではなく、WiiRemoteをラジコンと合体、つまり、WiiRemoteをラジコン化した「WiiRemoteTank」を開発します。

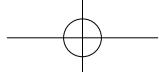
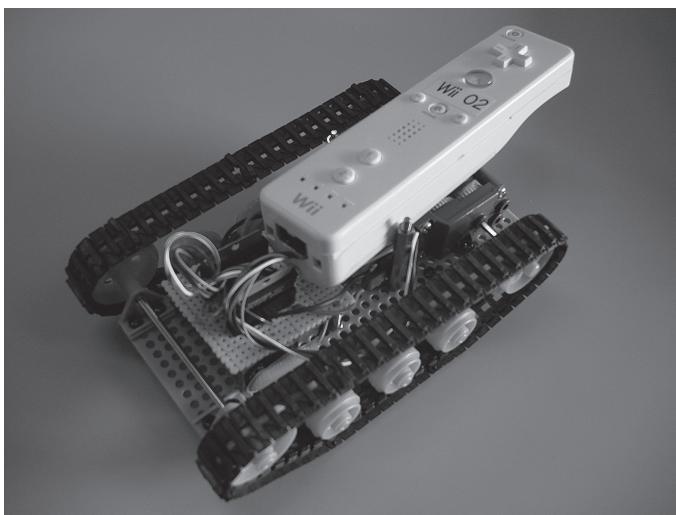


図9-27 ロボット兵器「WiiRemoteTank」



ほとんどロボット兵器です。武器はありません。

原理は簡単です。WiiRemoteあるプレイヤーインジゲーター（4つの青色LED）は、WiimoteLibのSetLEDs関数で信号を送るだけで、ON/OFFの出力ができます。このLEDの電力をモータードライバーに接続することでモーターを制御することができます。

モータードライバーとは、その名の通りモーターを制御する電子部品です。2つの信号の組み合わせによって、モーターの回転、反転、停止を行うことができます。

たとえば、[01]で前進、[10]で反転、[00]でストップといった2bitのデジタル信号で制御できるので、LEDの点灯制御を出力させてやるだけで、モーターの動作をコントロールできるわけです。

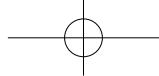
## 「WiiRemoteTank」の開発

ここではロボットの開発を演習している、大阪大学応用理工学科機械系3年生の演習「機械創成工学演習」の教科書を参考にしています。

大阪大学准教授細田耕氏による「機械創成工学演習III」テキスト

URL <http://www.robot.ams.eng.osaka-u.ac.jp/hosoda/enshu/start.html>

ここで紹介されているモータードライバー「TA7291P」を使います。WiiRemoteのLEDは4つ



るので、このドライバーを使って2個のモーターを制御することができます。モーター2個で操作できる戦車といえば、「タミヤタンク工作基本セット」でしょう。オンラインで1,500円で購入できます。

#### Toshibaのモータードライバー「TA7291P」データーシート

URL [http://www.robot.ams.eng.osaka-u.ac.jp/hosoda/enshu/doc/TA7291F\\_TA7291SG\\_ja\\_datasheet\\_070613.pdf](http://www.robot.ams.eng.osaka-u.ac.jp/hosoda/enshu/doc/TA7291F_TA7291SG_ja_datasheet_070613.pdf)

#### タミヤタンク工作基本セット

URL [http://tamiyashop.jp/shop/product\\_info.php?cPath=17\\_149&products\\_id=70108](http://tamiyashop.jp/shop/product_info.php?cPath=17_149&products_id=70108)

続いてコントローラー用のWiiRemoteと制御用のWiiRemoteの2台を用意します。まず制御用のWiiRemoteを分解し、LEDの信号を取り出します。

#### WiiRemoteを「分解」...?

「WiiRemoteを分解」とあっさり書いていますが、WiiRemoteのネジは特殊なドライバーでなければ回すことすらできませんが、そのようなドライバーなどなくとも開けようと思えば開けることはできます（もちろん、全く保証外の行為です）。

特殊ネジをはずしたら、普通のネジを入れておきましょう。

URL <http://ameblo.jp/akihiko/entry-10056910390.html>

図9-28 精密ドライバーとペンチを使う



さて、この先一番難しいのは、「WiiRemoteを分解しLEDに配線し、元通りに収めること」かと思います。かなりの集中力が要求されます。小坂研究室のTipsにWiiRemoteのLEDを換装する記事があるので参考にしてください。

#### 小坂研究室Tips「Wiiリモコンの青色LEDを赤色LEDに変更」

URL <http://www.kosaka-lab.com/tips/2009/05/wiiledled.php>

テスターを使って確認しながら進めてください。LEDのための信号を拾って、モータードライバーに接続します。

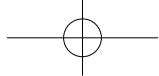


図9-29 LED信号をモータードライバーに接続



コントローラー用のWiiRemoteの傾きをPCが読み取り、その傾きに合わせて、制御用WiiRemoteに信号を送るプログラムを別途作成しておきます。モーターが2つありますので、加速度センサーの傾きに合わせて2つのLEDに対して[00]～[11]を出力するようなプログラムで十分でしょう。

完成版の動画は小坂研究室にて見ることができます。

#### 小坂研究室 Tips 「Wiiでラジコン」

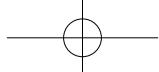
URL <http://www.kosaka-lab.com/tips/2009/05/wii-2.php>

コントローラー用WiiRemoteを傾けると、WiiRemoteTankが進みます。今回はWiiRemoteをプロポにして操作していますがWiiBoardなどで操作してもおもしろそうです。

LEDの出力はまだ2チャンネル分残っていますから、他にも武器やデコレーションを装備したり、WiiRemoteの赤外線カメラを利用して、赤外線を自動に追尾して動くロボットや、ライントレーサー（黒い線に従って動くロボット）としても展開することができるでしょう。

今回はWiiRemoteをロボットへ内装する例として、LEDから信号をとる方法を紹介しました。LED以外にスピーカーのアナログ出力や、拡張端子のI2Cインターフェースを使う方法も可能性がありそうです。

ここでの例ではロボット戦車ですが、かわいらしいモンスターのぬいぐるみを着せたり、ゲームと連動させたりすると、ビッグなビジネスチャンスがありそうです（笑）。

**WiiRemoteによるロボット開発****PRACTICE****【演習】☆☆☆**

WiiRemoteTankを応用して、周囲の赤外線を探して近づいてくる「ロボットペット」を開発せよ。

**9.8**

## サービス編： 体が不自由な方のための インターフェース

この節では、第8章で学んだ技術を応用して、WiiRemoteを利用することで、ハンディキャップのある方に自由にコンピューターを触れるように、何ができるかを中心に考えてみたいと思います。

**体が不自由な方のためのインターフェース****PRACTICE****【演習】☆☆**

WiiBoardを使って、手が使えなくてもマウス操作できるソフトウェアを作成せよ。

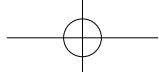
**【演習】☆☆☆**

WiiRemoteだけで文字入力ができる「WiiRemoteキーボード」を開発せよ。

**【演習】☆☆☆☆**

WiiRemoteに「ぬいぐるみ」の皮をかぶせて、幼児に持たせる「安心ぬいぐるみロボット」を開発せよ。

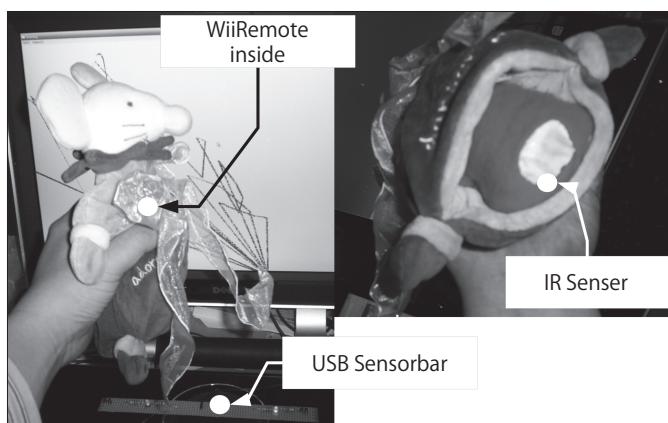
「体が不自由」と一言でいっても、先天的に不自由な方だけでなく、事故や病気で不自由になつた人や、一時的に不自由な人など、それぞれです。そんな方々にWiiRemoteだけでブラウザを操作したりメールを書いたりすることができるインターフェースを開発できれば、半身不随の方などの人生を大きく変えるかもしれません。



ちなみに「ハンディなんて自分には関係ない」と思っていると損をします。かくいう筆者も、生まれたばかりの自分の赤ん坊に2時間おきに授乳しなければならないときがありました。夜は論文を書いたりしていることが多いので、妻に代わって私が担当なのですが、赤ん坊にミルクを飲ませているときは両手がふさがっているので何もできません(とても眠くなる)。この時間を使ってメールに返信したり、ブラウザを触ったりといった簡単な作業ができれば、子育て初期の授乳ストレスはどんなに有意義な時間になったでしょうか! つまり両手があいている元気なうちに「こういうソフトウェアを作つておけばよかった!!」と何度も後悔した、ということです(笑)。

「幼児に持たせるロボット」は加速度センサーの値をうまく使って、眠ったり歩いたり、スピーカーを使ったり…とアイディアが広がります。フランス語では安心塗りぐるみのこと、「Poupee」といいますが、筆者は過去にこのアイディアで幼児向けのラクガキシステム「Papier Poupee Painter」を開発したことがあります。振るだけで塗り絵っぽいことができる作品でした。

図9-30 幼児向けラクガキシステム「Papier Poupee Painter」を使った「安心ぬいぐるみ」



幼児もインタラクティブ技術の視点では「ハンディを持つユーザー」とあまり変わりありません。いわゆるペイントブラシのような色を選ぶような機能は限定して、ラクガキのおもしろさだけを際立たせる、という仕掛けに WiiRemote の無線機能と安定性能は大変役に立ちました。

**WiiMedia:Painting "Papier Poupee Painter" ver.Alpha (動画)**

URL [http://www.youtube.com/watch?v=S8kYQbfN\\_9I](http://www.youtube.com/watch?v=S8kYQbfN_9I)

演習問題とはいえ、せっかく作ったソフトウェアですから、いいものができたら公開しましょう。なんといっても WiiRemote とソフトウェアだけですから、一般的な医療福祉機器に比べて非常に気軽に利用できます。

## 9.9

## 研究編(1)： 赤外線を極める

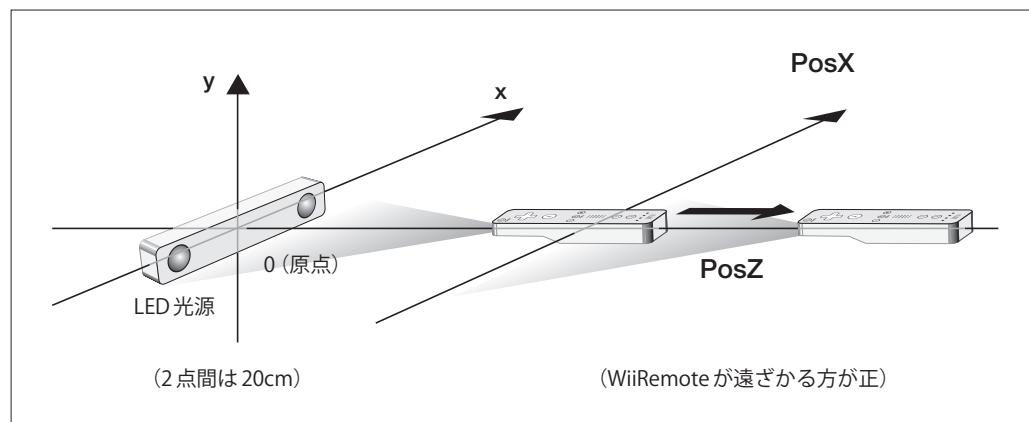
WiiRemoteの赤外線センサーは非常に高速で、使い道がたくさんあります。ここでは、2点の赤外線LEDの情報だけで、どこまで正確な「奥行きを含めた3次元座標」が取得できるか理論的に突き詰めてみます。

### 赤外線奥行き測定の基本理論

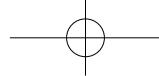
ここでは、幾何的な方法を使って2点のLED座標からWiiRemoteの3次元奥行き付きの座標を取得する方法を考えます。考え方のトレーニングだと思って読んでみてください。

いま、 $P(x,y,z)$ という位置にあるWiiRemoteが、原点 $O(0,0,0)$ という場所にあるセンサーに向かって赤外線センサーを向けたとき、WiiRemoteで取得できる2つのLED群の位置を $[IrX1, IrY1]$ 、 $[IrX2, IrY2]$ とします。

図9-31 ここでの座標系、LEDとWiiRemoteの位置関係（側面図）



この2つのLEDのX座標、 $IrX1$ 、 $IrX2$ について、その差の絶対値 $IrZ$ について仮説を立ててみます。



$$IrZ = | IrX1 - IrX2 | \dots \text{(式 9-1)}$$

いま、このWiiRemoteを原点から遠ざかる方向に移動させた場合、このIrZの値は遠近法に従って、遠くにいけば遠くにいくほど2つのLEDの差は小さくなります。WiiRemoteとセンサーの距離(以後PosZと標記、単位はmm)は比例関係にあるかもしれません。

式で表せば、

$$PosZ = K * IrZ \dots \text{(式 9-2)}$$

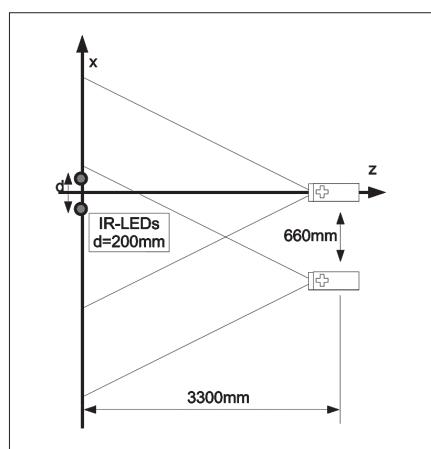
という関係が作れる可能性があります。もし、このKを簡単に求めることができるなら、WiiRemoteの赤外線LEDの値(IrX1、IrX2)から、奥行きPosZが算出できそうです。

なお、この式9-2でのKは比例関係を表しているだけで、定数かどうか、つまり1次関数なのかどうかは、いまのところわかりません。より複雑な2次関数以上かもしれません。実際に測定してみるとこにしましょう。

## 実験：赤外線特性の測定

まず、赤外線の測定値が取得できるプログラムを用意します。新たに開発するのが面倒であれば、WiinRemoteなどを使って測定してもかまいません。

図9-31 WiiRemote赤外線特性測定の配置(上面図)



いま、原点O(0,0,0)に自作のLEDセンサーバーの中心があり、センサーバー内にある2つのLED光源グループ間が、X軸方向に200mm離れているとして、これをLED2点間距離dと呼びます。

2つのLEDの中点がこれから実験する座標系の原点O、 $\{x,y,z\}=\{0,0,0\}$ にあり、測定に使用するWiiRemoteは座標P(x,y,z)にあるとします。WiiRemoteをセンサーバーからまっすぐ遠ざけていく方向を、求めたい「奥行きZ」、左右方向をX、上下方向をYと呼びます。Pの座標ではなく距離を表現するときは「PosZ」(単位はmm)と呼ぶことにします。

測定を始めましょう。測定しやすい床などの安定した場所にセンサーバーを置きます。このとき床面に赤外線が反射していると実験が失敗してしまうため、WiiRemoteやデジカメを使って、赤外線光の強い反射がないか確認しましょう。床がどうしても反射する場合は紙を使って反射を拡散させるとよいでしょう。センサーバーを三脚に乗せるなどしてもよいですが、できればWiiRemoteを同一平面に置いてください。

まず、WiiRemoteを原点Oに近い場所に置き(ぶつかってしまいますので)、ゆっくりとセンサーバーから遠ざけていきます。WiiRemoteとLEDの距離があまりに近すぎると測定できません。これは、赤外線の発光強度が強すぎたり、1つのLEDしかセンサーの視界に入らなかつたりすることに起因します。徐々に奥行きを広げていき、PosZ = 300mm程度の距離になると、個々の赤外線測定値IRX1, IRX2を読むことができるので、測定値を測定シート(Excelなどに直接入力してもよいでしょう)にメモしていきます。

今回の実験では奥行き方向の距離をそれぞれ

$$\text{PosZ} = \{330, 660, 990, 1320, 1650, 1980, 2310, 2640, 2970, 3300\}(\text{mm})$$

の10種類としました。

また左右方向の特性も確認するために、X方向にもそれぞれ

$$\text{PosX} = \{0, 330, 660\}(\text{mm})$$

の3種類で測定しています。すべての組み合わせで30通りあるので根気よく、流れをつかんで、2人一組など実験するとよいでしょう。

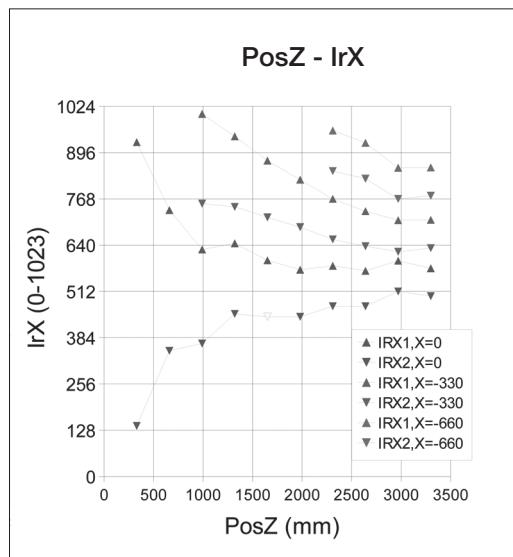
以下の表のように実測値をまとめます。

表9-1 【測定シート】赤外線センサーの特性測定実験 ( $d=200\text{mm}$ )

[Pos]	X = 0	X = 0	X = -330	X = -330	X = -660	X = -660
PosZ	IRX1	IRX2	IRX1	IRX2	IRX1	IRX2
330	925	140	—	—	—	—
660	737	348	—	—	—	—
990	628	368	1003	755	—	—
1320	645	450	941	746	—	—
1650	598	442	874	717	—	—
1980	572	442	821	690	—	—
2310	583	471	768	656	957	845
2640	569	471	734	637	923	824
2970	597	512	709	622	854	768
3300	576	499	710	632	855	777

「—」となっている箇所は赤外線LEDは常の値が読めなかった場所、つまり赤外線センサーの画角の外側です。実際に測定可能なポイントは22点になりました。このデータを PosZ を横軸、赤外線の測定値 IrX を縦軸としてプロットすると、図9-32 のようになります。

図9-32 WiiRemote 赤外線特性測定の結果



グラフを見たところ、IrX1 と IrX2 は奥行き PosZ に対して、単純な比例関係を持ってはいないようです。しかし左右に対してはほぼ対象といえるでしょう。そして PosX が中心から外れることで ( $\text{PosX} = -330\text{mm}, -660\text{mm}$ )、左右の対象性は崩れていくようです。

図9-33 左：実際の奥行き(PosZ) - 赤外線の差(IrZ)、右：PosXそれぞれに対するKの様子

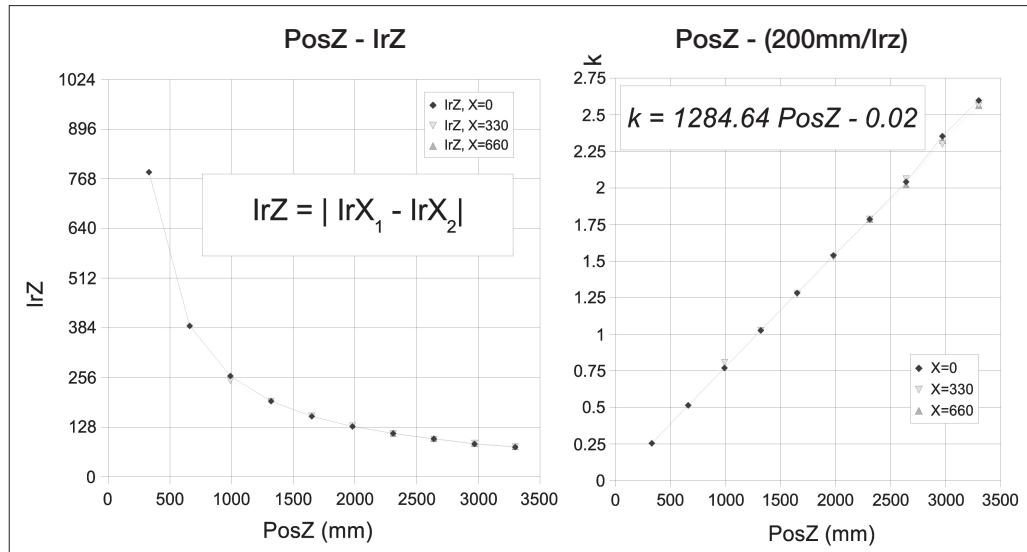
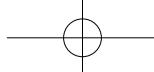


表9-2 各位置(PosX,PosZ)でのIrZ(2つのLEDの差の絶対値)

PosZ	X=0	X=330	X=660
330	785		
660	389		
990	260	248	
1320	195	195	
1650	156	157	
1980	130	131	
2310	112	112	112
2640	98	97	99
2970	85	87	86
3300	77	78	78

次に各位置での2つのLEDの差の絶対値IrZについて算出します。PosZを横軸、IrZと縦軸にとったグラフにプロットすると、PosZに対する反比例に見えます。今度はX={0, 330, 660}に対してそれぞれ、PosZを横軸、そして本来定数であるはずのK、すなわち「d / IrZ」を縦軸として図9-33(右)のようにプロットしてみます。

XがそれぞれX={0, 330, 660}と異なるにもかかわらず、見事に1本の直線に乗っています。この直線の傾きをPosZの最大-最小から求めると



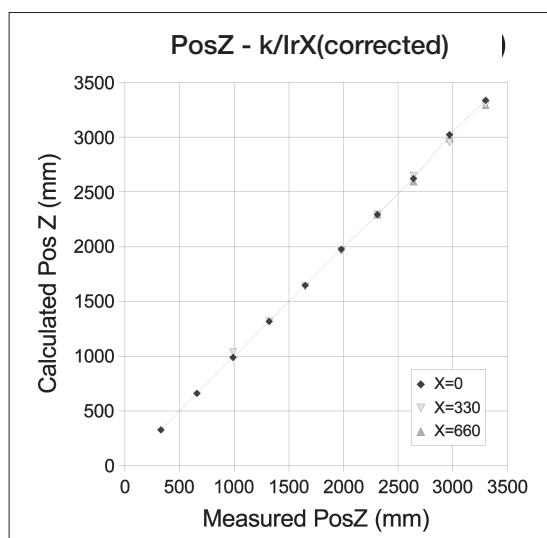
$$K = 1284.64 * \text{PosZ} - 0.02 \dots (9-3)$$

という、KとPosZの直線の1次方程式で表現することができます。

このKを用いて実測のPosZと、最大最小など2点程度のIrZを計測すれば、その間の奥行きZを簡単に求めることができます。

図9-34は、この理論を用いて算出した奥行きと、実測の奥行きがほぼ一致することを示しています。

図9-34 実測Z(横軸)－算出Z(縦軸)



### PRACTICE

#### 赤外線を極める

##### 【演習】☆

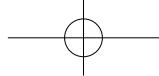
赤外線センサーの視野角(画角)は何度か。水平方向、垂直方向について最大値を測定し、角度として算出せよ。

##### 【演習】☆☆☆

本セクションで解説した理論を参考にして、赤外線の計測値と実際の奥行きをキャリブレーションするプログラムを作成せよ。

##### 【演習】☆☆☆☆

加速度センサーやアフィン変換を組み合わせて、より広範囲を検出できる仕組みを考えよ。



水平方向の画角についてはこの節で紹介したデータをもとに、算出することができます（意外と狭いです）。

広範囲化についてはさまざまな方法があり得ますが、次の図をヒントに考えてみるとよいのではありませんでしょうか。

図9-35 加速度センサーを使った広範囲化



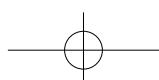
加速度センサーと連携する方法以外にもアイディアはあります。WiiRemoteの赤外線センサーはデジカメのCCDなどの画像センサーと異なり、歪んでもボケても得られる値は同じですので、「斜めから見る」という方法で実質の測定範囲を広くする方法はあるでしょう。

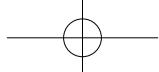
次の節で紹介する、ジョニー・リーのサンプルも、赤外線ポインタを「斜めに見る」ことで、広い範囲が測定可能になるコードを含んでいるようです。

## 9.10

# 研究編(2) : Johnny Chung Lee 氏から 学ぼう

アメリカ人のジョニー・リー氏(Johnny Chung Lee)は、カーネギーメロン大の学生当時、WiiRemoteを使ったプロジェクト「Head Tracking for Desktop VR Displays using the Wii





Remote (WiiRemoteを使ったデスクトップVRディスプレイのための頭部追従)」を2007年12月21日にYouTubeで公開し、今日まで721万回以上再生され、世界的に有名になりました。

図9-36 Mr. Johnny Chung Leeと彼のぼろぼろのタブレットPC(Laval Virtual 2008にて)



#### Johnny Chung Lee 氏の WiiRemote 関係のプロジェクト一覧

URL <http://johnnylee.net/projects/wii/>

他の2つの WiiRemote プロジェクトも、それぞれ200万回以上再生されています。

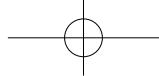
過去の WiiRemote 関係は氏の Web サイトにまとめられています。すべてソースコードと実行ファイルが入手できます。まずは、動画とともに以下の日本語解説を読んでみてください。

- WiiRemote で貴方の指をトラッキング (2007年11月08日公開)

赤外線LEDの行列を WiiRemote と統合し、再帰性反射材テープ(自転車の防犯反射テープのようなもの)を使って、映画「マイノリティリポート」調の空中多点操作を実現しています。WiimoteLib1.1 と C#、DirectX を使ってています。

- WiiRemote を使ったローコスト、複数点インタラクティブホワイトボード (2007年12月07日公開)

赤外線LEDを仕込んだペンを使って、インタラクティブなホワイトボードソフトウェアを実現しています。WiiRemote をスタンドに乗せてからプロジェクターに向かってペンを持つ構成になっています。特に、位置合わせのソフトウェアが秀逸です。WiimoteLib1.2.1 と C# で書かれています。オープンソース化され、Mac/Linux 版も公開されています。



- WiiRemoteを使ったデスクトップVRディスプレイのための頭部追従(2007年12月21日公開)  
逆転の発想です。テレビの前にWiiRemoteをおいて、センサーバー代わりの赤外線LEDをメガネの両脇につけます。「WiiDesktopVR」というデモプロジェクトが公開されており、頭を動かすと3DCGで描いた3次元的に配置されたものが視点にあわせて動きます。大きな風景写真なども視点にあわせて動きます。ちょうど窓枠を通してみるような感じです。WiimoteLibとC#、DirectXによるプログラムです。

「アイディア一発勝負！」の非常にシンプルなデモですが、動画公開の日付を見てもわかるように、開発のスピードがとても速いことも話題になりました。

またジョニーは研究者としてもしっかりしていて、WiiRemote以外にもローコストな特殊カメラや、プロジェクターを使って好きな場所(たとえば、手に持った扇)に好きな映像を投影する研究を行っています(だからこそWiiRemoteの活用も早かったのですね)。他にも写真作品や、巨大なペイントボールパチンコの制作など、いろいろ楽しいプロジェクトを実現されています。

現在、彼はMicrosoftの研究所で実用科学(Applied Sciences)グループで働いているため、表立ったWiiRemote関係の活動はありませんが、長い空白の後、なんと2009年6月1日のブログエントリーで「Project Natal」に関係していることを告白しました。

**procrastineering**(ジョニー・リー氏のブログ)

URL <http://procrastineering.blogspot.com/>

**Microsoft Project Natal**(動画)

URL <http://www.youtube.com/profile?user=xboxprojectnatal>

「Project Natal」とは、動画を見ていただければわかりますが、任天堂Wiiに対抗するマイクロソフトの全身型ゲームインターフェースです(詳細は次の章で紹介します)。

### 研究しよう

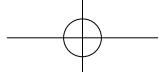
### PRACTICE

#### 【演習】☆☆

ジョニー・リーのサンプルを実行し、動作を確認せよ。赤外線LEDグッズの作成は第9.6章を参考にし、可能であれば日本語で返信YouTube動画をアップロードせよ。

#### 【演習】☆☆☆☆

サンプルをリビルドし自分のプロジェクトに再利用せよ。特にアフィン変換やデモ映像部分などは便利でしょう。



彼の研究者としての論文も非常におもしろいです。関連するおもしろい論文があればどんどん読んでみましょう。なお、海外の論文をすばやく調べるときには、「Google Scholar」、公開されているプログラムコードを検索するときは「Google Code」が役に立ちます。日本語の論文が気になるときは「CiNii」という国立情報学研究所が運営している論文検索エンジンを使うとよいでしょう。

#### 論文検索「Google Scholar」

URL <http://scholar.google.com/intl/ja/>

#### 国立情報学研究所「CiNII」(キーワード "Wii" で検索)

URL <http://ci.nii.ac.jp/search?q=Wii>

#### Google Code

URL <http://code.google.com/>

9.11

## プログラミング編 (2)： 自分で API を作る

本章の最後は、WiiRemoteにアクセスするAPIを自分で作る課題です。DDKを使った、C++のプログラミングです。特に必要な方は読み飛ばしていただいてかまいませんが、他の言語な

#### APIを使わない接続についての情報

##### WiiMedia2

URL <http://code.google.com/p/wiimedia/source/browse/#svn/trunk/Wiimedia2/>

Google Codeで公開されている、DDKとC++を使った機能最小限の自作APIです。

##### 筑駒パ研「電腦2007：Wiiリモコンをもう一回見直してみます」by lketaki

URL [http://paken.s1.hayasoft.com/files/down/denno2007\\_wii.pdf](http://paken.s1.hayasoft.com/files/down/denno2007_wii.pdf)

筑波大学附属駒場中・高等学校パーソナルコンピュータ研究部の文化祭で発行された部誌のPDF版です。非常に丁寧に解説されております。

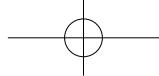
どに WiiRemote の API を移植するときなど、役に立つかかもしれません。

WiiMedia2 から重要な OpenWiiRemoteHID() 近辺を引用しておきます。

#### コード9-7 wm\_base.cpp wm\_base::OpenWiiRemoteHID()

```
HANDLE wm_base::OpenWiiRemoteHID(void) {
//LONG iHIDs;
    DWORD indexHID = 0;
    BOOL bEndofDeviceList = FALSE;
    BOOL bDeviceDetected = FALSE;
    PSP_DEVICE_INTERFACE_DETAIL_DATA DeviceDetail = NULL;
    DWORD size = 0;
    DWORD RequiredSize;
    HIDD_ATTRIBUTES Attributes;
    HidD_GetHidGuid( &guidHID );
    hDeviceInfo = SetupDiGetClassDevs(
        (LPGUID)&guidHID, NULL,
        (HWND)NULL,
        DIGCF_INTERFACEDEVICE | DIGCF_PRESENT
    );
    if ( 0 == hDeviceInfo ) { return INVALID_HANDLE_VALUE; } //失敗
do {
    bDeviceDetected=FALSE;
    //HIDインターフェースを列挙
    deviceInfoData.cbSize = sizeof(SP_DEVICE_INTERFACE_DATA);
    if ( SetupDiEnumDeviceInterfaces(
        hDeviceInfo, NULL, &guidHID, indexHID, &deviceInfoData)!=0 ) {
        printf("[%d] HID found.\n",indexHID);
        //列挙したHIDの詳細を取得
        SetupDiGetDeviceInterfaceDetail(hDeviceInfo,
            &deviceInfoData, NULL, 0, &size, NULL );
        DeviceDetail = (PSP_DEVICE_INTERFACE_DETAIL_DATA)malloc(size);
        DeviceDetail -> cbSize = sizeof(SP_DEVICE_INTERFACE_DETAIL_DATA);
        printf("HID detail size =%d.\n",DeviceDetail->cbSize);
        SetupDiGetDeviceInterfaceDetail (hDeviceInfo,
            &deviceInfoData, DeviceDetail, size, &RequiredSize, NULL);
        hWiiRemoteHID = CreateFile( DeviceDetail->DevicePath,
            GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ|FILE_SHARE_WRITE,
            (LPSECURITY_ATTRIBUTES)NULL, OPEN_EXISTING, 0, NULL);
        bDeviceDetected = FALSE;
        Attributes.Size = sizeof(Attributes);
        if ( HidD_GetAttributes( hWiiRemoteHID, &Attributes ) ) {
if ( Attributes.VendorID == 0x057e && Attributes.ProductID == 0x0306 ) {
if ( HIDP_STATUS_SUCCESS == GetDeviceCapabilities( hWiiRemoteHID ) ) {
            printf(" WiiRemote found.[V=0x%04d,P=0x%04d]\n",
                Attributes.VendorID,Attributes.ProductID);
```

次ページにつづく ➞



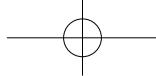
```
bDeviceDetected = TRUE;
} else {
    printf(" GetDeviceCapabilities() failed.\n");
}
} else {
    printf(" It didn't match with WiiRemote.[V=0x%04d,P=0x%04d]\n",
        Attributes.VendorID,Attributes.ProductID);
    CloseHandle( hWiiRemoteHID );
}
} else {
    printf(" HidD_GetAttributes() failed.\n");
    CloseHandle( hWiiRemoteHID );
}
}
free(DeviceDetail);
} else {
    bEndofDeviceList = TRUE;
}

indexHID++;
} while ( (bEndofDeviceList == FALSE) && (bDeviceDetected == FALSE) );

if ( bDeviceDetected == FALSE ) {
    printf("Finally, I couldn't find any WiiRemote.\n");
    hWiiRemoteHID = INVALID_HANDLE_VALUE;
} else {
    printf("Yes, I found a WiiRemote.\n");
}
SetupDiDestroyDeviceInfoList(hDeviceInfo);
return hWiiRemoteHID;
}
```

最初から手探りで作るのは大変ですから、もし自分自身でAPIを作成に挑戦する場合、上のコードの他に、WiimoteLib や WiiYourself! を参考にすると良いでしょう。

「GetDeviceCapabilities()」近辺が重要で、これが個々のPC環境やBluetoothスタックによって異なります。それらに対して、高度に完成しているAPIはWriteFile()などを工夫して、確実に通信が行えるようにしています。その他、レポートタイプの設定などは「Wiili.org」や「WiiBrew」などのWikiサイトで情報をあつめて構築していきます。

**PRACTICE**

このセクションでの演習問題は難易度を特に高く設定しています。特に腕に自信がある方のみ挑戦してみてください。

**【演習】☆☆☆**

DDKとC++を使って自分でAPIを開発せよ。さまざまなスタックで動作するよう、WiiYourself! やWiimoteLibを追加を参考にするとよい。

**【演習】☆☆☆☆**

Windows Mobileなど他のプラットフォーム用にAPIを移植せよ。

**【演習】☆☆☆☆☆**

スタックやDDKを活用し、Bluetooth接続を自動化できるドライバーを開発せよ。

これで本章は終わりです。今まで通りステップバイステップの解説を行ったところもありますが、ほとんどがアイディアの要点だけ、あとは演習問題という構成で解説よりも、課題設定に力点を置かせていただきました。

読者のスキルを幅広くとった本書において、英語で書いた論文をそのまま落とし込むのは難儀しました。残念ながら割愛したネタも数多くありますが、筆者の経験したWiiRemoteプロジェクトのいくつかを、初めて日本語で解説する機会に恵まれました。

本書でWiiRemoteプログラミングを学んだみなさんが、本書での「ネタ」をきっかけに、YouTubeなどを通して、より多く世界中の人々と交流されることを祈っています。その際「元ネタはWiiRemote本より」と、本書を引用元に書いていただけたとより励みになります。

またWiiMotionPlusの登場などで、本書の内容も古くなったり「もっといい方法があるよ！」といったご意見もあると思います。上記で紹介した「WiiMedia」プロジェクトや、Google Groupsにてコミュニティを立ち上げておりますので、ご活用いただければ幸いです。

## オンラインコミュニティ

**筆者HP**

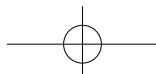
**URL** <http://akihiko.shirai.as/projects/WiiRemote>

本書のポータルとしてここに情報をまとめています。

**Google Code「WiiMedia」**

**URL** <http://code.google.com/p/wiimedia/>

本書で紹介したサンプルなど関連のコードをここで共有しています。



Google Groups「WiiRemote」

URL <http://groups.google.com/group/wiiremote>

ご質問や「こんなことできたよ！」という情報をお寄せ下さい。