アイディアストック・演習問題集

白井暁彦
shirai at mail.com
http://akihiko.shirai.as/projects/BookWii/

21:40 2009/06/15-

目 次

0.1 プログラミング編 (1)：XNA を使ったリアルタイム 3DCG での利用 1
0.1.1 XNA のインストール 2
0.1.2 ゲームプロジェクトの作成 3
0.1.3 WiimoteLib の組み込み 5
0.1.4 3D モデルファイルの準備 6
0.1.5 .x ファイルの読み込みと表示 7
0.1.6 完成版「WiiRemoteXNA」 10
0.1.7 【演習問題】XNA の利用 12
0.2 ゲーム応用編 (1)：レースゲームへの応用 13
0.2.1 反重力レーシングゲーム「AceSpeeder2」 13
0.2.2 レースゲーム用モーション認識 14
0.2.3 【演習問題】レースゲームへの応用 15
0.3 ゲーム応用編 (2)：「振る」の認識・剣術アクションへの応用 . 18
0.3.1 フェンシングゲーム「JaWii's Virtual Fencing」 . . . 18
0.3.2 古式フェンシング指南ゲームの開発 18
0.3.3 モーション検出のための評価関数を作る 22
0.3.4 【演習問題】振りの検出 25
0.4 作品編 (1)：WiiBoard を用いた「オーラ診断」 27
0.4.1 プログラミング初心者 4 人が作った「オーラ診断」 . . 27
0.4.2 補足：「BBOSC」とは？ 30
0.4.3 その他の「オーラを作る技術」 31
0.4.4 【演習問題】Mac と WiiBoard の利用 31
0.5 作品編 (2)：SecondLife で使う 35
0.5.1 SecondLife 用 PIE スクリプト 35
0.5.2 【演習問題】SecondLife ＋ GlovePIE を極めよう . . . 37
0.6 モノ編 (1)：センサーバーを自作する 39
0.6.1 センサーバーのしくみ 39
0.6.2 目には見えない「近赤外線」 39
0.6.3 電子部品をそろえる・工作する 41
0.6.4 【演習問題】赤外線センサーバーの自作 45
0.7 モノ編 (2)：ロボット兵器「WiiRemoteTank」 47

0.7.1 ロボット兵器「WiiRemoteTank」 47
0.7.2 「WiiRemoteTank」の開発 48
0.7.3 【演習問題】WiiRemote によるロボット開発 51
0.8 サービス編：体が不自由な方のためのインターフェース 52
0.8.1 【演習問題】体が不自由な方のためのインターフェース 52
0.9 研究編 (1) 赤外線を極める 54
0.9.1 赤外線奥行き測定の基本理論 54
0.9.2 実験：赤外線特性の測定 55
0.9.3 【演習問題】赤外線を極める 59
0.10 研究編 (2)Johnny Chung Lee 氏から学ぼう 61
0.10.1 【演習問題】研究しよう 63
0.11 プログラミング編 (2)：自分で API をつくる 64
0.11.1 【演習問題】 66

この章は「アイディアストック・演習問題集」として、いままで学んだ技術を応用することで、WiiRemote をつかって実現できる様々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」と名付けました。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様々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。

各セクションの終わりに「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。

0.1 プログラミング編 (1)：XNA を使ったリアルタイム 3DCG での利用

8 章までの知識を一歩進めて、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しょう。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。

このセクションでは、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「WiiRemo-

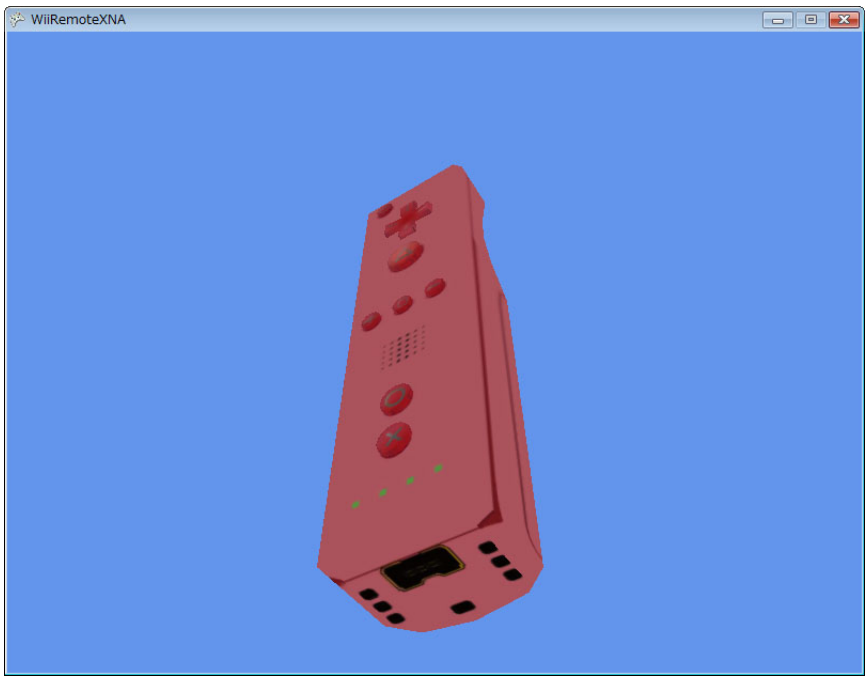


図 1: 「WiiRemoteXNA」完成版

teXNA」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。

0.1.1 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 をダウンロードしてインストールします。

Microsoft XNA Game Studio 3.1

Microsoft XNA Game Studio 3.1 も無償で入手可能です。
http://www.microsoft.com/downloads/details.aspx?familyid=80782277-D584-42D2-8024-893FCD9D3E82

XNA は無料で開発環境を手に入れることができます。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※) 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、Xbox 360 のような、高価なゲーム用 PC に比べて「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、それほど高価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...)。

Xbox 360 用ゲームの開発

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。

[URL] http://www.xbox.com/ja-JP/live/

[URL] http://creators.xna.com/en-US/tour_detail

図 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。

0.1.2 ゲームプロジェクトの作成

さて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

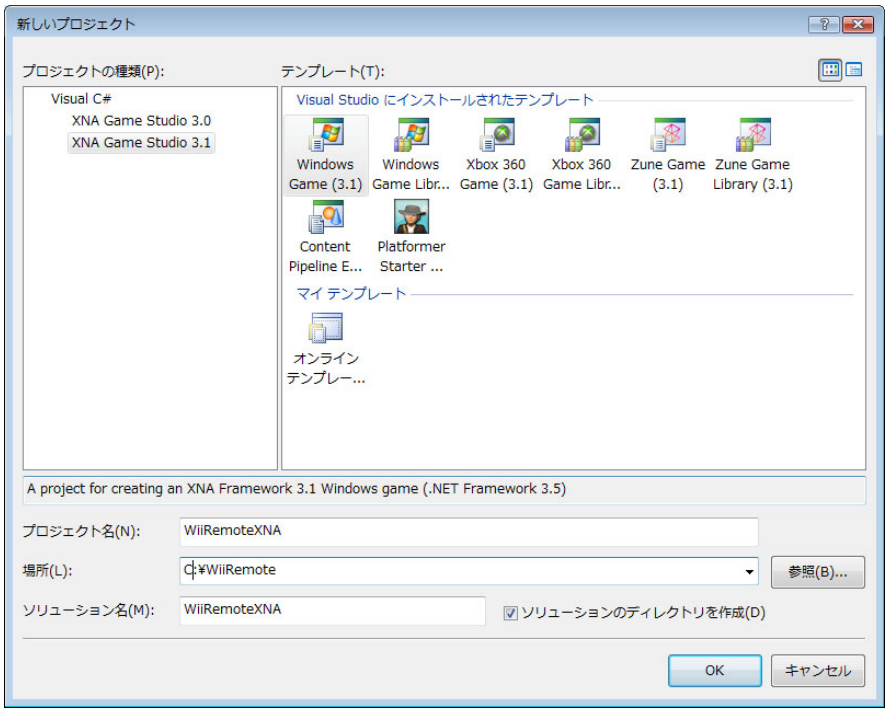


図 3: 新規プロジェクトの作成に Game Studio 3.1 のテンプレートが現れる

(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。

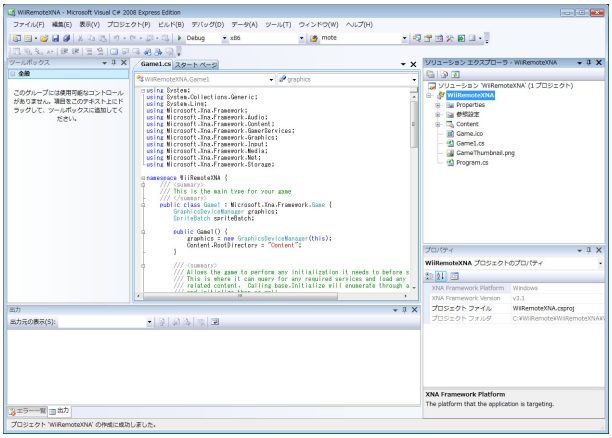


図 4: Visual Studio に作成された新しいゲームプロジェクト

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。

0.1.3 WiimoteLib の組み込み

次は WiimoteLib を組み込みます。XNA 環境でも 4 章や 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。

いま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。

リスト 1: [C#]Wiimote オブジェクトの作成 (Game1.cs)

```
using WiimoteLib;
<略>
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
<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。

リスト 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ります)。

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良いのですが、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。

リスト 3: [C#]Home ボタンで終了 (Game1.cs)

```
protected override void Update(GameTime gameTime) {
// Allows the game to exit ↓もともとのコードをコメントアウト
// if (GamePad.GetState(PlayerIndex.One).Buttons.Back
                                == ButtonState.Pressed)
 if(wm.WiimoteState.ButtonState.Home) {
   this.Exit();
 }
 base.Update(gameTime);
}
```

この段階で WiiRemote を Bluetooth 接続し、[F6] 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。

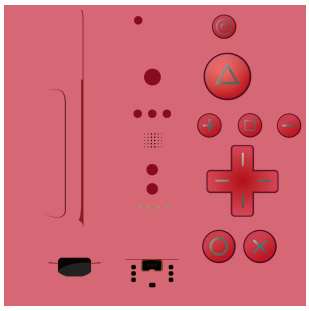
0.1.4 3D モデルファイルの準備

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ってしまいますので、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。

3D モデルファイル「wiimodoki.x」の入手

■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
■特製テクスチャファイル (texture_wii.jpg)



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。

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。

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」を使って作成したのち「cvXporter」を使ってコンバートしています。

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 ポリゴンモデラー「Metasequoia(メタセコイヤ)」でも可能でしょう。無料版と有料版 (シェアウェア) 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。

0.1.5 .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。

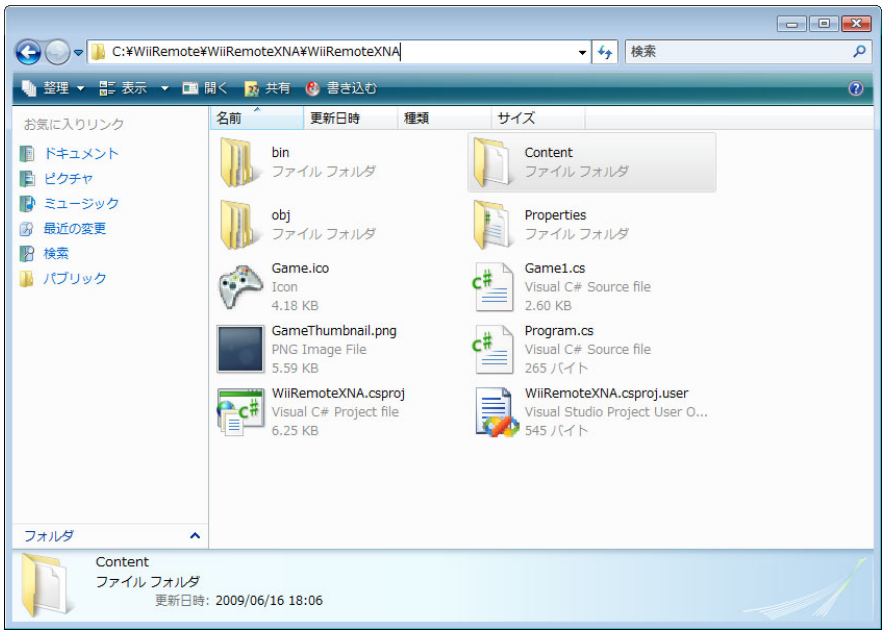


図 5: WiiRemoteXNA の「Content」に.x ファイルを配置する

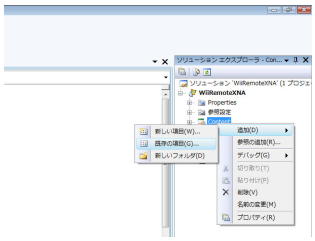


図 6: ソリューションエクスプローラーの「Content」に追加する

Maya からの.x ファイルエクスポーター「cvXporter」

[URL] http://www.chadvernon.com/blog/downloads/cvxporter/

3D ポリゴンモデラー「Metasequoia(メタセコイヤ)」

[URL] http://www.metaseq.net/

リスト 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 );
   }
  }
 }
<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しらべたり、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] で終了します。

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

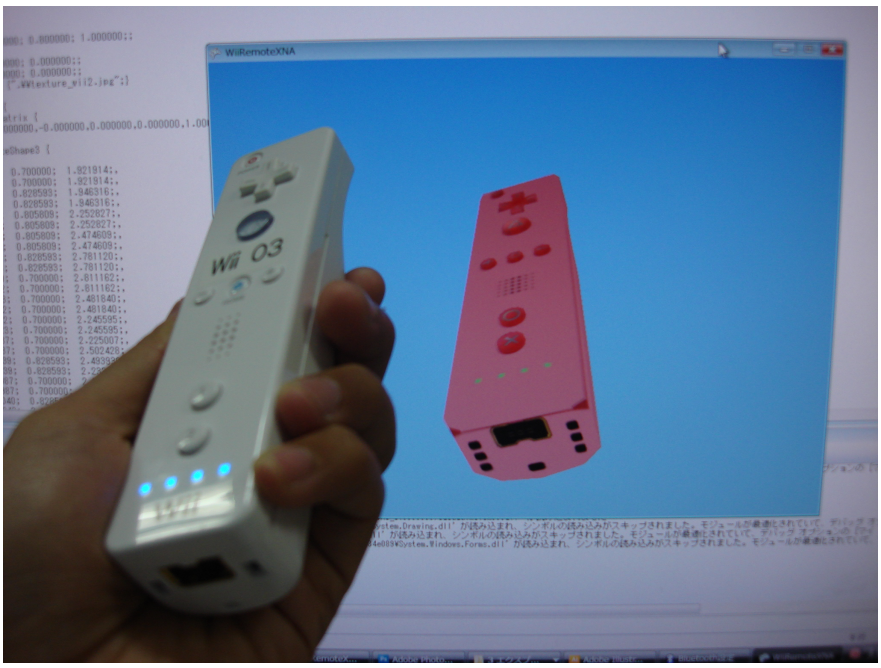


図 7: WiiRemote の動きに合わせて動く「WiiRemote もどき」

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もできます。

0.1.6 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。

リスト 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();  //WiiRemote 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(
      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);
  }
 }
}
```

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。

0.1.7 【演習問題】XNA の利用

【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ましょう。余裕があれば、「リスト」をつかって最近 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】☆☆ Metasequoia をつかって、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 モデルビューアー」としての機能を加えてみましょう。

0.2　ゲーム応用編 (1)：レースゲームへの応用

ここから先は内容によって、今までのようなステップバイステップの解説手法をとりません。アイディアのみをかいつまんで、自分のプロジェクトに利用していきましょう。

このセクション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。

0.2.1　反重力レーシングゲーム「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 8: 反重力レーシングゲーム「AceSpeeder2」

RAINGRAPH スタジオ「AceSpeeder2」

[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、既に 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。

0.2.2　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。

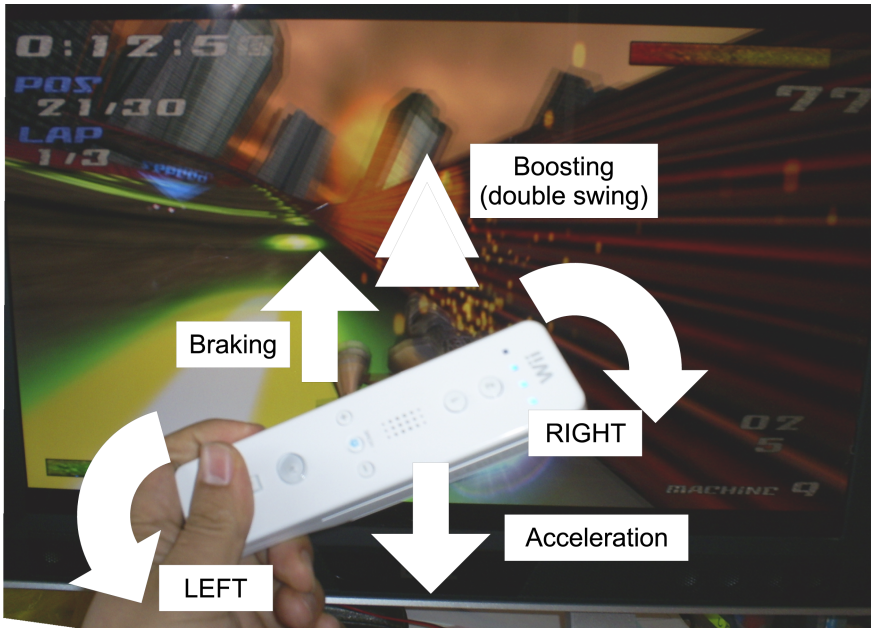


図 9: WiiRemote によるコントロール

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、エネルギーをためて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思で『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、つまり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。

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

実際の展示を通したユーザーテストでの観察によると面白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も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ていませんが、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。

0.2.3 【演習問題】レースゲームへの応用

加速度センサーによる傾きの検出は Atan2() 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既に 7 章、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 をつかって傾きをアナログ入力に利用せよ。

【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。

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、ということきは逆転の発想で 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。

以下、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。

AceSpeeder2 WiiMedia Edition(Youtube 動画)

[URL] http://www.youtube.com/watch?v=KowXAXdfO8E

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良いのですが、「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)といかに親和性を保ちながら、インタラ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」をインターフェースさせるための「理想の姿勢」というものを考えて、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。

余談「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と、つくづく思います。
[URL] http://sandbox.siggraph.org/about.html

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

0.3　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。

このセクション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」(ジャウィのバーチャルフェンシング) の開発をベースに WiiRemote をつか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。

0.3.1　フェンシングゲーム「JaWii's Virtual Fencing」

この作品「ジャウィ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。

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) 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・ジャリィ (Alfred JARRY, 1873〜1907) の没後 100 年祭を祝っていました。

ジャリィは作家としては「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「ジャリィ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、市民にジャリィの人物を伝える良い企画として持ち上がったのです。

0.3.2　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ありますので、ご参照下さい。

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

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。

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」のに技術よって実現しています。この会社のホームページ (http://www.xavix.jp/) にいくと、技術の高さと様々な産業へのインタラクティブ技術のインパクトがうかがい知れます。

図 10: 「ジャウィ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
Laval Virtual (日本語ページあり) [URL] http://laval-virtual.org/

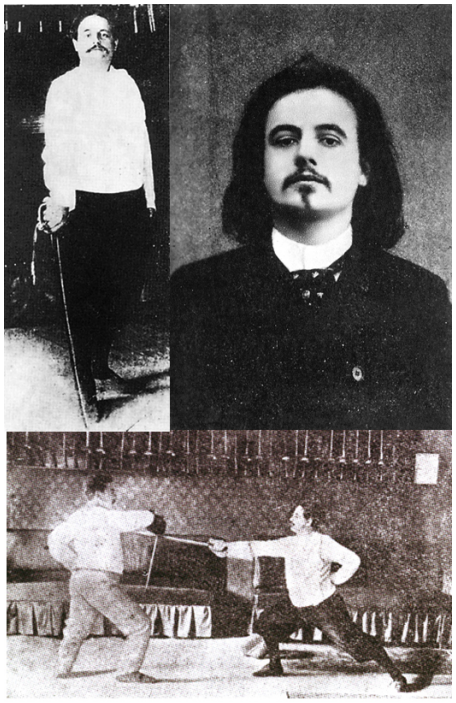


図 11: アルフレッド・ジャリィと古式フェンシング

「ジャウィのバーチャルフェンシング」(Youtube 動画)

WiiMedia:Sword Fighting ”JaWii’s Virtual Fencing”
[URL] http://www.youtube.com/watch?v=Kl_-KoVLtx4

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、防具もつけません。

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、子供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、ジャリィがたしな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ー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 12: 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、プレイヤー自己視点で遊びます。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 13: リアルタイムアニメーションは Virtools で開発

0.3.3 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 プログラミングの課題です。

プロジェクター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、大人や子供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つく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 例えば「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。最大の加速度を取得してから「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 9-14 は加速度センサーの加速度データひとつひとつを 3 次元ベクトルにして積分し、得た速度をもとに、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。

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 7 章で WiiRemote を「そーっとうごかすと検出されない」という実験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面白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。

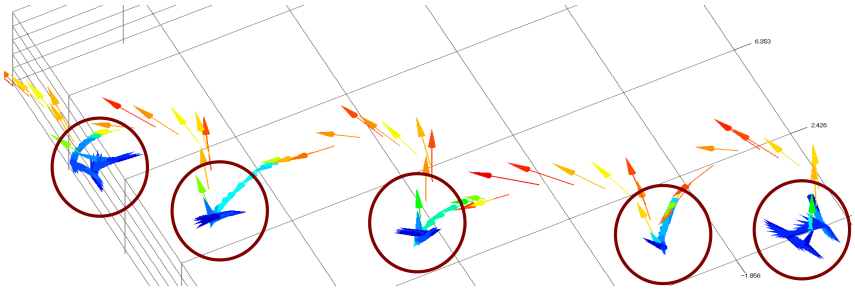


図 14: 加速度センサーの値を積分して得られる 3 次元位置

実際にはこの瞬間、WiiRemote は頭の後ろで手首をつか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もっているので、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。

以後、このような「評価関数」を作りこ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それぞれを作っていきます。

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 9-15 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。

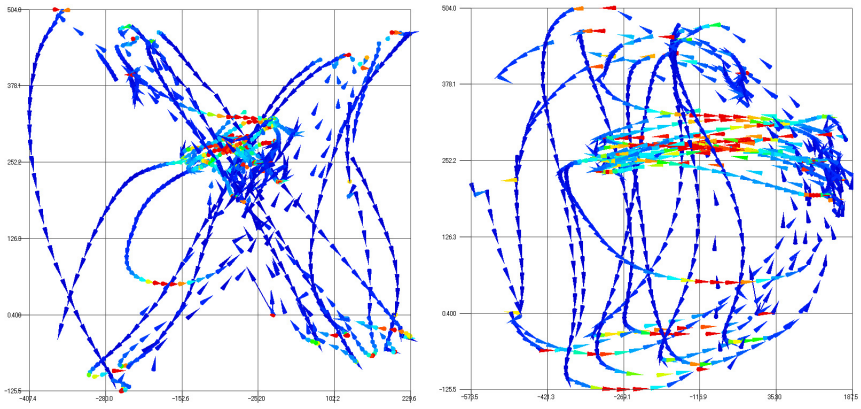


図 15: モーションキャプチャーによる「振りの」様子 (左：正面図、右：上面図)

モーションキャプチャは高価な機材なので気軽に使うことはできませんが、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の近いものを 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つかって、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 9-15 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) に対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無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ますので、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「ジャウィ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。

0.3.4 【演習問題】振りの検出

【演習】☆☆ 第 7.5 章の「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ば、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ね (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面白味がありますが、最後はこの評価関数のデザインセンスが、ゲームの面白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

プログラマー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。

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。ゲームの流れや面白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 関係のプラグインを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/) で様々な情報やソース、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さんや、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。

■株式会社クレッセント　http://www.crescentvideo.co.jp/virtools/
■三徳商事　http://virtools.jp/

0.4　作品編 (1)：WiiBoard を用いた「オーラ診断」

次のテーマは「バランス Wii ボード」を扱います (本書では「WiiBoard」と標記しています)。

第 1.4 章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。

このセクションでは 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) の皆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。

0.4.1　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。

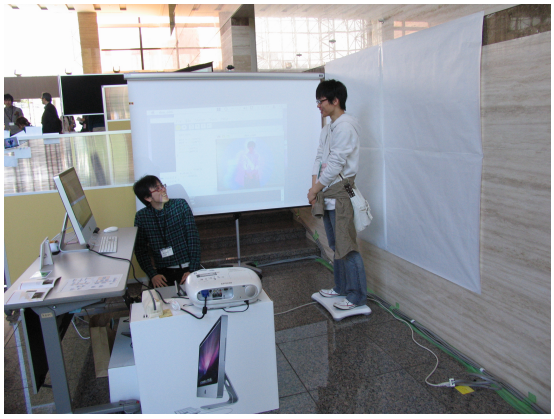


図 16: Mac と Processing と WiiBoard で作った「オーラ診断」

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そのまま引用しています)。

「オーラ診断」

YouTube 動画「オーラ診断」
[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

「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。

図 17: 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事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、入力した事を気がつかせない。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ことができます。具体的には、左右の足の前後、合計 4 カ所にかかる重さを得ており、それぞれを比べる事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事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 カ所の体重の情報を、Processing に渡すわけです。このあたりは、Web Designing 3 月号 (2008 年) の記事 [Beyond the Browser] を参考にさせていただきました。

大雑把に言えば、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

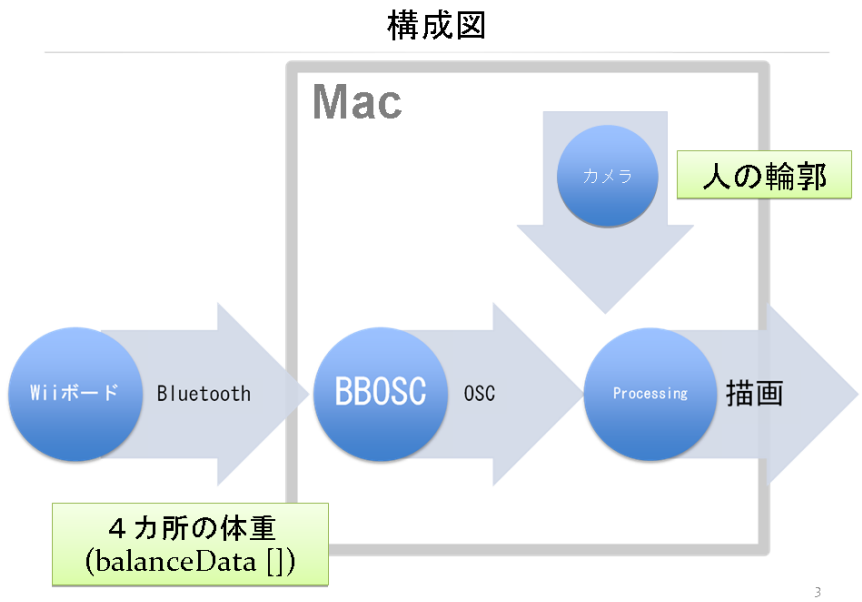
Processing はビジュアル表現が容易なオブジェクト指向のプログラミング言語、らしい。はっきり言って名前すら知りませんでした。

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。そう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。



0.4.2　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すと、情報が足りません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「ウェブデザイニング」連載記事のために開発したもので、WiiRemote で Mac を操作できる「DarwiinRemote」などを開発した Hiroaki Kimura 氏による、MacOS 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。

参考：Hiroaki Kimura 氏のブログ「Hirolog」

「DarwiinRemote」WiiRemote で Mac を操作できる
[URL]　http://blog.hiroaki.jp/2006/12/000433.html
「WiiRemote Framework」
[URL]　http://blog.hiroaki.jp/2007/05/000456.html

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つく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 (http://opensoundcontrol.org/)。

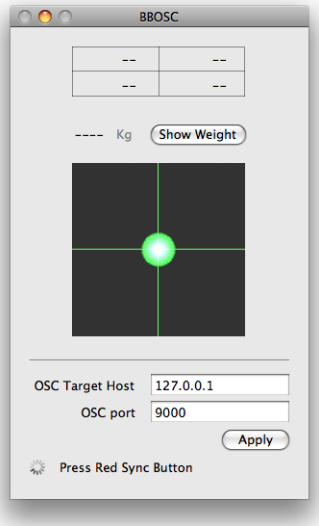


図 18: 「BBOSC」

「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 カ所にかかる体重を 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。

0.4.3 その他の「オーラを作る技術」

「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。

0.4.4 【演習問題】Mac と WiiBoard の利用

なお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」のホームページには「オーラ診断」の他にも面白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「バックトゥ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。

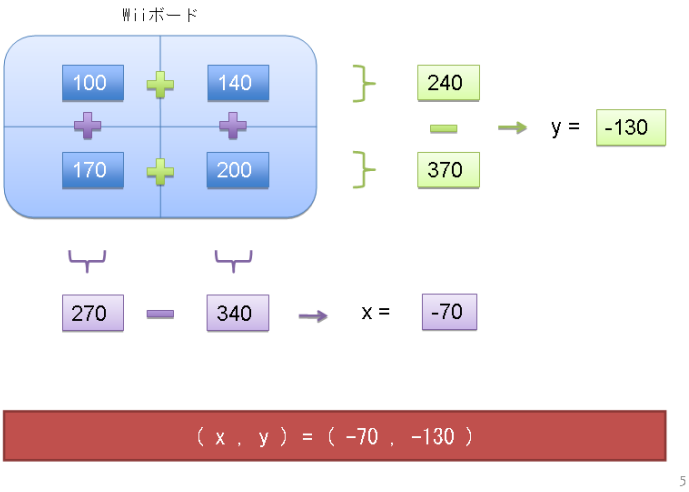


図 19: WiiBoard から得られるセンサー値の利用

その他：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)。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ーを使った作品を作った人です。
「Myron Krueger」(YouTube 動画)
[URL] http://www.youtube.com/watch?v=A6ZYsX_dxzs
「blobDetection」[URL] http://www.v3ga.net/processing/BlobDetection/
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」のことで、人物などの検出をするには向いています。なおこのホームページ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。

【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。なお「足踏み動作解析」については、東京農工大の藤田欣也先生他、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。

このセクションでは WiiBoard と MacOS での学生プロジェクトを扱いました。本書では MacOS での WiiRemote プログラミングを Procsssing と ActionScript 以外は扱ってきませんでしたが、Bluetooth 接続の安定感もあり「WiiRemote Framework」など、WindowsXP 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。

またこのセクションで扱ったような 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 をつかった VJ 活動などもよくききます。Windows 環境だけにとらわれる必要は無いのです。

「オーラ診断」で使ったような、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。

東京工科大学「メディアコンテンツ展 2009」

「ライフエンタテインメント」
[URL] http://www.teu.ac.jp/mce/2009/work/lifeenter.html

「WiiBoard」C#.NET での開発

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良いでしょう。
[URL] http://www.kosaka-lab.com/tips/2009/02/wiiwii-fit.html

0.5　作品編 (2)：SecondLife で使う

世界的に有名なバーチャルワールドサービス「SecondLife」で WiiRemote を使えるようにしてみましょう。SecondLife 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」を想像すると良いでしょう。

SecondLife 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 SecondLife を WiiRemote で操作することができるようになります。SecondLife 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の渡邉英徳先生が、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ので紹介します。

図 20: Second Life 作品「Laval VRchive 2009」

0.5.1　SecondLife 用 PIE スクリプト

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 SecondLife 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良いので、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、SecondLife 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。

SecondLife 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、ディベロッパーモード [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] 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 21: WiiRemote を使って片手で操作できる

リスト 6: [GlovePIE]SecondLife「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＞
```

0.5.2 【演習問題】SecondLife + GlovePIE を極めよう

【演習】☆☆☆ 上記 PIE スクリプトと、本書 3 章を参考にして、「GlovePIE」と「SecondLife」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。

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。赤外線を見せるなどして、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います。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が...)。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。

0.6　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、このセクションは読み飛ばしていただいてもかまいません。

0.6.1　センサーバーのしくみ

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の自由度も広がるでしょう。

0.6.2　目には見えない「近赤外線」

さて、ここでは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。

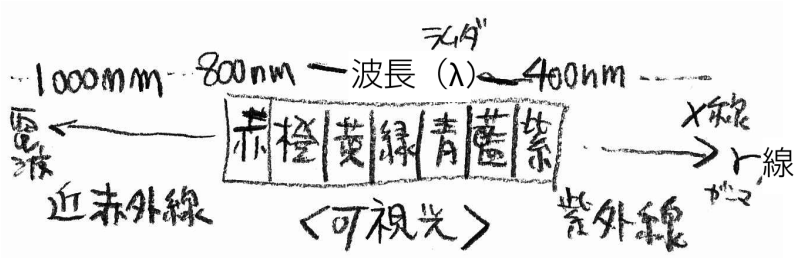


図 22: 目には見えない近赤外線

「近赤外線」とは

『赤外線』と一言で言っても、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 μ m〜1000 μ 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……と波長ごとにいろん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、人間の目に見えづらいという以外は、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 (フォトインタラプタ) 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 μ m の波長、つまり遠赤外線です。

見えない赤外線を見えるようにするには？

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、可視光や熱は必要でなく、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 光源にあわせて、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。

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つかって、夜中や暗闇でも撮影できるカメラを実現しています。

0.6.3 電子部品をそろえる・工作する

こちらが自作 USB センサーバーの回路図の例です。

材料としては、赤外線 LED と抵抗、基盤、半田ごて一式、あとは不要な USB のケーブルを 1 本、切断して作ります。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

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」が非常に良くできています。波長毎、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」
製作著作：文部科学省／監修：河田聡（独立行政法人理化学研究所）
http://www.riken.go.jp/r-world/topics/080404_2/lightmap.pdf

図 23: センサーバーの赤外線をデジカメで撮影した様子

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るのでしょう？

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 社 (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 年の時に書いた論文「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、浜松ホトニクス社製の 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を記憶しています... !

【注意】

[!] ここ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。

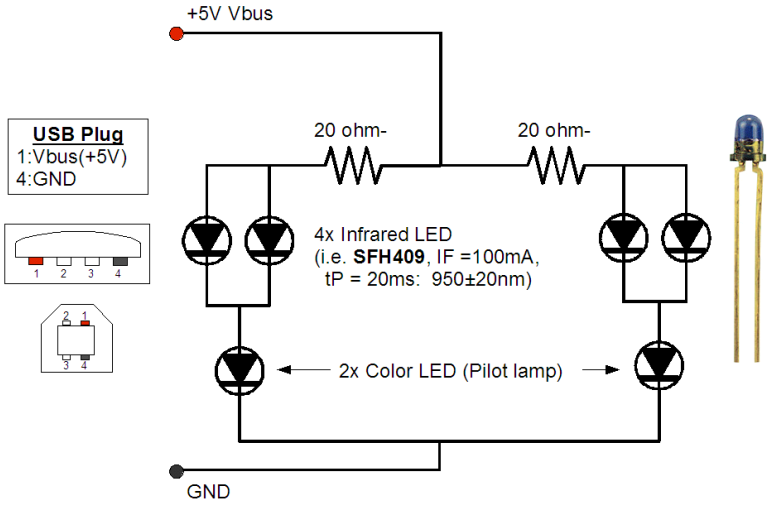


図 24: 自作 USB センサーバーの回路例

番を回路の+5V に、USB4 番を回路の GND にそれぞれ半田をつかってつなぎます。

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
http://akizukidenshi.com/catalog/g/gI-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)。なおここで紹介した「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いっしょに買ってください。制限抵抗を間に入れないと、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。

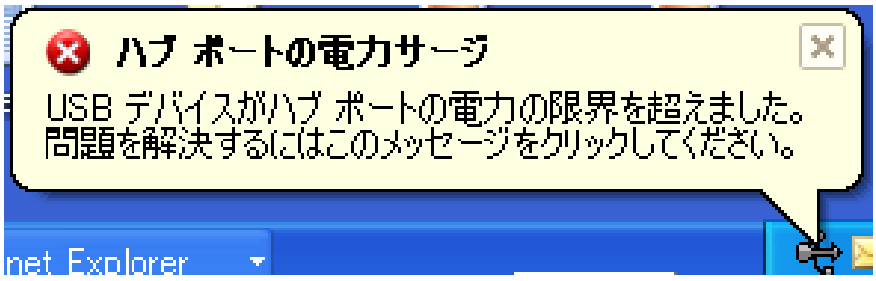


図 25: USB 電力のサージ警告

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ます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てみたり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。

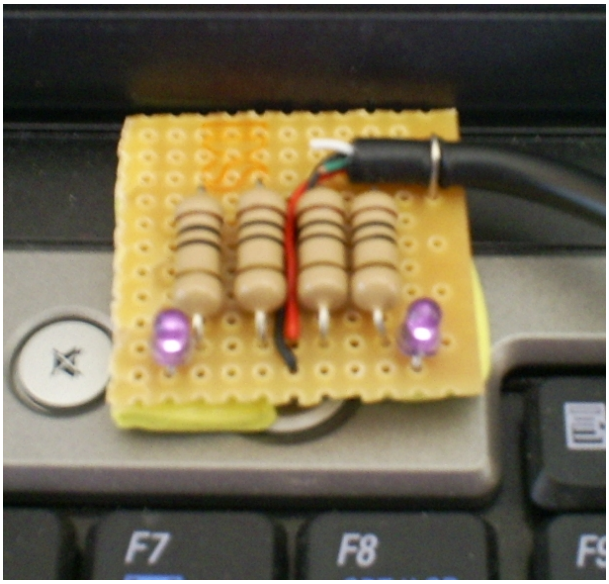


図 26: 自作の極小 USB センサーバーの制作例

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。

0.6.4 【演習問題】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」、解答例は Youtube 動画「Sound Quest V1」(http://www.youtube.com/watch?v=TMK7ULUG7S4) がある。

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 点まで検出できますので、三角形の向きが拾えたり、面が推定できたりと、いろいろな応用があります。

図 27: 赤外線 LED4 点で二等辺三角形を検出させる例

0.7　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 をつかってラジコンカーを操作

http://gigazine.net/index.php?/news/comments/20061222_wii_rc/
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」

もうひとつの動画も同様で、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。

0.7.1　ロボット兵器「WiiRemoteTank」

人がやったことをただ真似ても面白くありません。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。

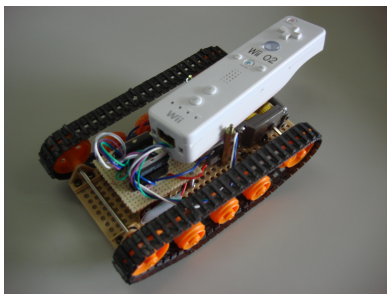


図 28: ロボット兵器「WiiRemoteTank」

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ますので、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。

0.7.2 「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 細田耕先生による「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

■ 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

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 つあるので、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可能です。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。

[URL] タミヤ タンク工作基本セット

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。

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。

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。

WiiRemote を「分解」...？

小坂先生は「WiiRemote を分解」とあっさり書かれていますが、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。もちろん全く保証外の行為ですが、そのようなドライバーなど無くても開けようと思えば開けることはできます。

【参考】[URL] http://ameblo.jp/akihiko/entry-10056910390.html

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。

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

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。

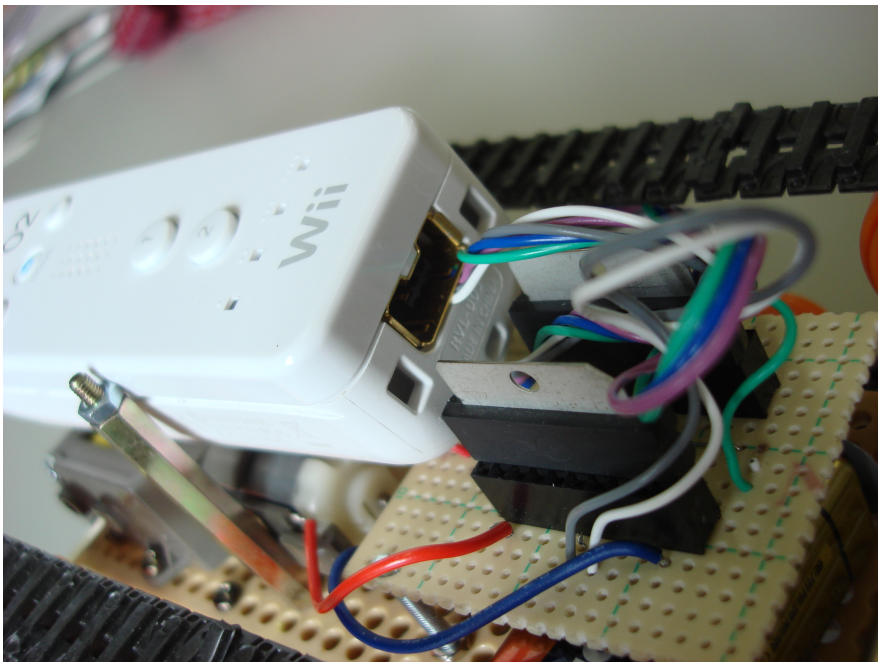


図 29: LED 信号をモータードライバーに接続

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 などで操作しても面白そうです。

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。

0.7.3 【演習問題】WiiRemote によるロボット開発

【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。

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)。

0.8 サービス編：体が不自由な方のためのインターフェース

このセクションでは、第 8 章で学んだ技術を応用して、ハンディキャップのある方に WiiRemote を使って、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。

0.8.1 【演習問題】体が不自由な方のためのインターフェース

【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った人や、一時的に不自由なひとなどそれぞれです。

そんな方々に WiiRemote だけでブラウザー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。

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。なんと言っても 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。

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、ミルクをあげている時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、ブラウザー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っておけばよかった !! 」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を「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。

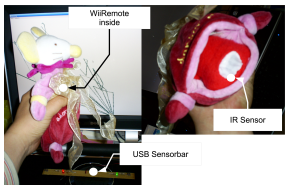


図 30: 幼児向けラクガキシステム「Papier Poupee Painter」で使った「安心ぬいぐるみ」

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、ラクガキの面白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 (YouTube 動画)
[URL] http://www.youtube.com/watch?v=S8kYQbfN_9I

0.9　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。

0.9.1　赤外線奥行き測定の基本理論

ここでは、幾何的な方法をつかって 2 点の LED 座標から WiiRemote の 3 次元奥行きつ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バ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。

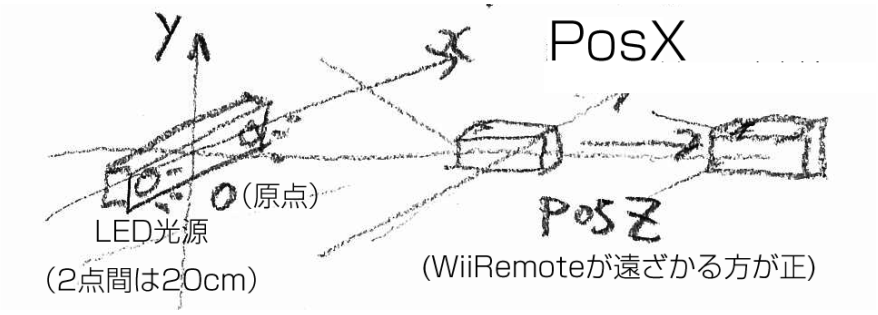


図 31: ここでの座標系、LED と WiiRemote の位置関係 (側面図)

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。

$$IrZ = | IrX1 - IrX2 | \quad ...(9\text{-}1)$$

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、遠くに行けば遠くにいくほど 2 つの LED の差は小さくなります。WiiRemote とセンサーバーの距離 (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、

$$PosZ = K * IrZ \quad ...(9\text{-}2)$$

という関係がつくれる可能性があります。もしこの K が簡単に求まるなら、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今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ことにいたしましょう。

0.9.2　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しょう。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。

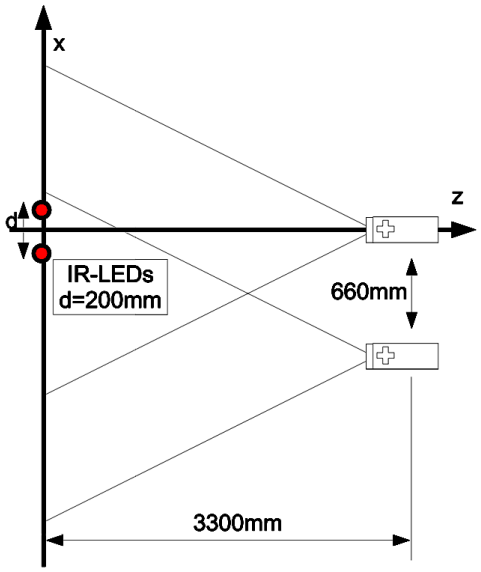


図 32: WiiRemote 赤外線特性測定の配置 (上面図)

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いますので、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良いでしょう。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。最初、WiiRemote と LED の距離があまりに近すぎると測定できません。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ったりすることに起因します。徐々に奥行きを広げていくと、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 等に直接入力しても良い) 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 PosZ = {330, 660, 990, 1320, 1650, 1980, 2310, 2640, 2970, 3300}(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 PosX = {0, 330, 660}(mm) の 3 種類で測定しています。全ての組み合わせで 30 通りありますので根気よく、流れをつかんで、ふたりひと組など実験すると良いでしょう。

以下の表のように実測値をまとめます。

表 1: 【測定シート】赤外線センサーの特性測定実験 [d=200mm]

[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 Ix を縦軸としてプロットすると以下のようになります。

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 (PosX = -330mm, -660mm)、

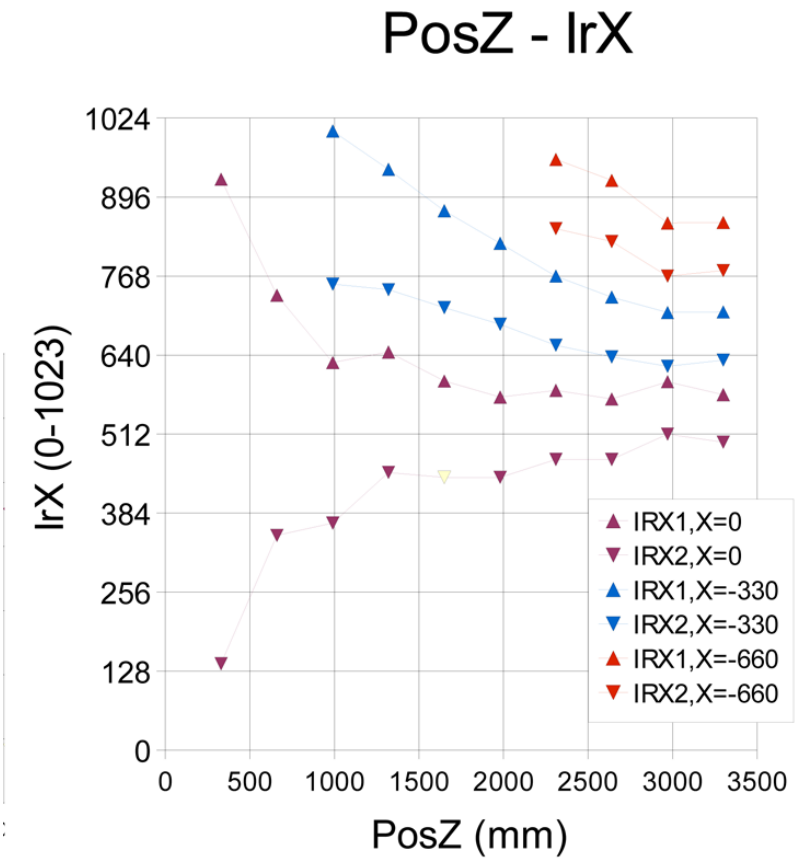


図 33: WiiRemote 赤外線特性測定の結果

左右の対象性は崩れていくようです。

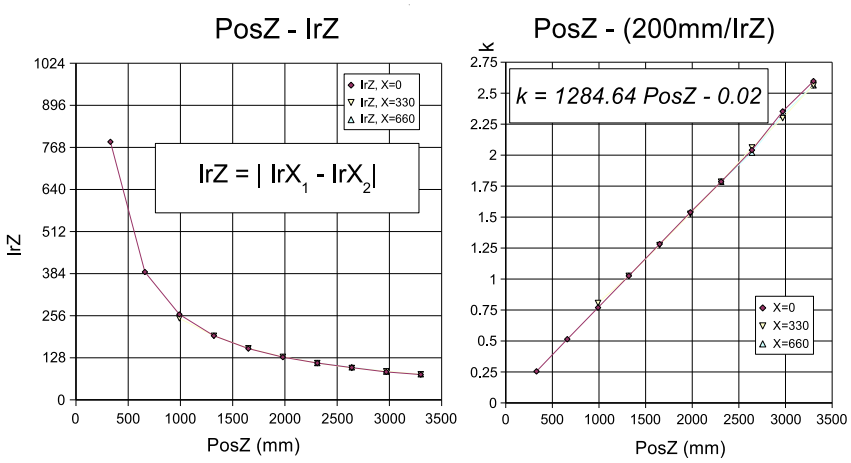


図 34: 実際の奥行き (PosZ) - 赤外線の差 (IrZ)[左]，PosX それぞれに対する K の様子 [右]

表 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 9-32(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 * PosZ - 0.02 ...(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 9-33 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。

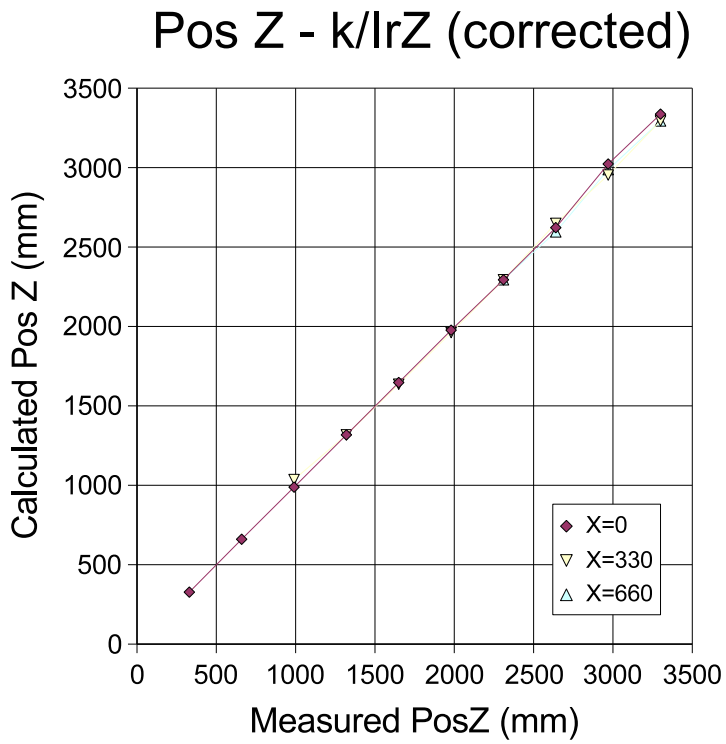


図 35: 実測 Z(横軸) – 算出 Z(縦軸)

0.9.3 【演習問題】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セクション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えますが、以下の図をヒントに考えてみると良いのではないでしょうか。

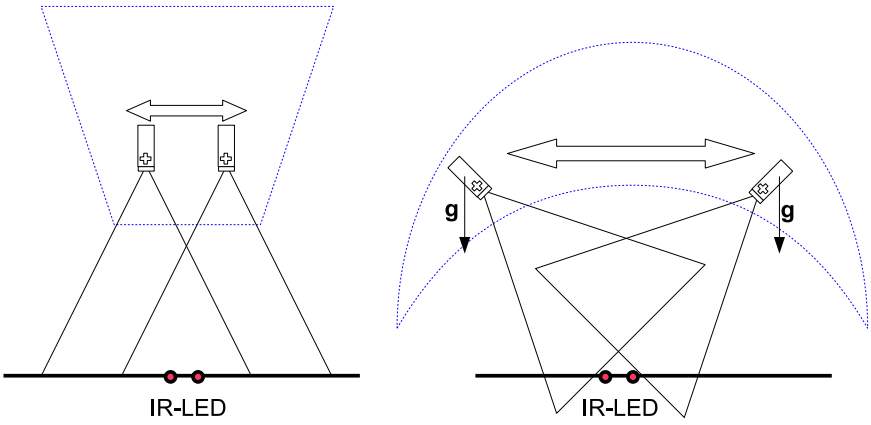


図 36: 加速度センサーを使った広範囲化

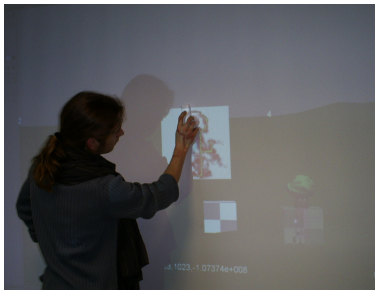


図 37: スプレー缶のような持ち方

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。

次のセクションで紹介する、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。

0.10　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, http://johnnylee.net/) 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 38: (★トリミング希望)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

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) をつかって、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校正になっています。特に位置あ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マト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 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 (例えば、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おります。

現在、彼はマイクロソフト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。

ジョニー・リー氏のブログ「procrastineering」

[URL] http://procrastineering.blogspot.com/

■ Microsoft Project Natal(YouTube 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)。

0.10.1 【演習問題】研究しよう

【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面白いです。関連する面白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良いでしょう。

論文等検索サービス

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
ソース検索「Google Code」[URL] http://code.google.com/

0.11　プログラミング編 (2)：自分で API をつくる

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。特に用事がない方は読み飛ばしていただいてかまいませんが、他の言語などに WiiRemote の API を移植するときなど、役に立つかもしれません。

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/
上記、Google Code で公開されている、DDK と C++をつかった機能最小限の自作 API です。
筑駒パ研「電脳 2007：Wii リモコンをもう一回見直してみます」by Iketaki
[URL] 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。

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

リスト 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);
         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、上のコードのほかに、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。

0.11.1 【演習問題】

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。様々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。いま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皆さんが、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おります。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る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

[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。