1. 目的

任天堂Wii、WiiUの標準的ゲームコントローラであるWiiリモコン（以下WiiRemote）を用いて、オリジナルのヒューマンインタラクション作品が開発できるよう、基盤となる知識、C#.NETを用いたプログラミング、発想の具現化と表現、工学系レポート執筆の基礎を学ぶ。

1. 背景

様々なコンテンツやソフトウェア、ハードウェアが開発、運用されている現在、それらを扱うためのヒューマンインターフェースも共に進化し続けていくはずである。今後の工学や産業の発展には欠かせないものとなることが考えられるヒューマンインターフェースについて学ぶため、基盤となる知識、C#.NETを用いたプログラミング、発想の具現化と表現、工学系レポートの執筆の基礎を学ぶ。

2-1 実世界指向インターフェースとは

実世界指向インターフェースは「マウスやキーボードなどを使わず、手にした物や端末などを移動させたり、日常的な動作をしたりすることによって、コンピューターの存在を意識せずにコンピューターを操作したり、何らかのサービスを受けたりするインターフェース」である[1]。

1. 理論

3-1 WiiRemote(Wiiリモコン)とは

「Wiiリモコンは、片手でテレビ画面に向けるだけのカンタン操作。さらに振ったり、ひねったり、指したり…と、直感的にプレイすることができるので、ニンテンドーDSのタッチスクリーンと同様に、年齢・性別・ゲーム経験の有無を問わず、だれもが自由自在に操作できます。そして、これまでに見た事もない新しいプレイスタイルは、隣にいる人を“プレイの輪”の中へ誘い込みます。」（任天堂ホームページより抜粋）[2]

つまり、今までコントローラーのようにボタンを押すだけでなくコントローラー自体を動かすという遊び方ができるということである。言い換えれば体を使って遊ぶことが可能ということで、手を動かすことでゲーム内の何かを動かすことができる直感的なものになったのである。

3-2 加速度センサ

　加速度センサとは、加速度の測定を目的とした装置である。加速度を測定し、適切な信号処理を行うことによって、傾きや、動き、振動や衝撃といった様々な情報が得られる。

この加速度センサの本格的な普及は自動車のエアバッグから始まり、物体の前後左右の動作と加速度を感知するもので、「2次元加速度センサ」と呼ばれる。

　しかし、2次元では検出範囲が限定されるため、前後左右に上下方向を加えた「3次元加速度センサ」が現在の主流である。WiiRemoteのように人間の動きに合わせた入力ができるようになったのも、この3次元加速度センサのおかげであるといえる[3]。

　WiiRemoteに使用されている加速度センサの方式は「静電容量方式」で、これはセンサ内部の稼動部分が動くことで発生した電流を測定する仕組みである。別の方式として「ピエゾ抵抗方式」があり、これは半導体に生じたゆがみにより変化する抵抗率を測定するものである[4][5]。

3-3 インターフェースとアフォーダンス

インターフェースには「アフォーダンス」という概念と深く関わっている。「アフォーダンス」とはアメリカの知覚心理学者のジェームズ・ギブソンの造語で、簡単に言うと「その環境で取りえるすべての行動」である。その内で、知覚できるものを重視した「知覚可能なアフォーダンス」という概念を提唱したのがアメリカの認知科学者ドナルド・ノーマンで、その物が何であり、どう使えば何ができるのかを見ただけで判断できるようにしなければならない、と論じた[6]。

考えてみれば当たり前であるが、機材やソフトウェアなどを用意してもインターフェースが何を動かすものなのか、どう扱うものなのかがわからなければ意味がない。WiiRemoteはその形状からテレビのリモコンのように持つということが説明書を読まなくてもわかるデザインになっている。

インターフェースについて考える際、「アフォーダンス」は非常に重要な概念であり、これを無視することはできない。

1. 方法

演習は教科書[1]p.34~39を参考にしてBluetoothペアリングを行ったWiiRemoteを使用する。使用した機材、ソフトウェアを以下に示す。

・機材セットID, No.31

Nintendo RVL-CNT-01 (デバイスアドレス, D8:6B:F7:E7:A1:AA)

・使用したBluetooth Stack

Bluetooth Stack for Windows by Toshiba Version v7.00.0 Version v7.00.05

Bluetooth Specification Version 2.1, 2.0, 1.2, 1.1

・使用したソフトウェア

おなかすいた族が公開する「WiinRemote」[2], Brian Peek氏がCodePlexにて公開している「WiimoteLib1.7」[3]及び、Microsoft Visual Studio 2010

4-1　演習1：WiinRemoteによるセンサの確認

　WiinRemoteを利用して、Wiiリモコンの加速度センサの動作確認を行う。結果をレポート5-1に示す。

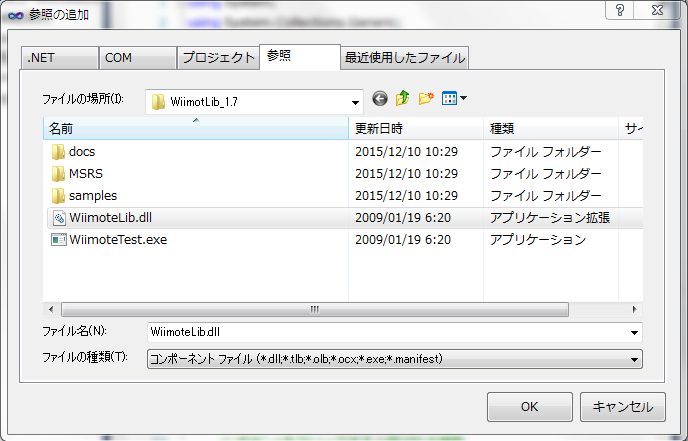
4-2　演習2：赤外線のふるまい

　WiinRemoteとセンサバー等を使用し、Wiiリモコンの赤外線センサの動作確認を行う。結果をレポート5-2に示す。

4-3　演習3：バイブレータのON/OFF

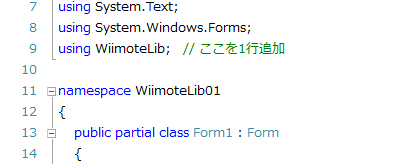
　Visual Studio 2010 でC#を利用したフォームアプリケーションを作成し、WiiRemoteのバイブレータの動作確認を行う。フォーム画面のボタンを押すことでバイブレータのON/OFF切り替えをすることで動作を確認する。以下に手順を示す。

1. Visual Studio 2010 でC#フォームアプリケーションのプロジェクトを作成する。
2. 参照設定において、WiimoteLibを追加する。
3. ソリューションエクスプローラ内の参照設定の欄を右クリック
4. 参照の追加を選択。
5. 参照の追加ウィンドウ内の参照タブを選択。
6. WiimoteLib.dllを選択し、OKボタンをクリックする(図1参照)。



【図1：WiimoteLib.dllの追加】

1. Form1.csの先頭に「using WiimoteLib;」を追加する(図2参照)。



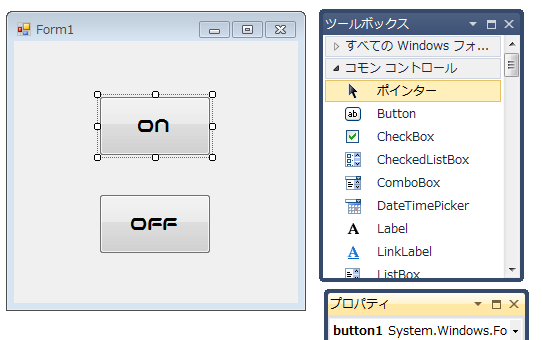
【図2：WiimoteLibの追加】

1. Wiimoteクラスのインスタンスを作成、初期化し、Wiimoteに接続するコードを追加する。ボタンが押された場合の処理を実装する。



【図3：WiimoteLibの宣言と接続、及びボタンの実装】

1. ツールボックスからボタンを配置する(図4参照)。



【図4：ボタンの配置】

1. プログラムを実行して、ボタンをクリックする。結果をレポート5-3に示す。

4-4　演習4：LEDの点灯と消灯

Visual Studio 2010 でC#を利用したフォームアプリケーションを作成し、WiiRemoteのLEDの動作確認を行う。フォーム画面のボタンを押すごとにWiiRemoteのLED表示を切り替えることで動作を確認する。以下に手順を示す。

(1) レポート4-3と同様にVisual Studio 2010 でプロジェクトを作成し、WiimoteLibの参照を追加する。Wiimoteクラスのインスタンスを作成する。

(2) 次にint型のカウント変数を宣言し、0で初期化する(図5参照)。

(3) フォーム画面にボタンを配置し、そのボタンが押されたときの処理を実装する（図5参照）。



【図5：カウント変数の宣言、ボタンの実装】

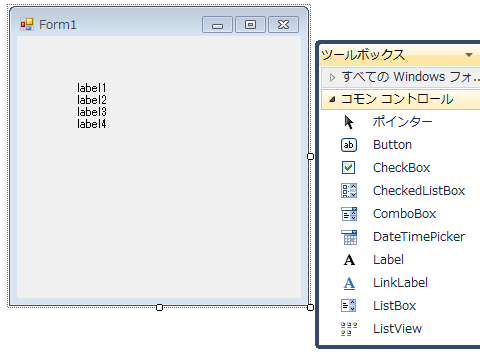
(4) プログラムを実行し、動作を確認する。結果をレポート5-4に示す。

4-5　演習5：ボタンイベントの取得

Visual Studio 2010 でC#を利用したフォームアプリケーションを作成し、WiiRemoteのボタンイベント取得の確認を行う。WiiRemoteのボタンを押すことで発生するイベントを利用し、フォーム画面のラベルを切り替えることで動作を確認する。以下に、手順を示す。

(1) レポート4-3と同様に、Visual Studio 2010でプロジェクトを作成し、WiimoteLibの参照を追加し、Wiimoteクラスのインスタンスを作成する。

(2) フォーム画面にラベルを追加する(図6参照)。



【図6：ラベルの追加】

(3) 必要なプログラムを作成する(図7参照)。



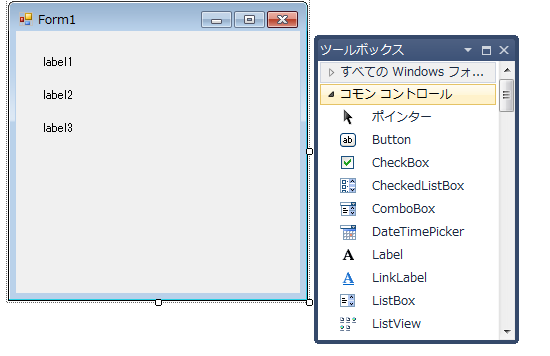
【図7：イベント関数の登録、フォーム画面描画関数の作成】

(4) プログラムを実行し動作を確認する。結果をレポート5-5に示す。

4-6　演習6：加速度センサの使用

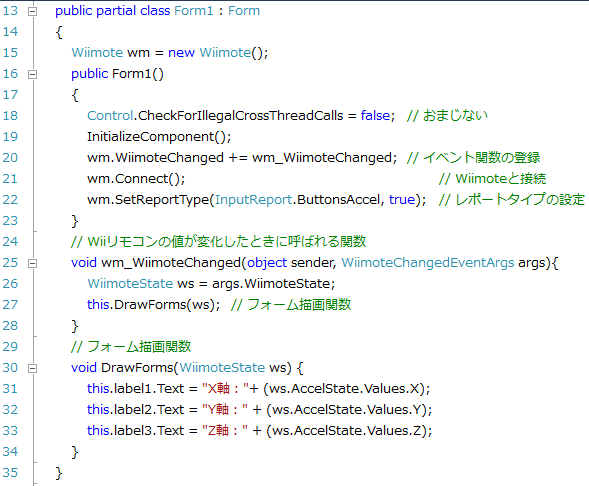
Visual Studio 2010 でC#を利用したフォームアプリケーションを作成し、WiiRemoteの加速度センサの動作確認を行う。フォーム画面にて、WiiRemoteを振ることで発生するイベントを利用し、フォーム画面のラベルでx,y,z軸それぞれの加速度センサの数値を表示させて動作を確認する。以下に、手順を示す。

1. レポート4-3と同様に、Visual Studio 2010でプロジェクトを作成し、WiimoteLibの参照を追加し、Wiimoteクラスのインスタンスを作成する。
2. フォーム画面にラベルを配置する(図8参照)。



【図8：ラベルの配置】

1. 必要なプログラムを作成する。教科書[2]と異なる部分があるので注意する。



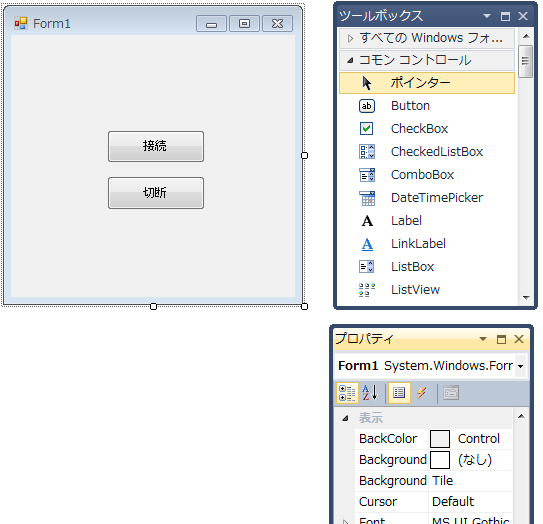
【図9：イベント関数の登録、レポートタイプの設定、フォーム描画関数の設定】

1. プログラムを実行し、動作を確認する。結果をレポート5-6に示す。

4-7　演習7：赤外線センサの使用

Visual Studio 2010 でC#を利用したフォームアプリケーションを作成し、WiiRemoteの赤外線センサの動作確認を行う。WiiRemoteが赤外線を感知した場合バイブレータを動作させることで動作を確認する。以下に、手順を示す。

1. レポート4-3と同様に、Visual Studio 2010でプロジェクトを作成し、WiimoteLibの参照を追加し、Wiimoteクラスのインスタンスを作成する。
2. フォーム画面にボタンを配置する(図10参照)。



【図10：ボタンの配置】

1. 必要なプログラムを作成する(図11参照)。



【図11：ボタンの処理と赤外線センサの処理】

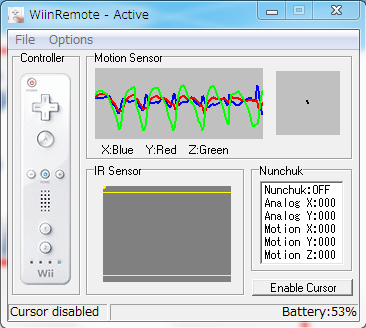
1. プログラムを実行し、動作を確認する。結果をレポート5-7に示す。
2. 結果・考察

5-1　演習1：WiinRemoteによるセンサの確認

　WiinRemoteを起動し、WiiRemoteを垂直に持って動かすと、MotionSenser欄のグラフが変化した。WiiRemoteを左右に振ると青色のグラフ、上下に平行に振ると赤色のグラフ、前後に振ると緑色のグラフが動いた(図11参照)。

　どの方向を向いても同じ動かし方で同じようにグラフが変化したので、WiiRemoteを持った人の向きに関係なく、WiiRemoteの向きを基準にしたX,Y,Z軸に対する加速度を検知していることがわかった。

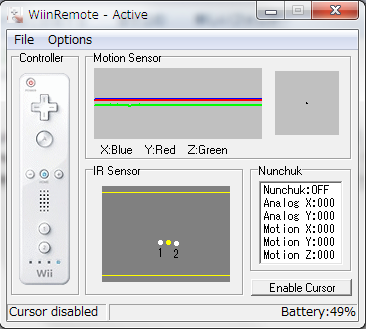
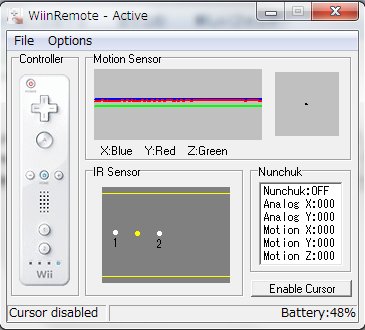
人間は機械ではないので、軸に対して完全に平行に動かすことは非常に難しいため、加速度センサを利用したアプリケーションを作成する際には注意する必要があると考えられる。



【図11：WiiRemoteを前後に振ったときのWiinRemoteの画面】

5-2　演習2：赤外線のふるまい

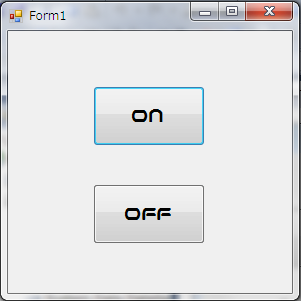
教室に設置されたセンサバー同等品に対してWiiRemoteを近づけたり離したりするとWiinRemoteのIR Sensorの欄の表示が変化した。図12は離れた位置(200~300cm)からの場合、図13は近い位置(30~50cm)からの場合のWiinRemoteの画面である。

【図12、図13：赤外線センサの確認】

5-3　演習3：バイブレータのON/OFF

　ONのボタンをクリックするとWiiRemoteのバイブレータが動作し、OFFのボタンをクリックするとWiiRemoteのバイブレータが止まった（図14参照）。ここで、OFFのボタンをクリックせずにプログラムを終了すると、WiiRemoteのバイブレータが動作したままになってしまった。もう一度プログラムを実行するとバイブレータが止まった。



【図14：バイブレータ確認プログラム実行画面】

5-4　演習4：LEDの点灯と消灯

　実行画面のボタンをクリックすることでLEDの表示が変化した(図15参照)。SetLEDs関数に渡す引数の値によってLEDの表示のさせ方を変えることができることが分かった（図16、17参照）。

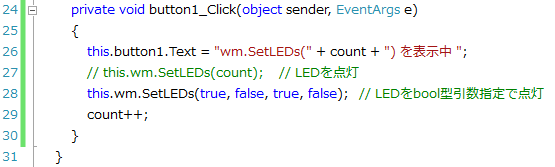


【図15：LED動作確認プログラム実行画面】

【図16，17：LEDの表示の変更】

また、SetLEDs関数に渡す引数をint型1つではなく4つのbool型にすることができた。第1引数、第2引数、第3引数、第4引数がそれぞれLED1、2、3、４に対応しており、値がtrueの場合点灯、falseの場合消灯する。引数にint型を渡す場合と比べてこちらの方がどのLEDを点灯・消灯させるかをより直感的に指定できる。(図18,19参照)。



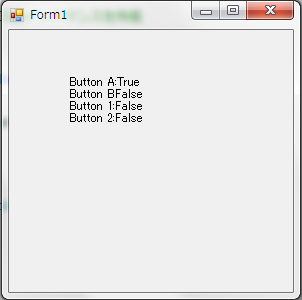
【図18：SetLEDs関数の書き換え】



【図19：LED表示の変更その2】

5-5　演習5：ボタンイベントの取得

　WiiRemoteのボタンを押すとそれぞれに対応したラベルのFalseの表示がTrueに変化した。図20は、WiiRemoteのAボタンのみを押した場合の実行結果である。

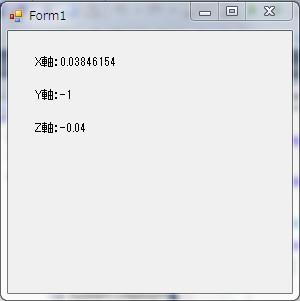
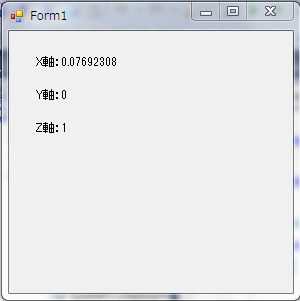


【図20：WiiRemoteボタン動作確認プログラム実行画面】

5-6　演習6：加速度センサの使用

　プログラムを実行し、WiiRemoteを振るとX,Y,Z軸それぞれに対応したラベルの数値が変化した。また、WiiRemoteを特定の向きに固定してもそれぞれの数値が変化した。

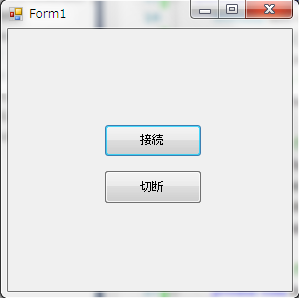
図21はWiiRemoteを垂直に立てた時の実行画面、図22はWiiRemoteのBボタンが天井を向くように置いた時の実行画面である。

【図21，図22：加速度センサ動作確認プログラム実行画面】

5-7　演習7：赤外線センサの使用

　接続ボタンを押した後、WiiRemoteを教室に設置してあるセンサバー同等品に向けて近づくとバイブレータが動作し、離れるとバイブレータが止まった。また、センサバー同等品に約10cm以下に近づけると反応しなくなった。



【図23：赤外線センサ動作確認プログラム実行画面】

1. 課題

私がWiiRemoteを利用して開発したいヒューマンインタラクションの実装アイデアは、加速度センサで重力加速度を検知して遊ぶバランスゲームである。ゲーム画面の棒が倒れないようにWiiRemoteを傾けてバランスを取る。

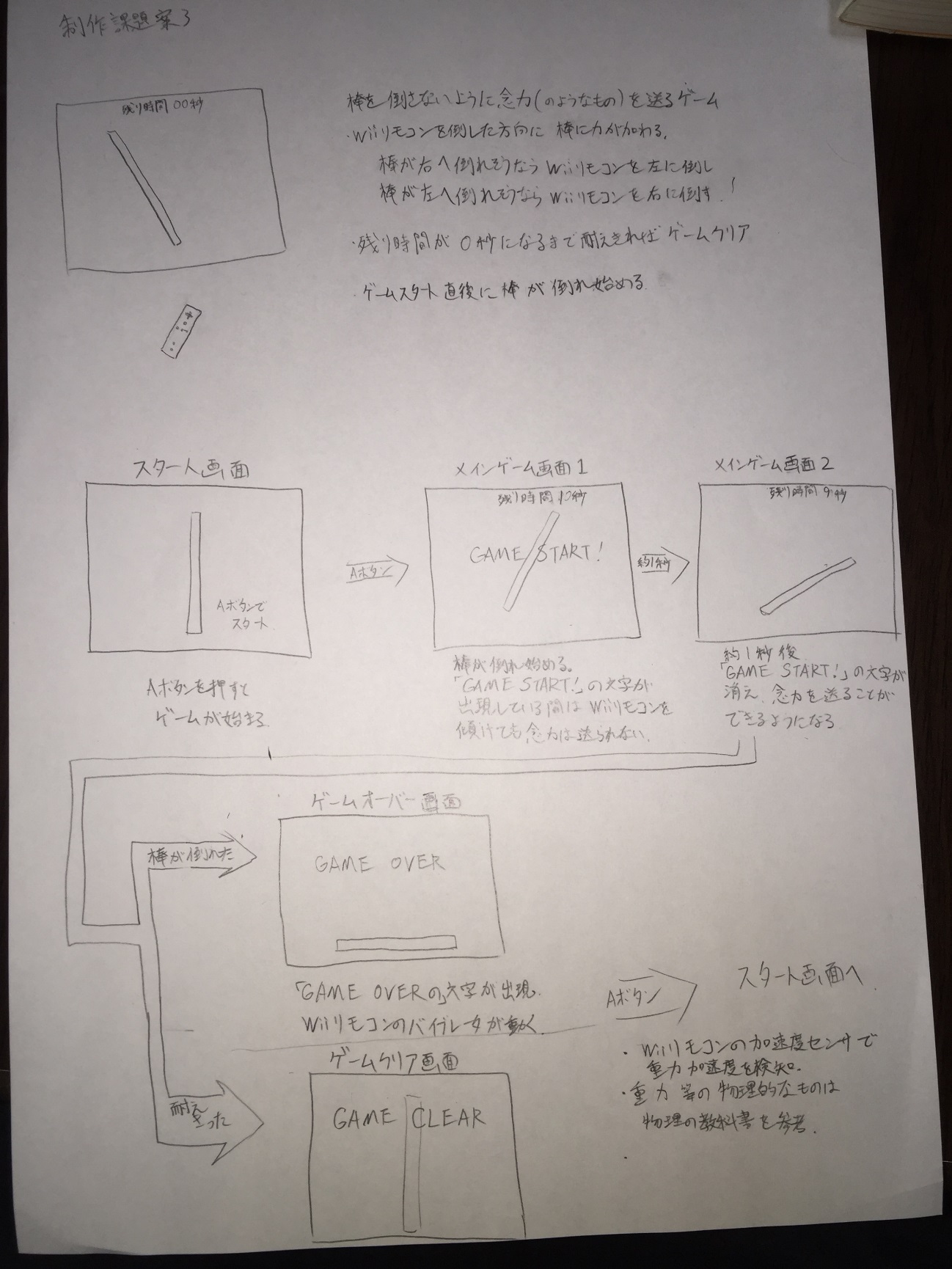
WiiRemoteの傾きを検知する方法は、WiiRemoteを横に傾けるとY軸の重力加速度は0に近づきWiiRemoteを立てるとY軸の重力加速度が-1に近づくので、これを利用してWiiRemoteをどれだけ傾けているかどうかを検知する。そしてWiiRemoteを右に傾けるとX軸の加速度が1に近づき、左に傾けるとX軸の加速度が-1に近づくので、これを利用してWiiRemoteをどちらに傾けているか検知する。

画像を回転させるにはDrawImage関数の第2引数に要素数が3つのPointF配列を渡す。これは、PointF配列の3点から平行四辺形を計算してその形に合うように画像を変形するものである[7]。ここで自分は、棒を回転させるイメージをより直感的にプログラムにしやすいよう画像を配置する中心の軸をもとにしてPointF配列を指定した。(図25プログラム176行目から192行目を参照。)

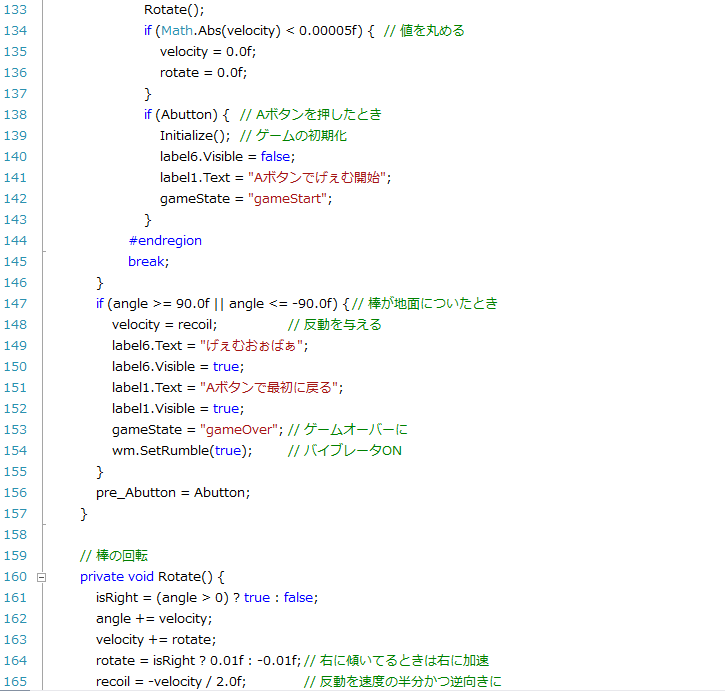
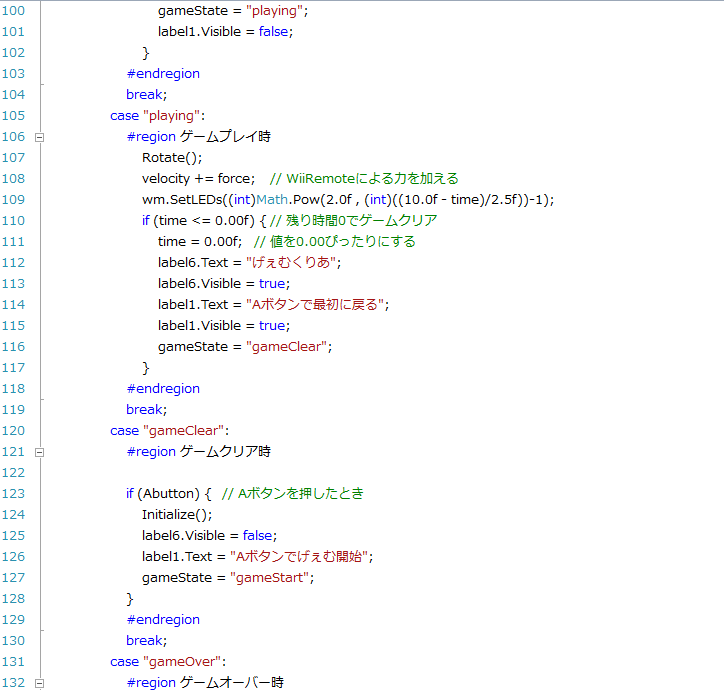
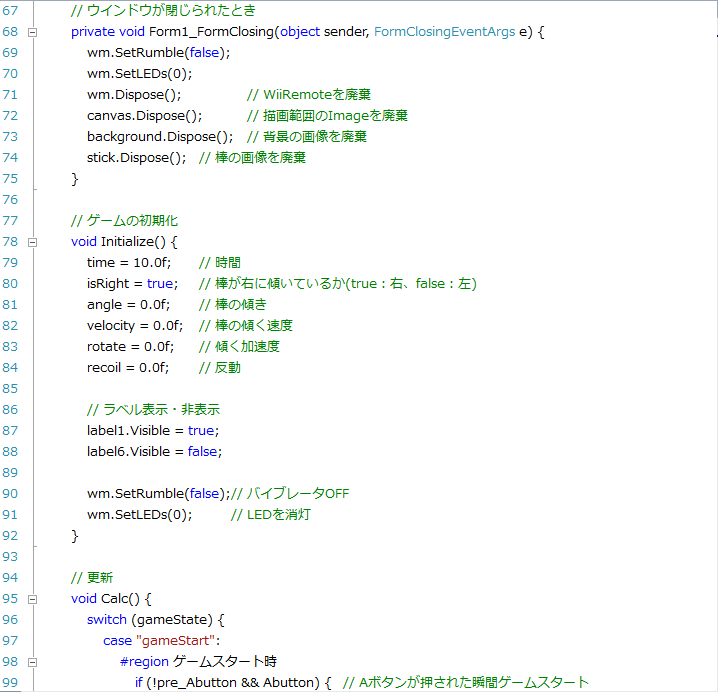
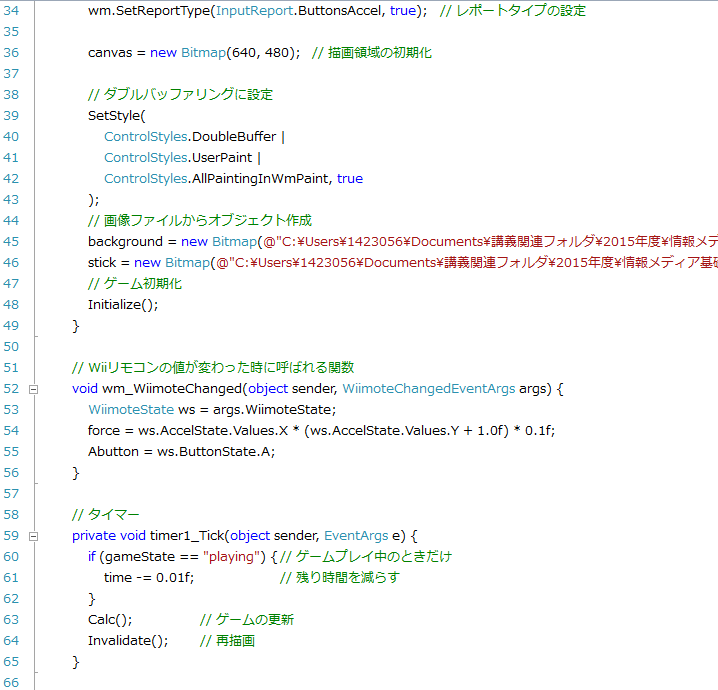
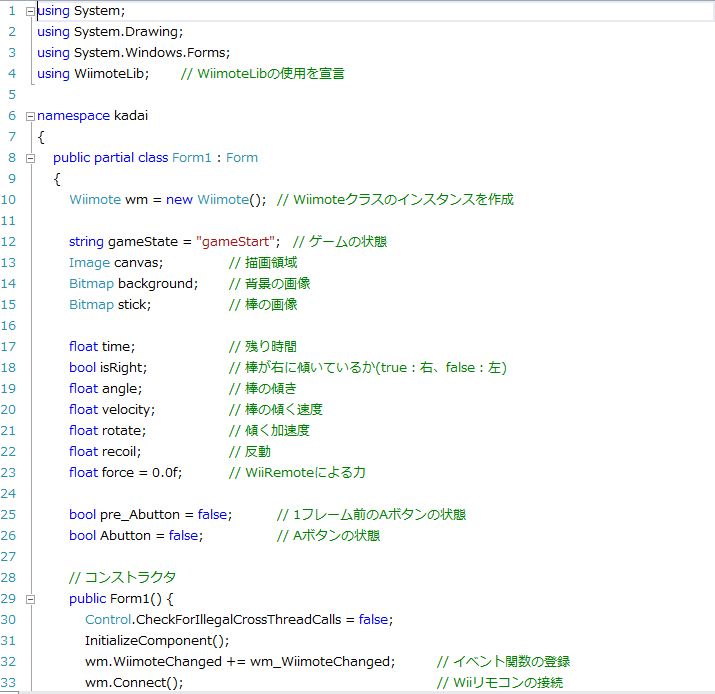
変数X、Yを回転の中心、つまり棒の下端の座標とし、変数X2、Y2を棒の上端の座標とする。この2点を結ぶ直線を画像の中心の軸と考え、回転したときの画像の左上、右上、左下の3点をPointF配列にすることで画像を回転させている。

次ページ以降にアイデアスケッチと作成したプログラムのソースコードを示す。

なお、プログラム内でラベルは4つ使用しており、番号がそれぞれ1、4、5、6であり2と3が使用されてないが、プログラム作成中に必要ないと判断して削除しラベルの番号をそのまま使用しているためである。



【図24：アイデアスケッチ】



【図25：作成したプログラムのキャプチャ画像】

1. 考察(全体で学んだこと)

今回の講義で、ユーザーに作った人の意図したとおりにインターフェイスを使用させるのは難しいということを学んだ。自分と他人の感覚や常識、経験などは全く異なるのが当たり前であり、自分が使いやすいとかわかりやすいと思っても他の人がそうであるとは限らないということが実感できた。

1. 参考文献

[1] コトバンク　実世界インターフェース

<https://kotobank.jp/word/実世界インターフェース-678152>

2015年12月20日アクセス

[2] 任天堂ホームページWii Remote <https://www.nintendo.co.jp/wii/features/wii_remote.html>

2015年12月20日アクセス

[3] ITmedia LifeStyle 「加速度センサー」　特徴と弱点、補完する角速度センサー

<http://www.itmedia.co.jp/lifestyle/articles/0905/21/news091.html>

2015年12月20日アクセス

[4] ITmedia LifeStyle 「加速度センサー」　新iPod nanoに見るトレンド

<http://www.itmedia.co.jp/lifestyle/articles/0809/11/news042.html>

2015年12月20日アクセス

[5] EDN Japan いまさら聞けない加速度センサ入門(1/2)

<http://ednjapan.com/edn/articles/1205/16/news110.html>

2015年12月20日アクセス

[6] Susan Weinxchenk 著、武舎 広幸・武舎 るみ・阿部和也 訳

「インタフェースデザインの心理学 ウェブやアプリに新たな視点をもたらす100の指針」、オーム社、2012年11月

[7] DOBONネット　画像を傾けて表示する

<http://dobon.net/vb/dotnet/graphics/skewing.html>

2015年12月23日アクセス