付録

目次

付録 A

ソースコード：サンプル

本章では付録に記載するソースコードの例を示しています.

A.1 sample.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

using WiimoteLib;

namespace Wiiremote
{
    public partial class Form1 : Form
    {
        Wiimote wm = new Wiimote();
        ButtonEvents wbe = new ButtonEvents();
        System.Drawing.Point ScreenSize;
        Boolean isConnected = false;

        float Nunchuk_Speed = 0.2f; //アナログスティックの遊び
        float Accel_Speed = 5.0f;  //加速度の判定

        public Form1()
        {
            InitializeComponent();
            Control.CheckForIllegalCrossThreadCalls = false;
            this.button2.Enabled = false;
            this.label1.Text = "リモコン未接続";
            this.label2.Text = "ヌンチャク未接続";
        }

        #region WiiRemote の状態が変化した時に呼ばれる関数
        void wm_WiimoteChanged(object sender, WiimoteChangedEventArgs args)
        {
            if (isConnected == true)
            {
                WiimoteState ws = args.WiimoteState;
                DrawForms(ws);
                IR_Cursor(ws);
                Accel_State(ws);
                wbe.Events(ws);
                Nunchuk_State(ws);
                //EffectsOut(ws);
            }
            else
            {
                this.wm.SetLEDs(0);
                this.wm.SetRumble(false);
                this.wm.Disconnect();
                this.wm.Dispose();
            }
        }
        void wm_WiimoteExtensionChanged(object sender, WiimoteExtensionChangedEventArgs args)
        {
#if DEBUG
            Console.WriteLine("nun");
#endif
            if (args.ExtensionType == ExtensionType.Nunchuk)
            {
                this.label2.Text = "ヌンチャク接続";
            }
            else if (args.ExtensionType == ExtensionType.None)
            {
                this.label2.Text = "ヌンチャク未接続";
            }
        }
        #endregion

        #region フォーム描画関数
        public void DrawForms(WiimoteState ws)
        {
            Graphics g = this.pictureBox1.CreateGraphics();
            g.Clear(Color.Black);
            if (ws.IRState.IRSensors[0].Found)
            {
                g.FillEllipse(Brushes.Red , ws.IRState.IRSensors[0].Position.X * 256, ws.IRState.IRSensors[0].Position.Y * 128, 5, 5);
                g.FillEllipse(Brushes.Blue, ws.IRState.IRSensors[1].Position.X * 256, ws.IRState.IRSensors[1].Position.Y * 128, 5, 5);
            }
            g.Dispose();
            this.label1.Text = "IR[0] " + ws.IRState.IRSensors[0].RawPosition.ToString() + "\nIR[1] " + ws.IRState.IRSensors[1].RawPosition.ToString();
        }
        #endregion

        #region 赤外線でマウスカーソル移動
        public void IR_Cursor(WiimoteState ws)
        {
            ScreenSize.X = Screen.PrimaryScreen.Bounds.Width;
            ScreenSize.Y = Screen.PrimaryScreen.Bounds.Height;

            if (ws.IRState.IRSensors[0].Found)
```

```
            {
                int px = (int)(ws.IRState.IRSensors[0].Position.X * ScreenSize.X);
                int py = (int)(ws.IRState.IRSensors[1].Position.Y * ScreenSize.Y);
                px = ScreenSize.X - px;
                System.Windows.Forms.Cursor.Position = new System.Drawing.Point(px, py);
            }
        }
        #endregion

        #region ヌンチャクの処理
        public void Nunchuk_State(WiimoteState ws)
        {
            float x = ws.NunchukState.Joystick.X;  //X 座標
            float y = ws.NunchukState.Joystick.Y;  //Y 座標
#if DEBUG
            //Console.WriteLine(x + ":" + y);
#endif
            //傾きによって矢印キーイベントの送信
            if (x > Nunchuk_Speed)
            {
                SendKeys.SendWait("{RIGHT}");
            }
            if (x < -Nunchuk_Speed)
            {
                SendKeys.SendWait("{LEFT}");
            }
            if (y > Nunchuk_Speed)
            {
                SendKeys.SendWait("{UP}");
            }
            if (y < -Nunchuk_Speed)
            {
                SendKeys.SendWait("{DOWN}");
            }
        }
        #endregion

        #region 加速度の処理
        public void Accel_State(WiimoteState ws)
        {
            float AX = Math.Abs(ws.AccelState.Values.X);    //X 軸の絶対値
            float AY = Math.Abs(ws.AccelState.Values.Y);    //Y 軸の絶対値
            float AZ = Math.Abs(ws.AccelState.Values.Z);    //Z 軸の絶対値

#if DEBUG
            //Console.WriteLine(AX+AY+AZ);
#endif
            //3 軸の絶対値の合計が AccelSpeed を超えたらキーイベント送信
            if ((AX + AY + AZ) >= Accel_Speed)
            {
                SendKeys.SendWait("{r}");
            }
        }
        #endregion

        #region LED・装飾
        public void EffectsOut(WiimoteState ws)
        {
        }
        #endregion

        #region フォームボタンの処理
        //接続
        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                if (this.isConnected == false)
                {
                    this.wm = new Wiimote();
                    this.wm.Connect();
                    this.wm.SetReportType(InputReport.IRExtensionAccel, true);
                    this.wm.SetLEDs(0);
                    this.wm.SetRumble(false);
                    this.button1.Enabled = false;
                    this.button2.Enabled = true;
                    this.wm.WiimoteChanged += wm_WiimoteChanged;
                    this.wm.WiimoteExtensionChanged += wm_WiimoteExtensionChanged;
                    this.isConnected = true;
                }
            }
            catch (WiimoteNotFoundException)
            {
                this.label1.Text = "接続できませんでした";
            }
        }
        //切断
        private void button2_Click(object sender, EventArgs e)
        {
            if (this.isConnected == true)
            {
                this.isConnected = false;
                this.wm.WiimoteChanged -= wm_WiimoteChanged;
                this.wm.WiimoteExtensionChanged -= wm_WiimoteExtensionChanged;
                this.wm.SetLEDs(0);
                this.wm.SetRumble(false);
                this.wm.Disconnect();
```

```
                this.wm.Dispose();
                this.button1.Enabled = true;
                this.button2.Enabled = false;
                this.label1.Text = "リモコン未接続";
                this.label2.Text = "ヌンチャク未接続";
            }
        }
        //終了
        private void button3_Click(object sender, EventArgs e)
        {
            if (this.isConnected == true)
            {
                this.wm.WiimoteChanged -= wm_WiimoteChanged;
                this.wm.WiimoteExtensionChanged -= wm_WiimoteExtensionChanged;
                this.wm.SetLEDs(0);
                this.wm.SetRumble(false);
                this.wm.Disconnect();
                this.wm.Dispose();
                this.isConnected = false;
                this.Close();
            }
            else
            {
                this.Close();
            }
        }
        #endregion
    }
}
```

ソースコードを Latex ファイルとして表示させています.

一行が長く，表示範囲からはみ出る場合には改行を加える等で調整してください.

付録 B

Latex について

B.1 参考

1. Latex を使用する場合には Windows 環境をお勧めします.
   環境によって文字サイズ・書式等が変わります.
2. 章や参考文献の数字は，一つ前にコンパイルしたファイルを元に表示しています.「?」表示になっていたり想定と違う番号が表示されている場合は，もう一度コンパイルすると解決します.
3. ソースコードは Latex ファイルでなくとも表示することができます．使いやすいほうを使用してください.
4. Latex にはさまざまなスタイルや書式が存在します．使用したいものがあれば各自で導入してください.