EBOOK

Michael Andre Franiatte

Wiimote/Joycon Setting with Codes in Gamepad Libraries to Play PC Games

Wiimote/Joycon Setting for Playing PC Games

EBOOK

Wiimote/Joycon Setting with Codes in Gamepad Libraries to Play PC Games

Wiimote/Joycon Setting for Playing PC Games

Michael Franiatte
06/09/2018

Wiimote/Joycon used on PC Games can be very competitive compared to keyboard and mouse and more enjoyable than joystick because it use motion recognizing and only four fingers are used to push the buttons, other fingers are used to stabilized the Wiimote/Joycon. All buttons and motion recognizing are easy of access. Information about license, EULA and contract for using these following works can be found at https://michaelfraniatte.wordpress.com.

Wiimote/Joycon Setting with Codes in Gamepad Libraries to Play PC Games

Michael Franiatte*

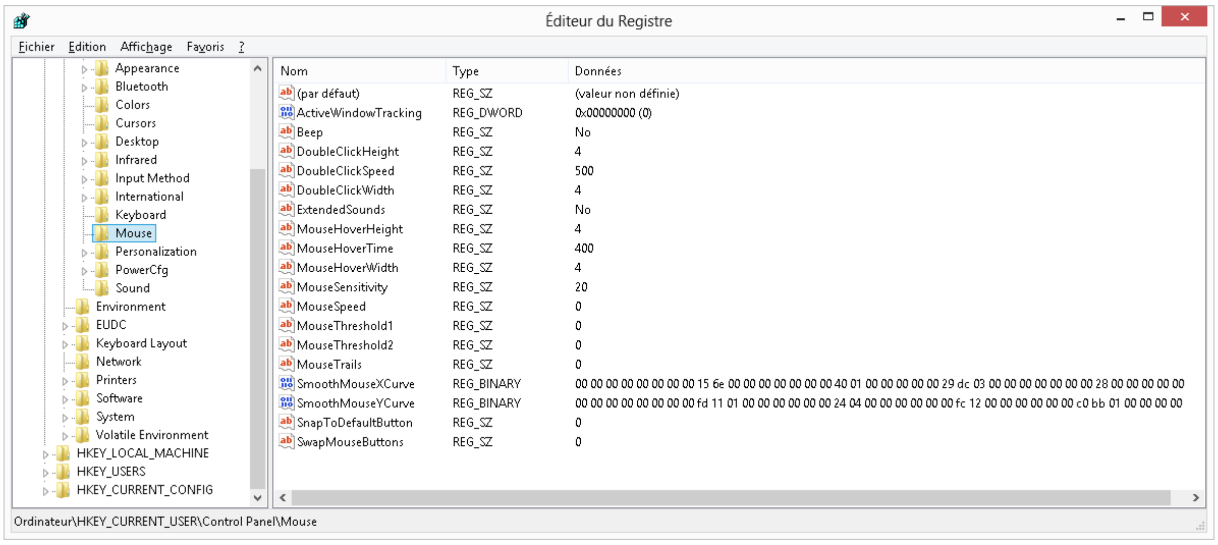
Abstract

With this book you will have easy to have a competitive controller or gamepad able to interface in all your PC (Personal Computer) games. The Wiimote/Joycon are designed to be the best controller for having the best of fun, when the program to play Wiimote/Joycon have the settings adapted to your games. The codes used to take the control of games are the best than you can find in comparison to other programs or other platforms. Since 2008, the works made on this controller have never touched such a perfection, and in comparison with other programs or controllers or platforms, such an enjoyment to play with it… because such a perfect controller with such perfect codes, on the best platform to play which is a PC, using a native program language, made the gameplay very realistic and immersive. If you don't have trouble to aim something or drive a car, Wiimote/Joycon connected with a Bluetooth key, the information presented here is made for you to enjoy a maximum to play all PC games on your favorite platform which is the PC having the best game fluidity and resolution. All other platforms with all other controllers can't have such realistic, easy and immersive gameplay.

Keywords: *Wiimote/Joycon, Nunchuck, PC, controller, play, codes*

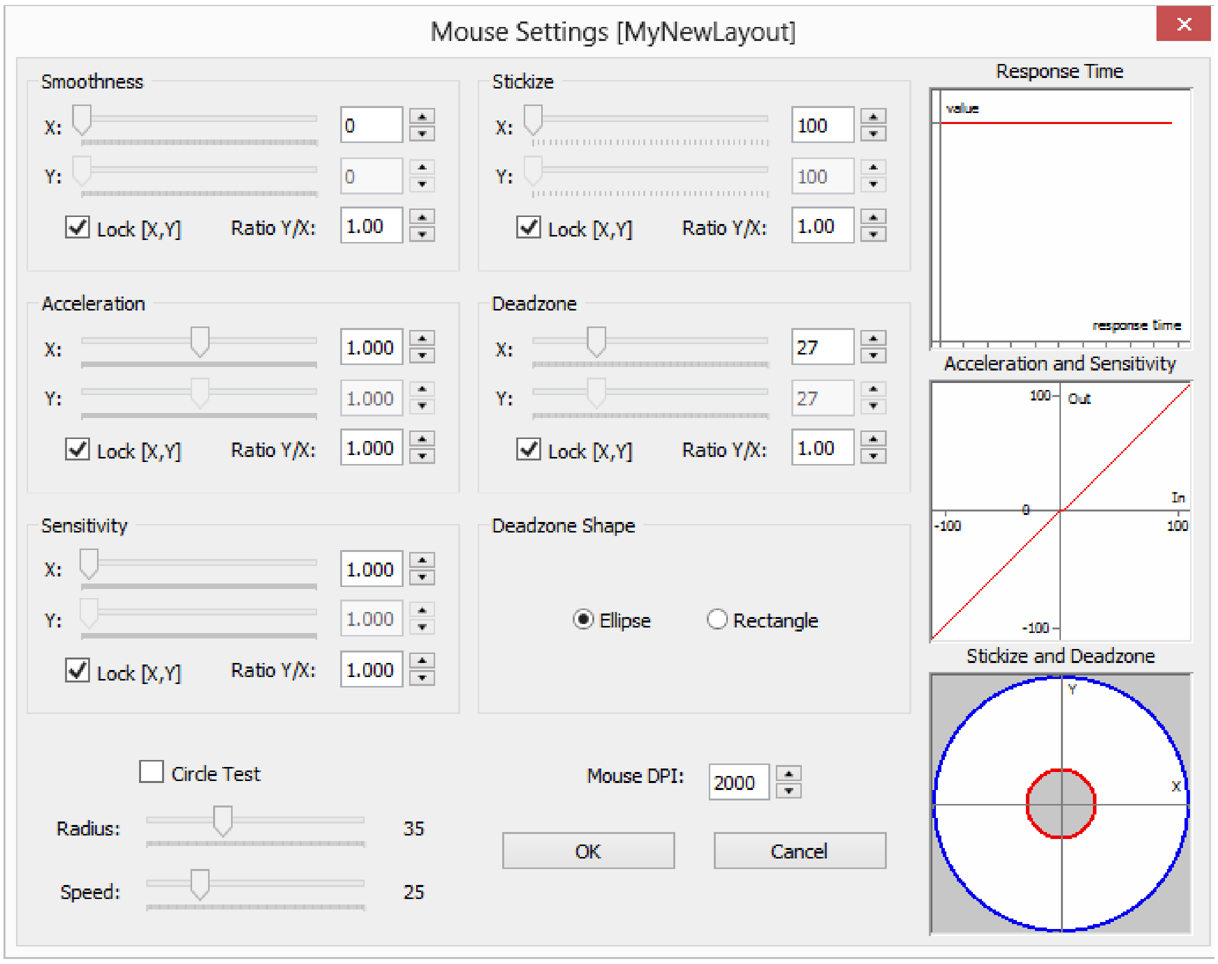
* Author correspondence: michael.franiatte@gmail.com

1. Configuration of mouse on PC for plugin X-Aim of Cronusmax Pro application



Nom	Type	Données
(par défaut)	REG_SZ	(valeur non définie)
ActiveWindowTracking	REG_DWORD	0x00000000 (0)
Beep	REG_SZ	No
DoubleClickHeight	REG_SZ	4
DoubleClickSpeed	REG_SZ	500
DoubleClickWidth	REG_SZ	4
ExtendedSounds	REG_SZ	No
MouseHoverHeight	REG_SZ	4
MouseHoverTime	REG_SZ	400
MouseHoverWidth	REG_SZ	4
MouseSensitivity	REG_SZ	20
MouseSpeed	REG_SZ	0
MouseThreshold1	REG_SZ	0
MouseThreshold2	REG_SZ	0
MouseTrails	REG_SZ	0
SmoothMouseXCurve	REG_BINARY	00 00 00 00 00 00 00 00 15 6e 00 00 00 00 00 00 00 40 01 00 00 00 00 00 29 dc 03 00 00 00 00 00 00 00 28 00 00 00 00 00
SmoothMouseYCurve	REG_BINARY	00 00 00 00 00 00 00 00 fd 11 01 00 00 00 00 00 00 24 04 00 00 00 00 00 00 fc 12 00 00 00 00 00 00 c0 bb 01 00 00 00 00
SnapToDefaultButton	REG_SZ	0
SwapMouseButtons	REG_SZ	0

Ordinateur\HKEY_CURRENT_USER\Control Panel\Mouse



Smoothness
X: 0
Y: 0
[✓] Lock [X,Y] Ratio Y/X: 1.00

Stickize
X: 100
Y: 100
[✓] Lock [X,Y] Ratio Y/X: 1.00

Acceleration
X: 1.000
Y: 1.000
[✓] Lock [X,Y] Ratio Y/X: 1.000

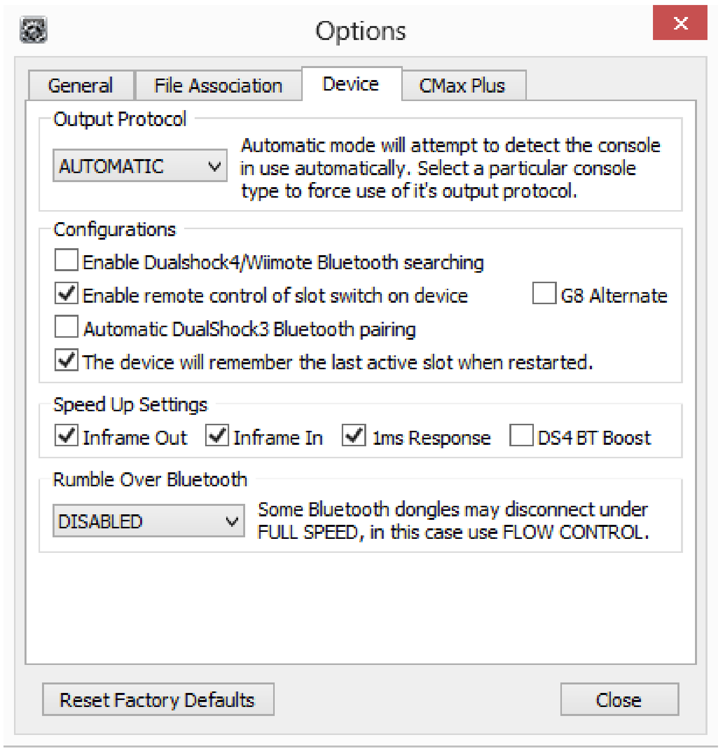
Deadzone
X: 27
Y: 27
[✓] Lock [X,Y] Ratio Y/X: 1.00

Sensitivity
X: 1.000
Y: 1.000
[✓] Lock [X,Y] Ratio Y/X: 1.000

Deadzone Shape
[✓ Ellipse] Rectangle

Circle Test
Radius: 35
Speed: 25

Mouse DPI: 2000


Options

General | File Association | Device | CMax Plus

Output Protocol

AUTOMATIC

Automatic mode will attempt to detect the console in use automatically. Select a particular console type to force use of it's output protocol.

Configurations

- [ ] Enable Dualshock4/Wiimote Bluetooth searching
- [x] Enable remote control of slot switch on device
- [ ] G8 Alternate
- [ ] Automatic DualShock3 Bluetooth pairing
- [x] The device will remember the last active slot when restarted.

Speed Up Settings

- [x] Inframe Out
- [x] Inframe In
- [x] 1ms Response
- [ ] DS4 BT Boost

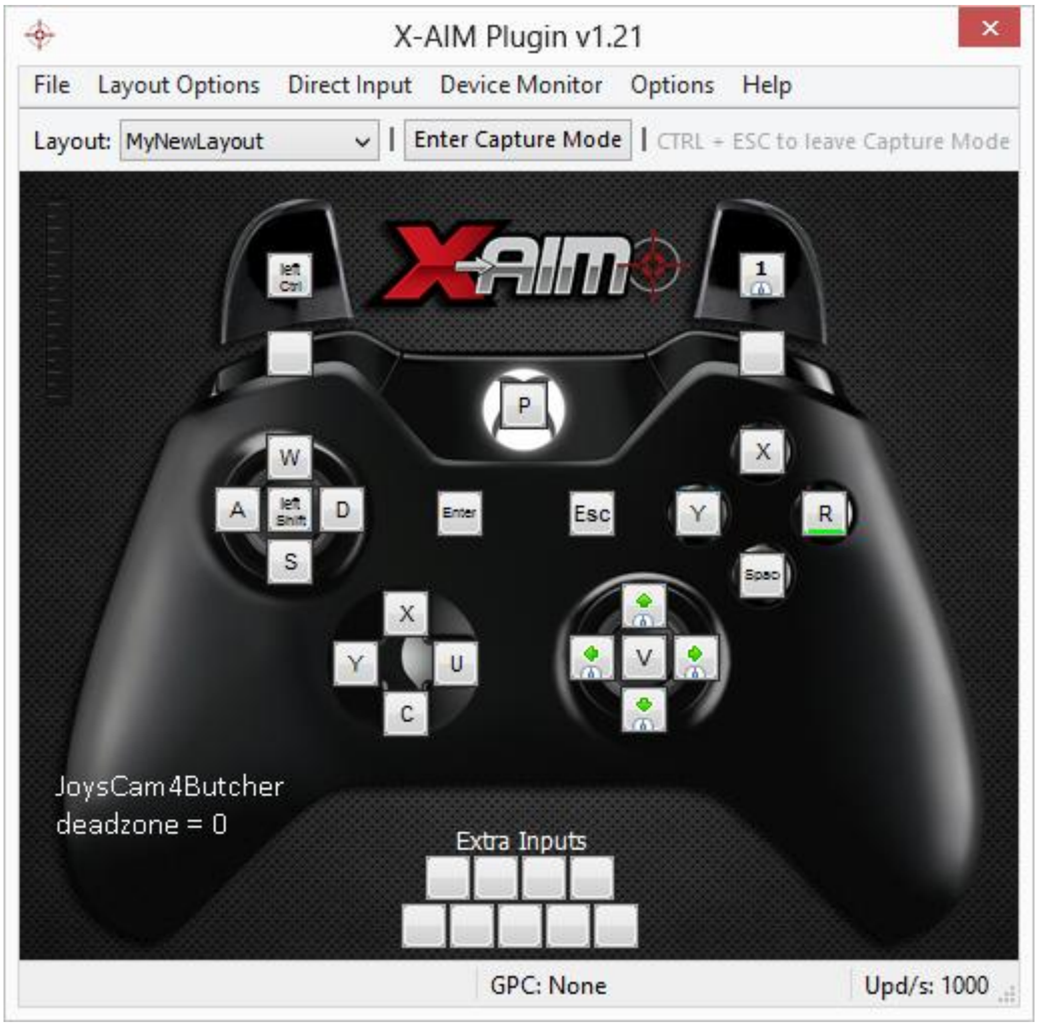
Rumble Over Bluetooth

DISABLED

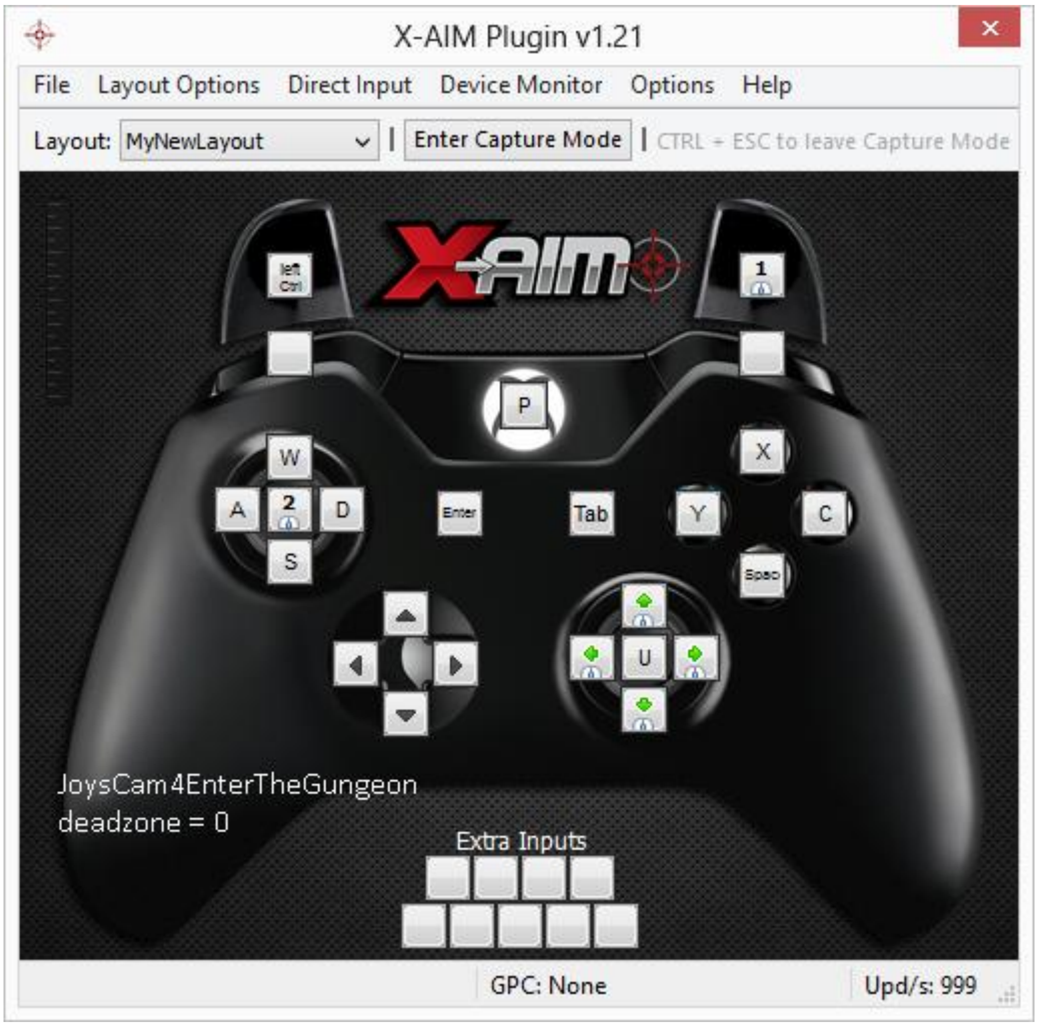
Some Bluetooth dongles may disconnect under FULL SPEED, in this case use FLOW CONTROL.

Reset Factory Defaults | Close

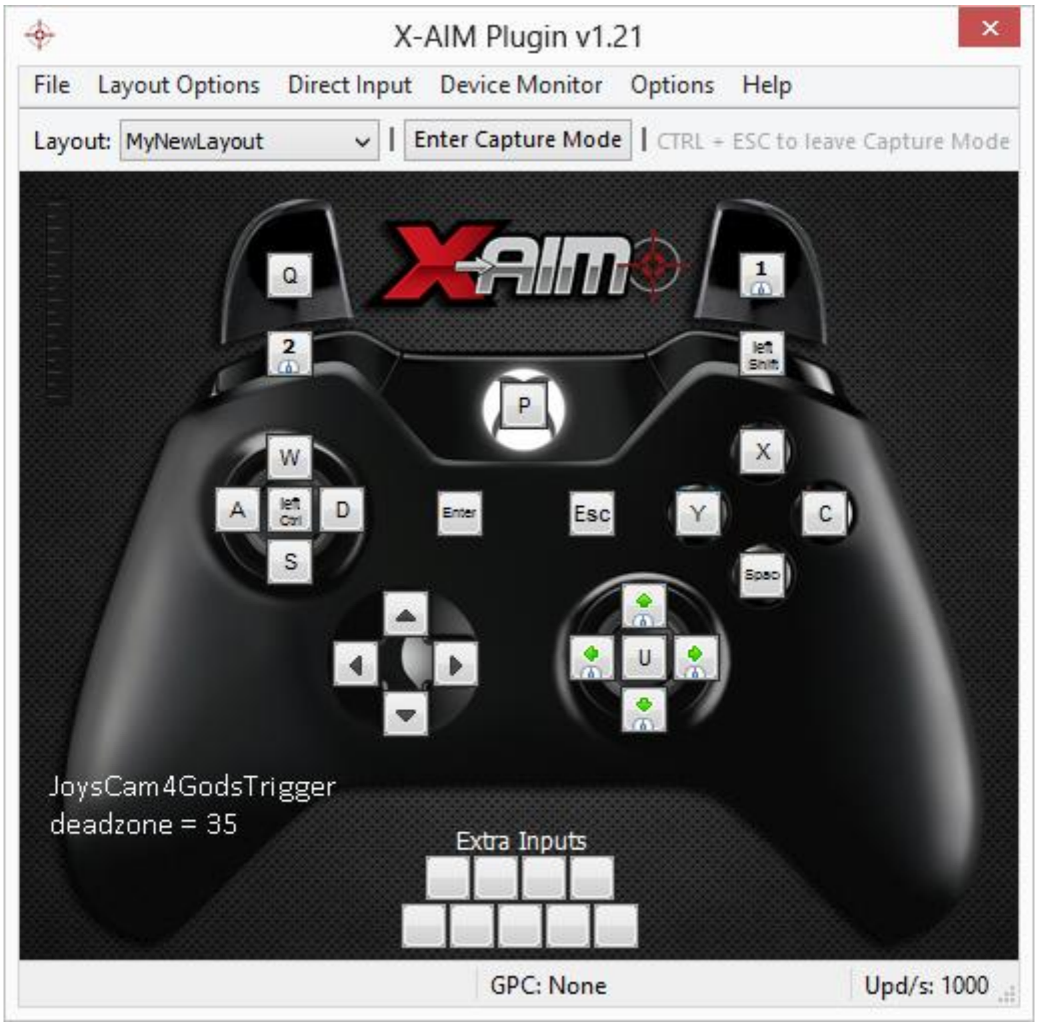
2. Configuration of plugin X-Aim of Cronusmax Pro application for games



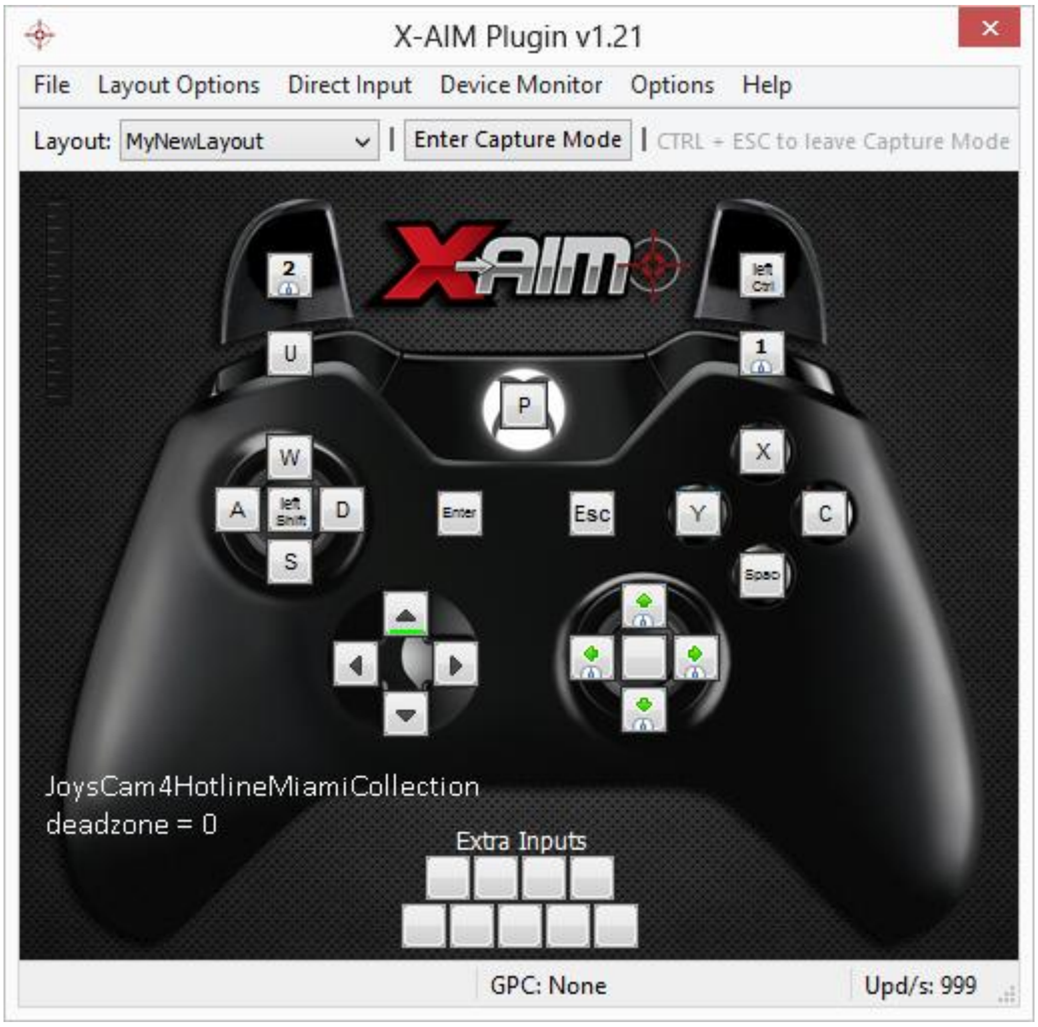
JoysCam4Butcher
deadzone = 0



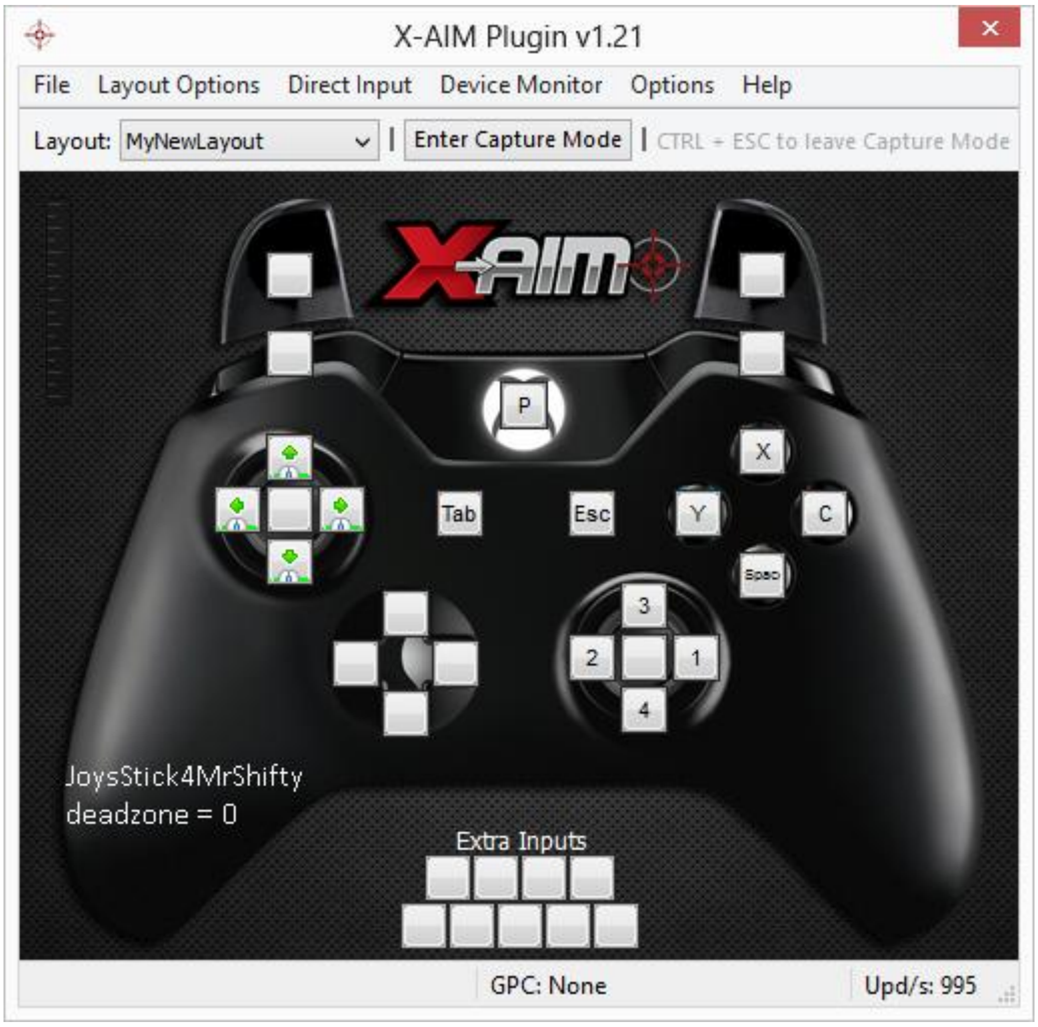
JoysCam4EnterTheGungeon
deadzone = 0



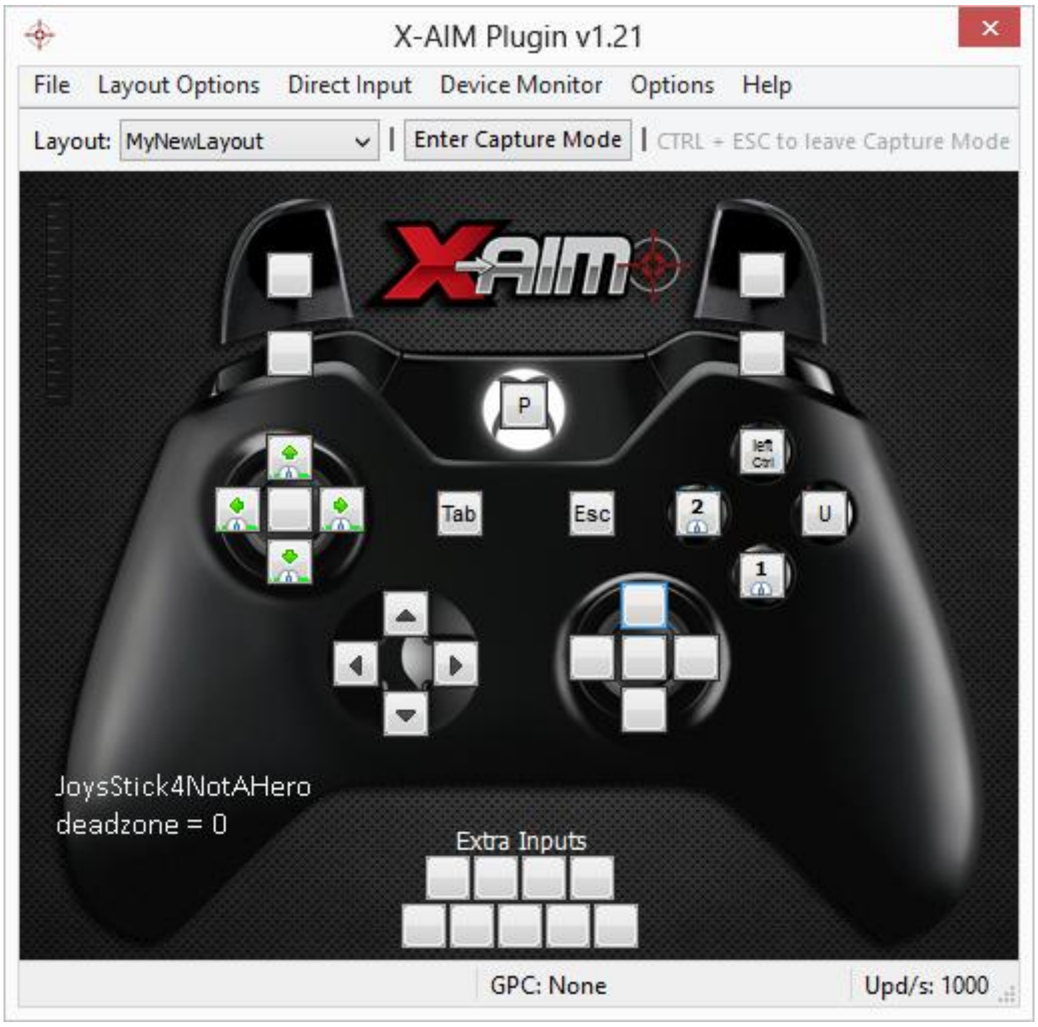
JoysCam4GodsTrigger
deadzone = 35

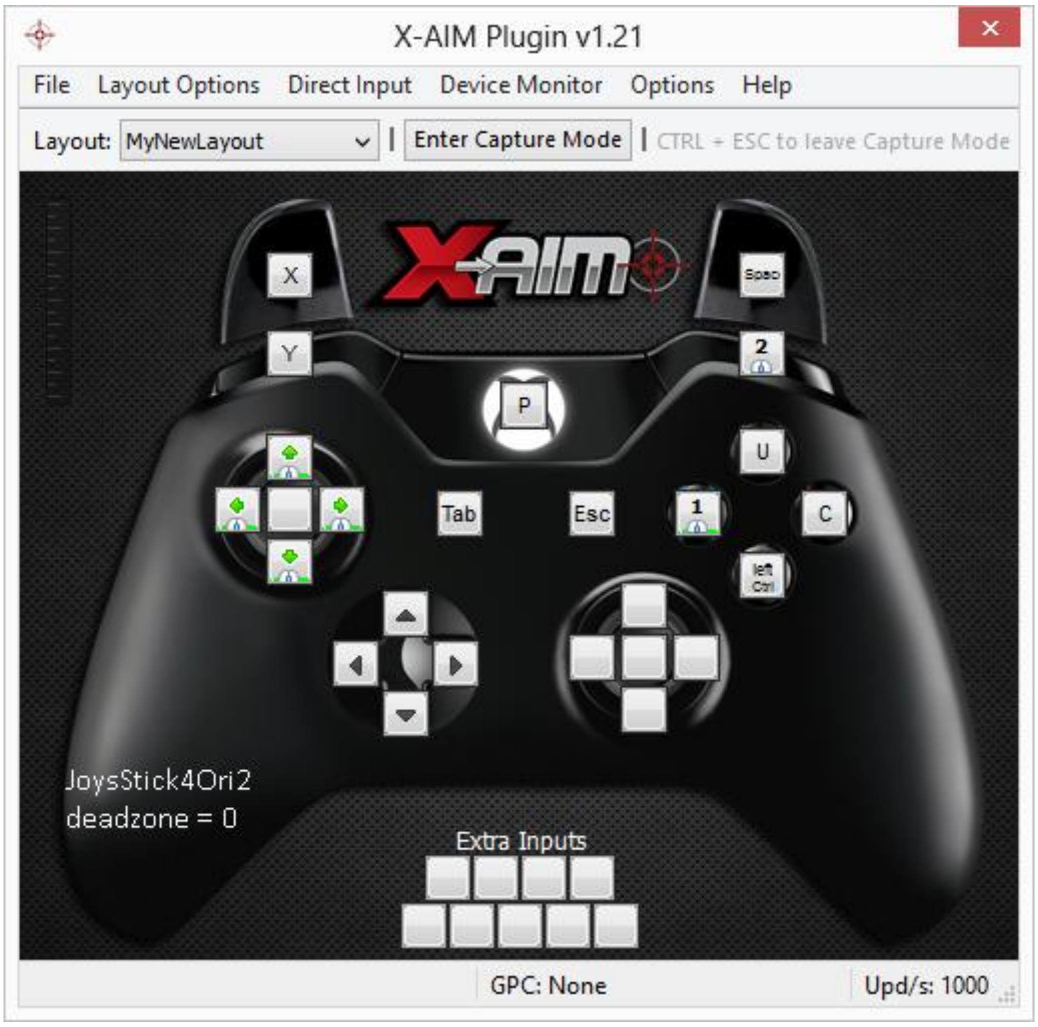


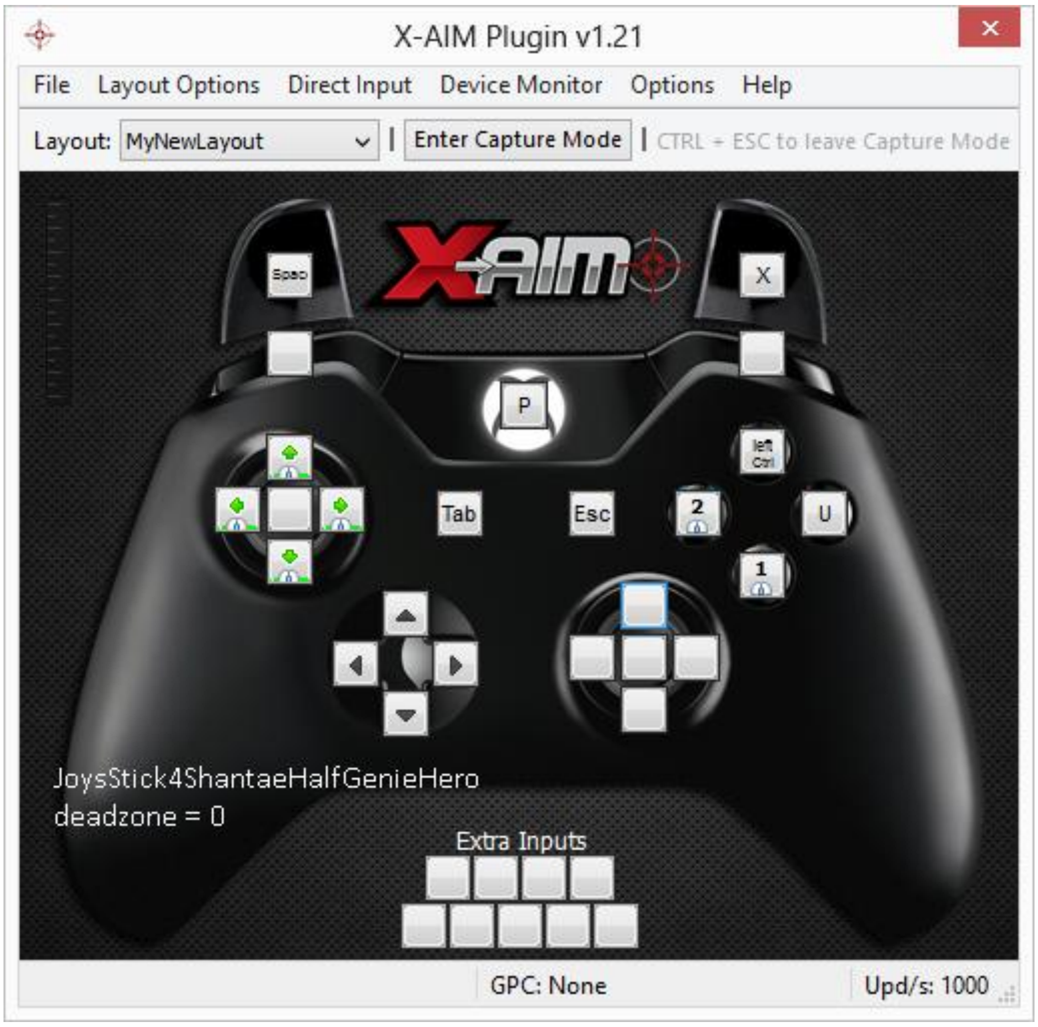
JoysCam4HotlineMiamiCollection
deadzone = 0

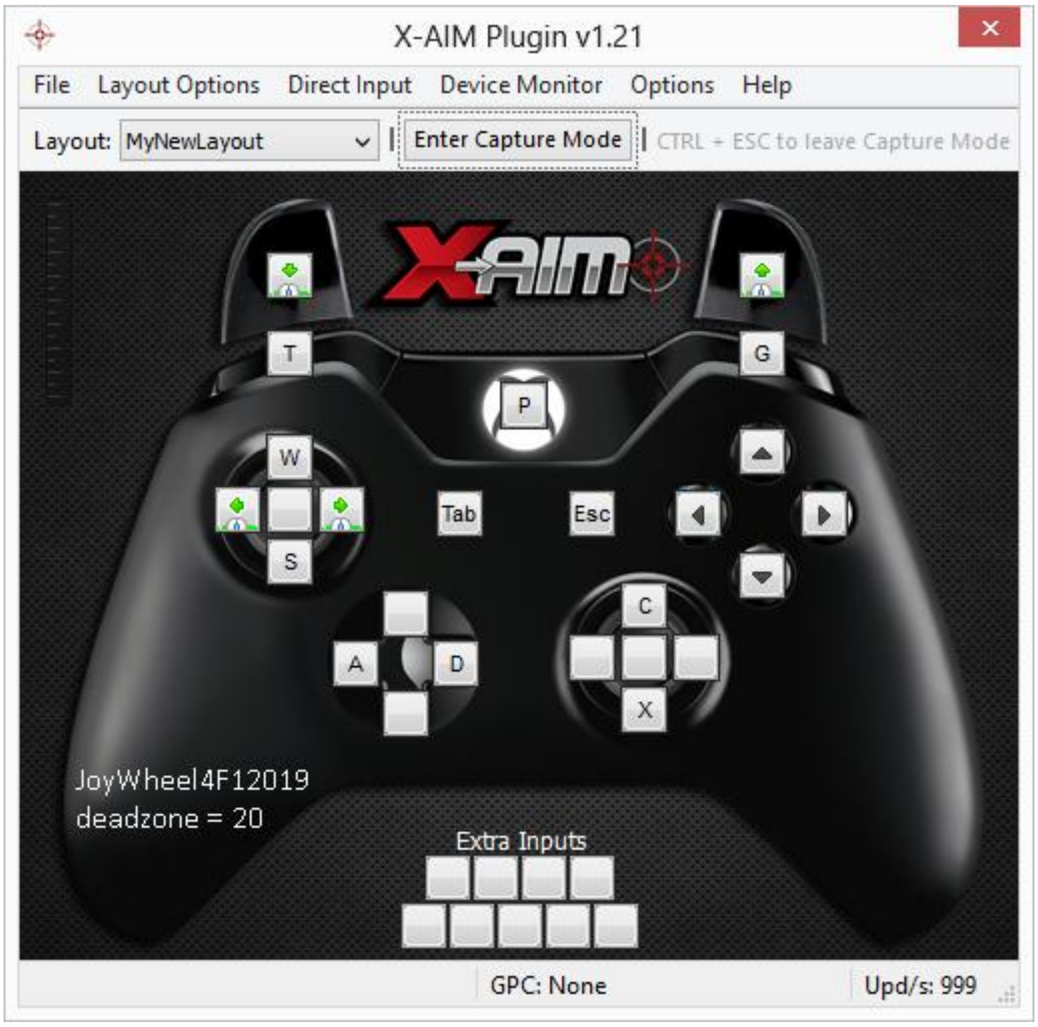


JoysStick4MrShifty
deadzone = 0

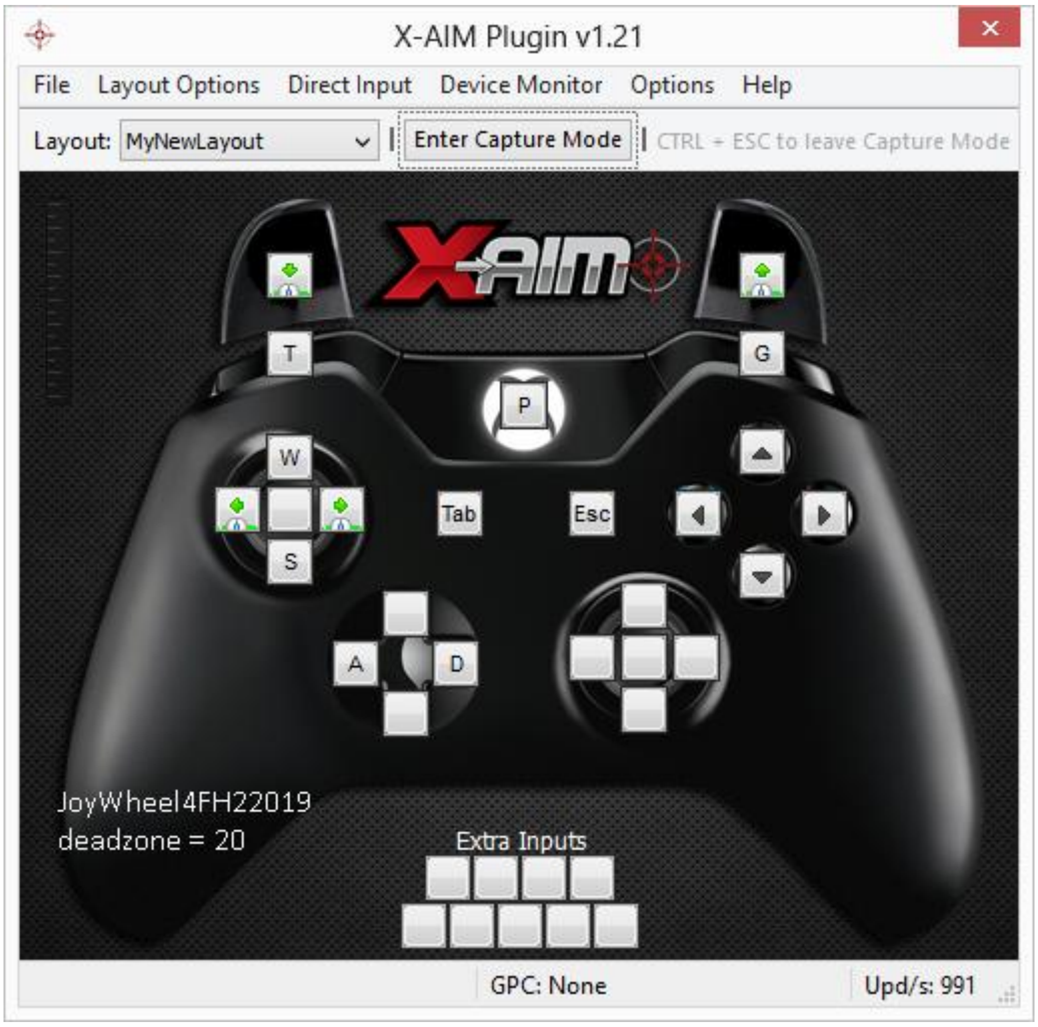

X-AIM Plugin v1.21
File
Layout Options
Direct Input
Device Monitor
Options
Help
Layout: MyNewLayout
Enter Capture Mode
CTRL + ESC to leave Capture Mode
X-AIM
P
left Ctrl
Tab
Esc
2
U
1
JoysStick4NotAHero
deadzone = 0
Extra Inputs
GPC: None
Upd/s: 1000


X-AIM Plugin v1.21
File
Layout Options
Direct Input
Device Monitor
Options
Help
Layout: MyNewLayout
Enter Capture Mode
CTRL + ESC to leave Capture Mode
X
Y
Spac
2
P
U
Tab
Esc
1
C
left Ctrl
JoysStick4Ori2
deadzone = 0
Extra Inputs
GPC: None
Upd/s: 1000

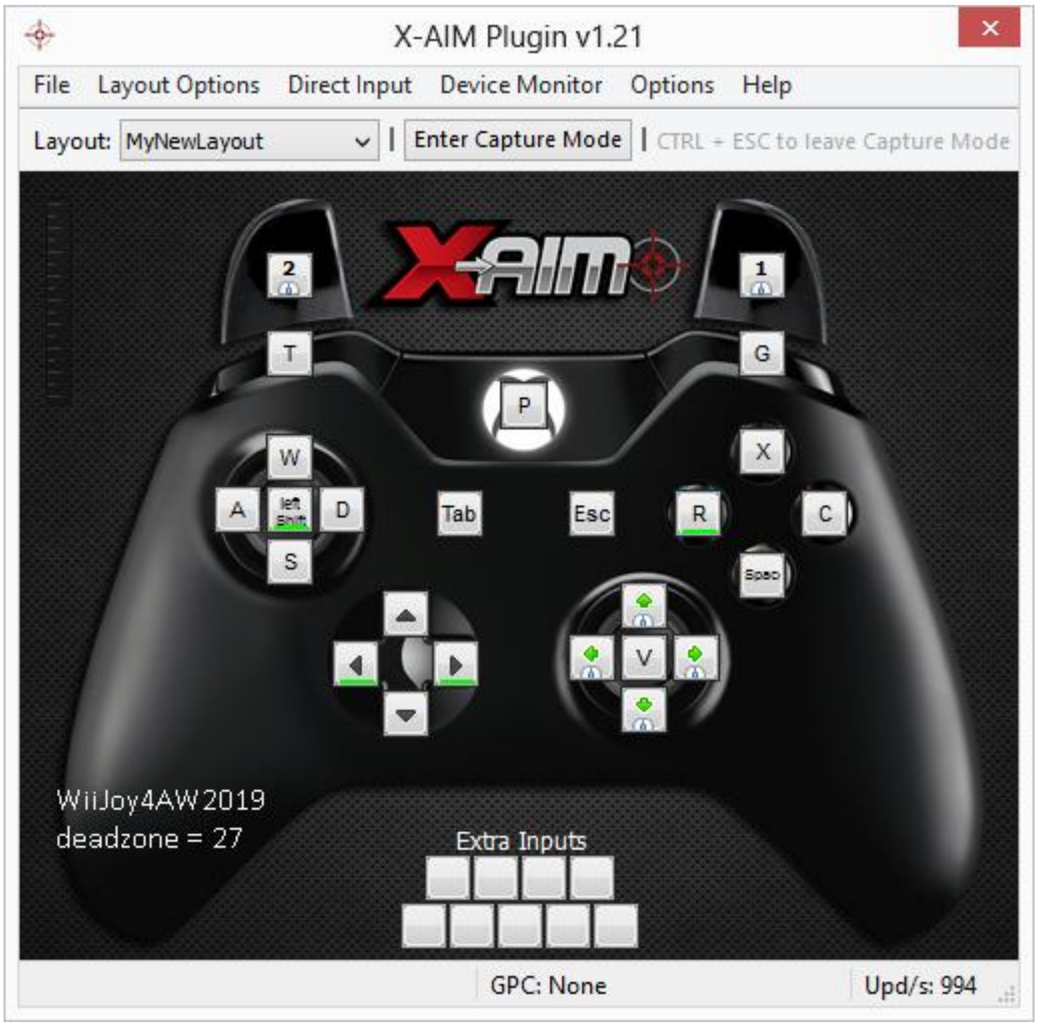

X-AIM Plugin v1.21
File
Layout Options
Direct Input
Device Monitor
Options
Help
Layout: MyNewLayout
Enter Capture Mode
CTRL + ESC to leave Capture Mode
Spac
X
P
left Ctrl
Tab
Esc
2
U
1
JoysStick4ShantaeHalfGenieHero
deadzone = 0
Extra Inputs
GPC: None
Upd/s: 1000



JoyWheel4F12019
deadzone = 20



JoyWheel4FH22019
deadzone = 20

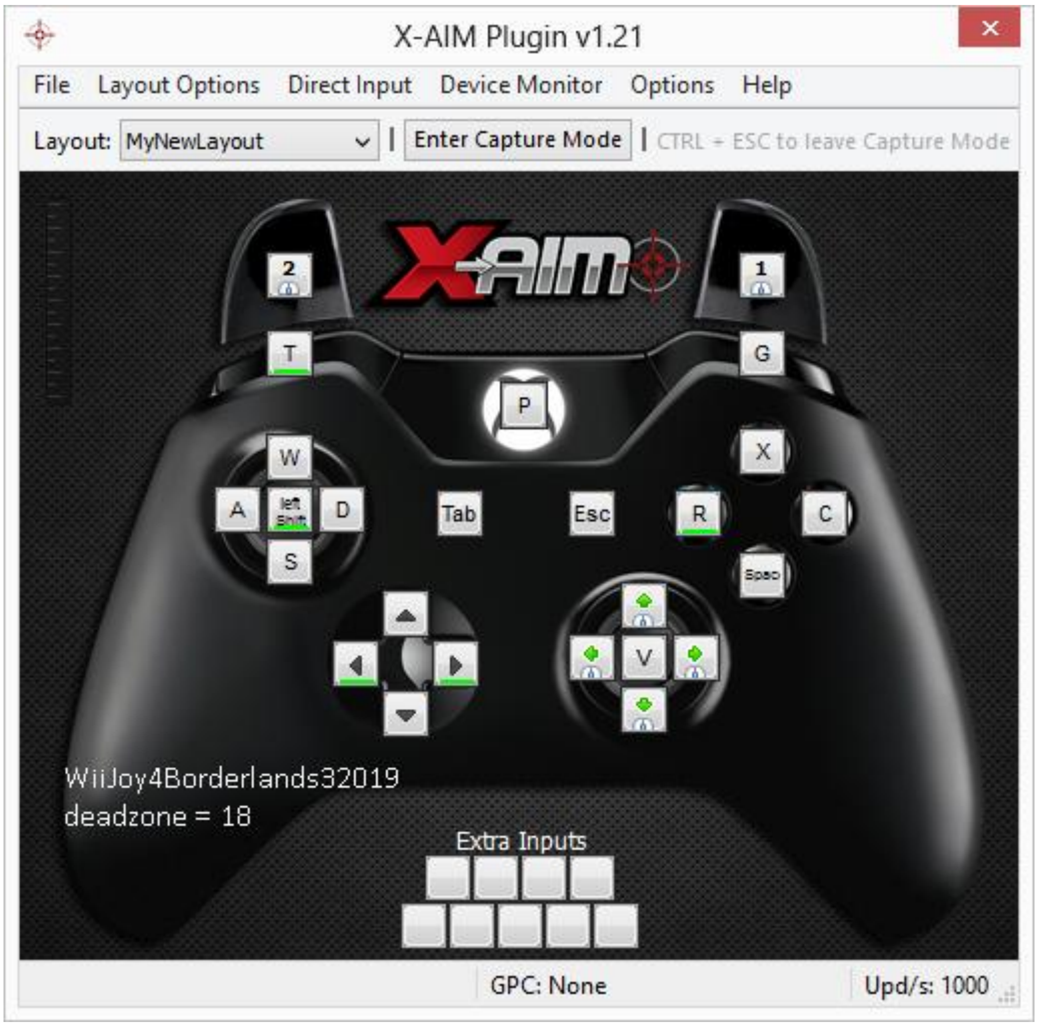


Layout: MyNewLayout

WiiJoy4AW2019
deadzone = 27

GPC: None

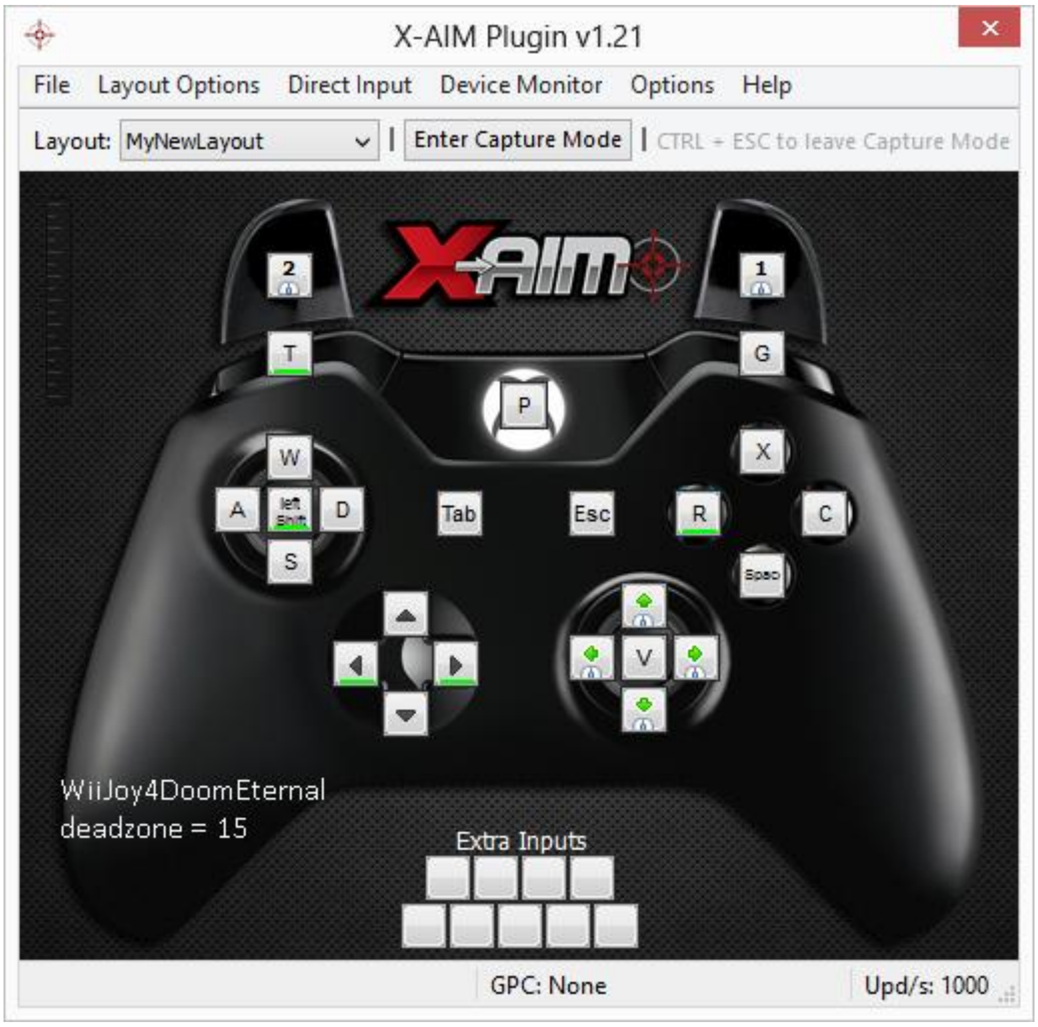
Upd/s: 994



Layout: MyNewLayout

WiiJoy4Borderlands32019
deadzone = 18

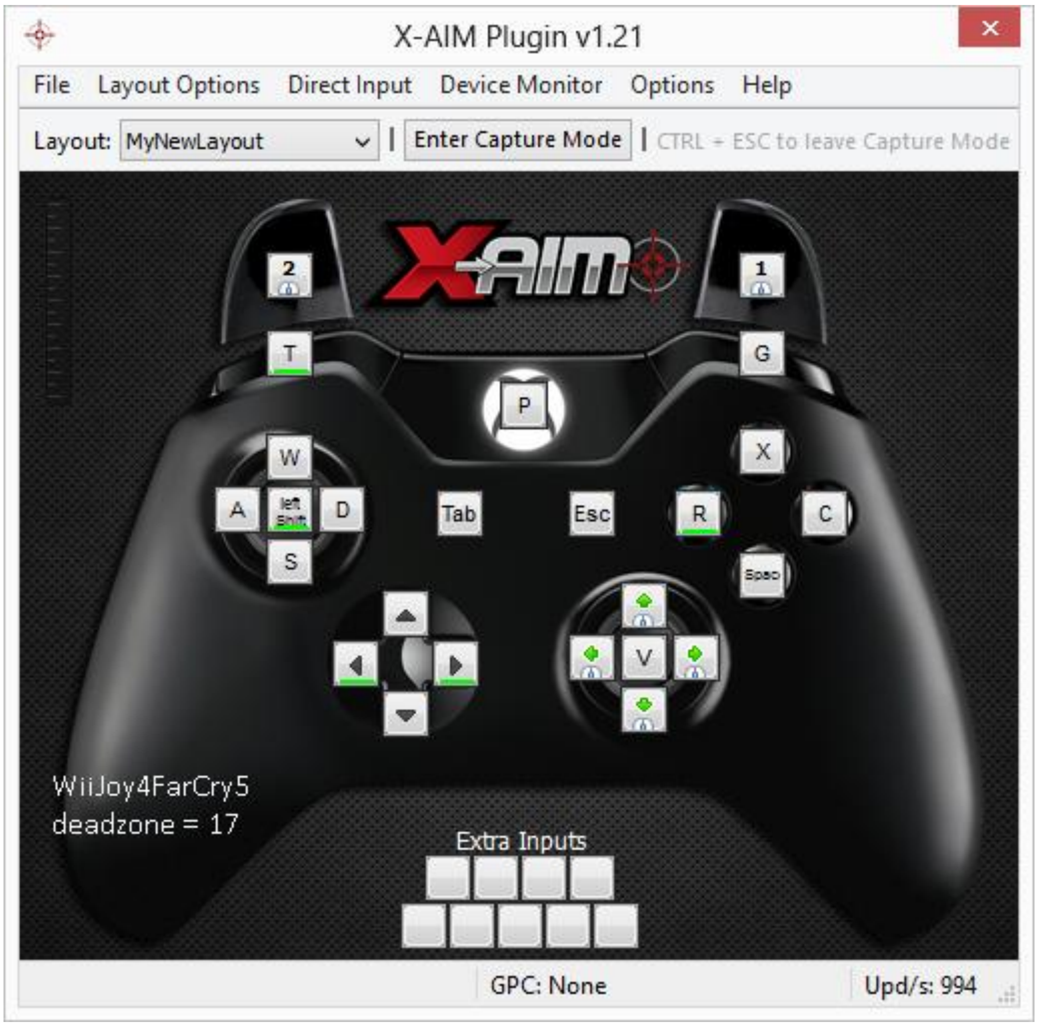
GPC: None
Upd/s: 1000



Layout: MyNewLayout

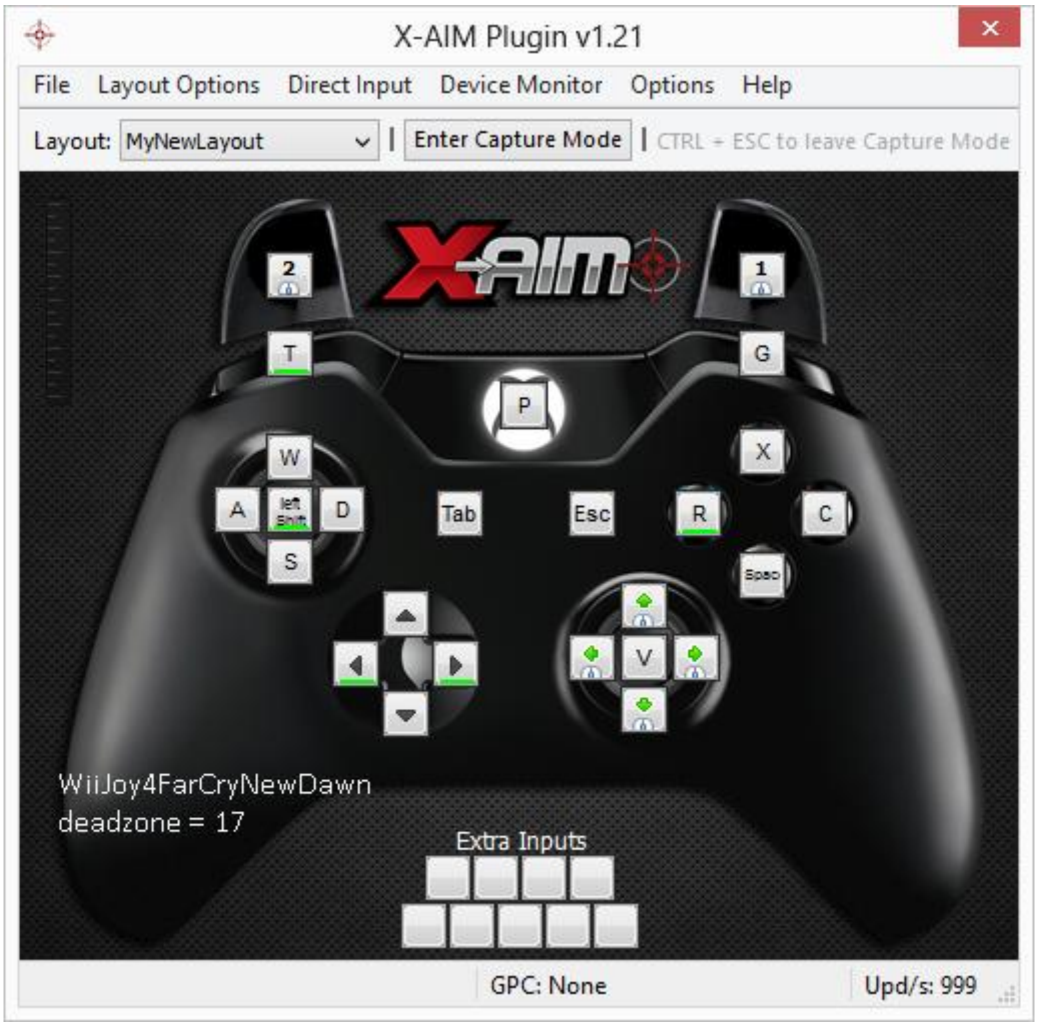
WiiJoy4DoomEternal
deadzone = 15

GPC: None
Upd/s: 1000

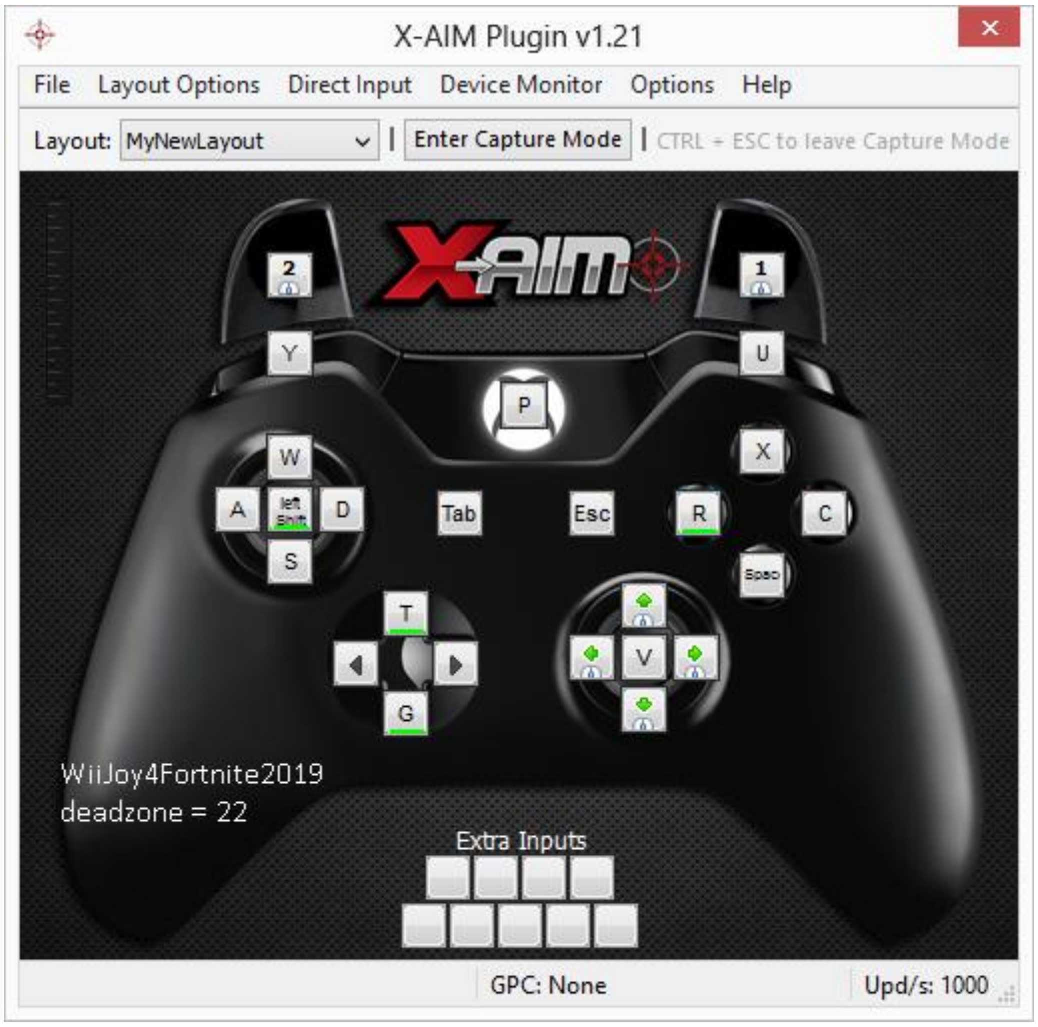


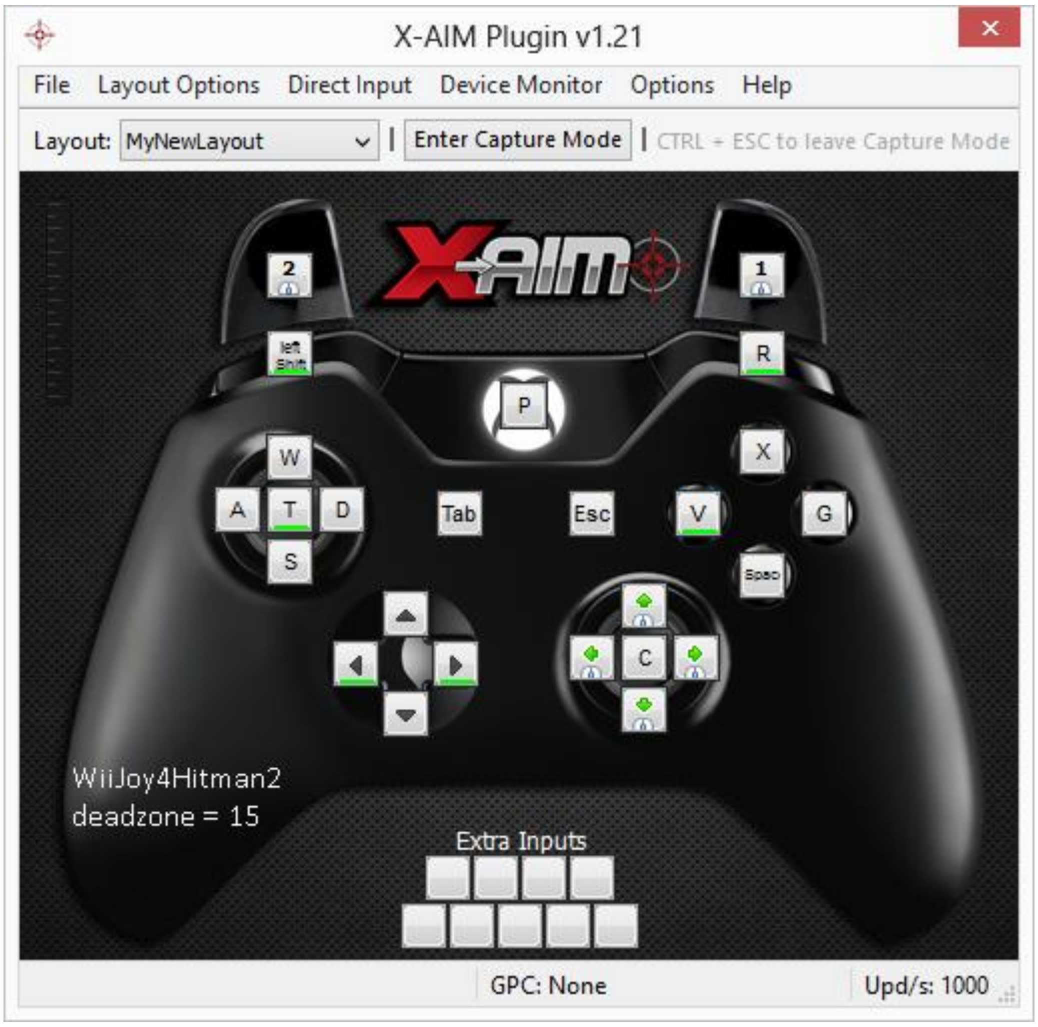
Layout: MyNewLayout

WiiJoy4FarCry5
deadzone = 17

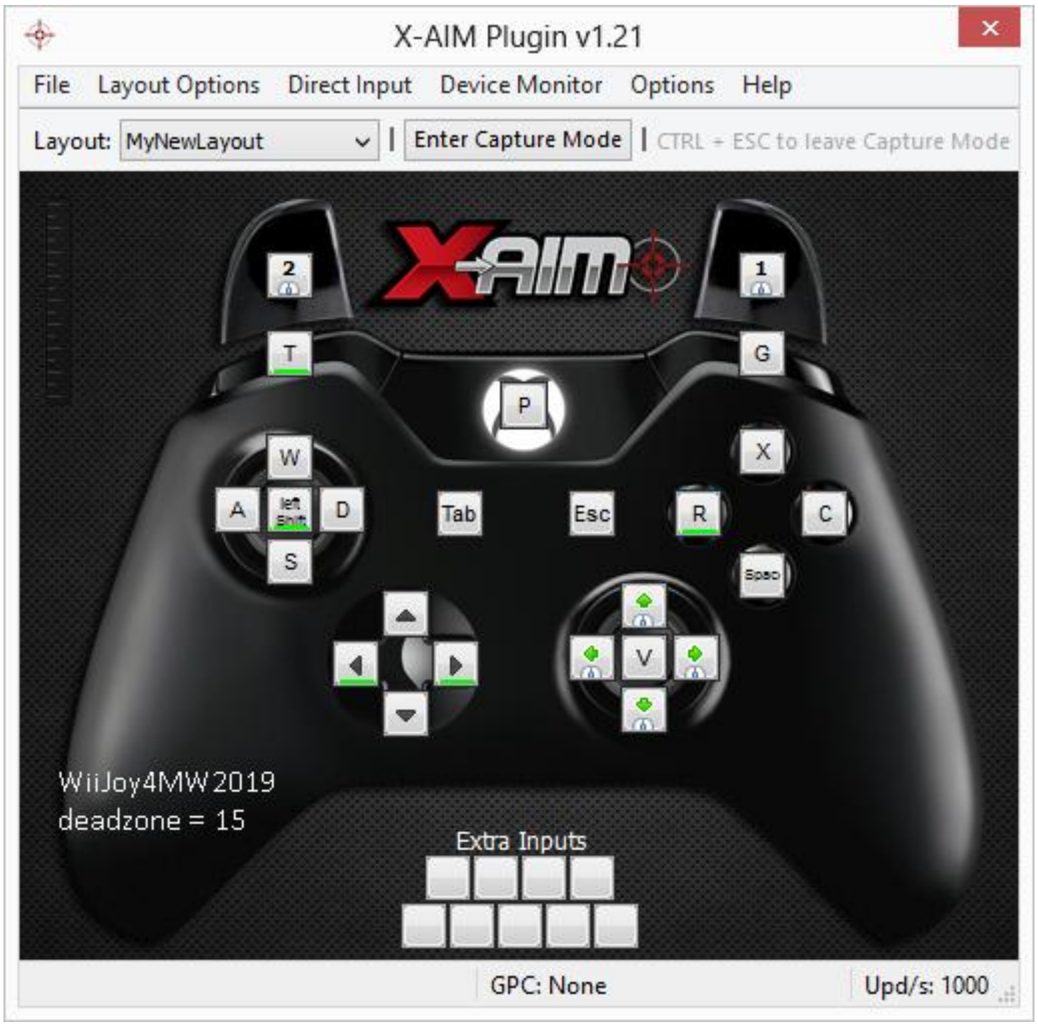


WiiJoy4FarCryNewDawn
deadzone = 17


X-AIM Plugin v1.21
File
Layout Options
Direct Input
Device Monitor
Options
Help
Layout: MyNewLayout
Enter Capture Mode
CTRL + ESC to leave Capture Mode
2
1
Y
U
P
W
A
left Shift
D
S
Tab
Esc
R
X
C
Spac
T
G
V
WiiJoy4Fortnite2019
deadzone = 22
Extra Inputs
GPC: None
Upd/s: 1000

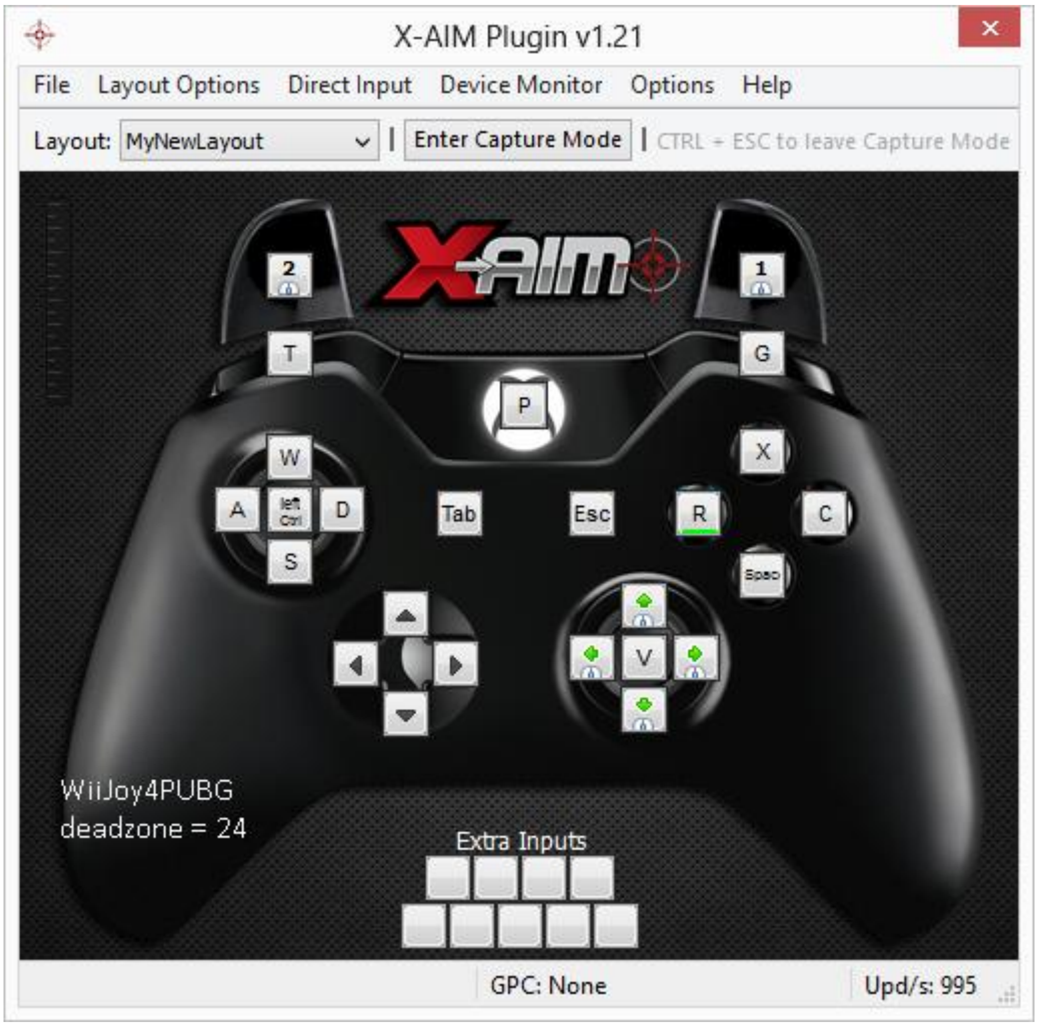


WiiJoy4Hitman2
deadzone = 15

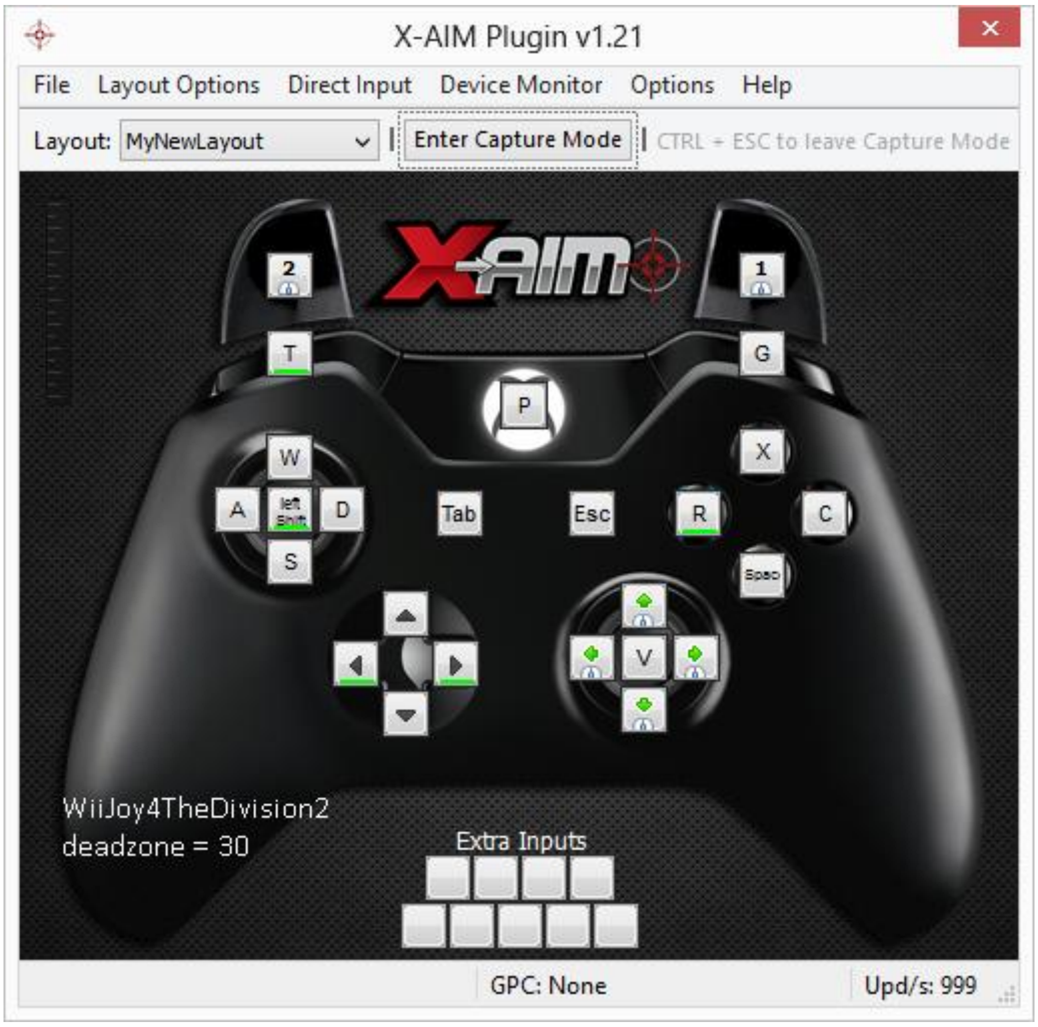


Layout: MyNewLayout

WiiJoy4MW2019
deadzone = 15



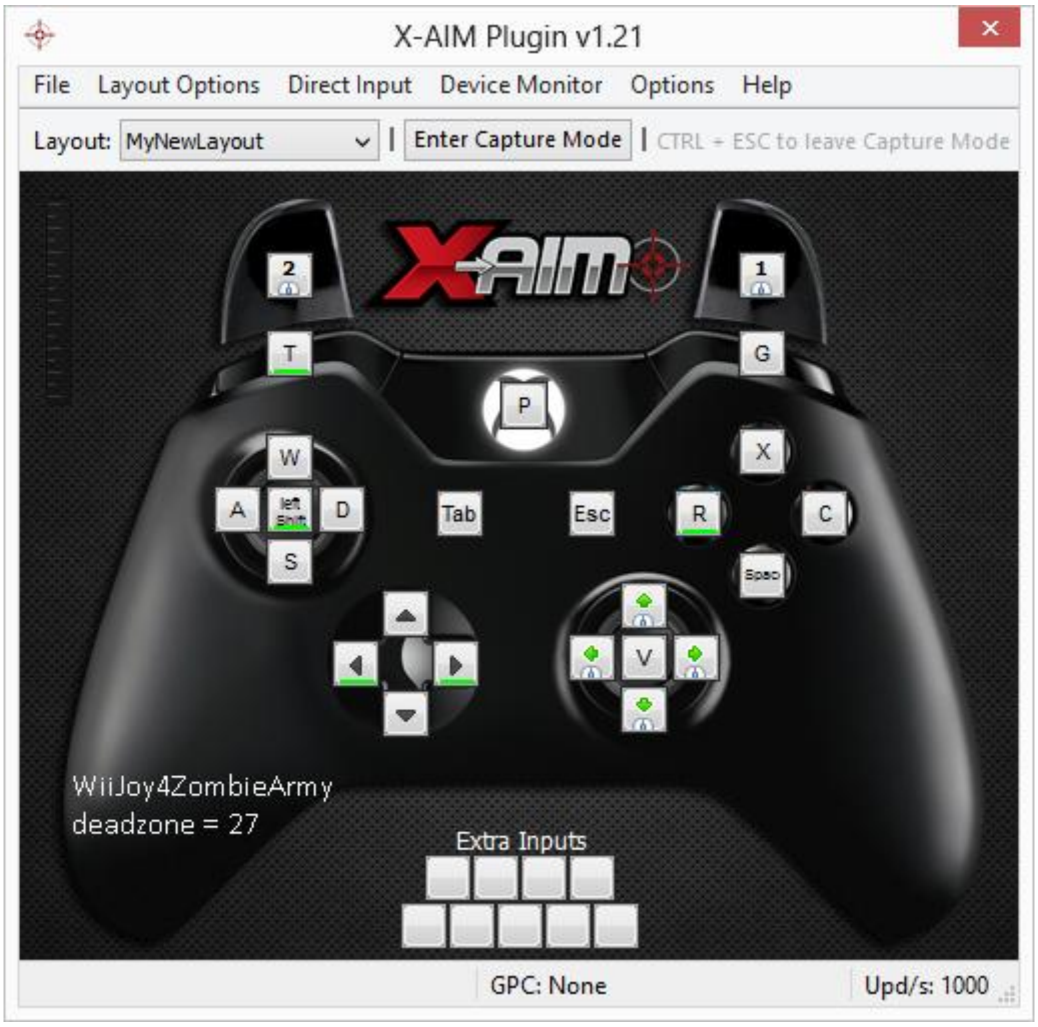
WiiJoy4PUBG
deadzone = 24



Layout: MyNewLayout

WiiJoy4TheDivision2
deadzone = 30

GPC: None
Upd/s: 999



WiiJoy4ZombieArmy
deadzone = 27

3. Configuration of key bindings for games

```
Joycon/Wiimote:
LeftButtonSHOULDER_2 = LeftControl
LeftButtonMINUS = Enter
AccelLeft = V
LeftButtonCAPTURE = P
LeftButtonDPAD_UP = UP
LeftButtonDPAD_LEFT = LEFT
LeftButtonDPAD_DOWN = DOWN
LeftButtonDPAD_RIGHT = RIGHT
LeftButtonSTICK = LeftShift
LeftButtonSHOULDER_1 = Space
LeftButtonSL = B
LeftButtonSR = N
StickLeft = ZQSD
RollLeft = A
RollRight = E
AccelRight = R
ButtonStateOne = Tab
ButtonStateDown = C
ButtonStateHome = F/F+T+G
```

```
ButtonStateRight = U
ButtonStateLeft = Y
ButtonStateUp = X
ButtonStateTwo = Escape
ButtonStatePlus = G
ButtonStateMinus = T
ButtonStateB = MouseButtonLeft
ButtonStateA = MouseButtonRight
IR = MouseMove

Joycon alone:
LeftButtonSR = MouseButtonLeft
LeftButtonSL = MouseButtonRight
AccelLeft = A
LeftButtonSHOULDER_2 = G
LeftButtonDPAD_DOWN = RIGHT
LeftButtonDPAD_UP = LEFT
LeftButtonDPAD_LEFT = DOWN
LeftButtonDPAD_RIGHT = UP
LeftButtonSTICK = Tab
LeftButtonSHOULDER_1 = T
LeftButtonMINUS = Escape
StickLeft = ZQSD
LeftButtonCAPTURE = P
Accelerometer/Stick/SRSL = MouseMove, XCYU/ZQSD

Joycons:
LeftButtonSHOULDER_1 = Control
LeftButtonMINUS = Return
LeftButtonCAPTURE = P
LeftButtonDPAD_UP = UP
LeftButtonDPAD_LEFT = LEFT
LeftButtonDPAD_DOWN = DOWN
LeftButtonDPAD_RIGHT = RIGHT
LeftButtonSTICK = LeftShift
LeftButtonSHOULDER_2 = MouseButtonRight
LeftButtonSL = A
LeftButtonSR = E
StickLeft = Z, S, Q, D
RightButtonPLUS = Tab
RightButtonDPAD_DOWN = Space
RightButtonHOME = F/F+T+G
RightButtonDPAD_RIGHT = C
RightButtonDPAD_LEFT = Y
RightButtonDPAD_UP = X
RightButtonSTICK = Escape
RightButtonSL = G
RightButtonSR= T
RightButtonSHOULDER_1 = U
RightButtonSHOULDER_2 = MouseButtonLeft
StickRight = 1, 2, 3, 4
```

```
Laser/Sticks = MouseMove
```

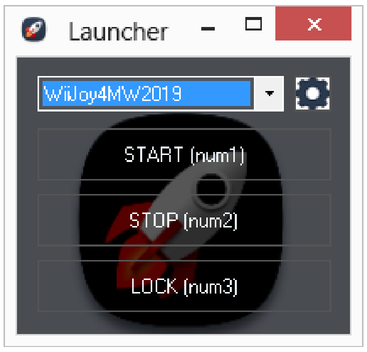
4. GPC scripts

Cronusmax device runs better with firmware operationnel mode set to Tournament Edition and you can still use gpc scripts with plugin X-Aim. For Ori 2, you can use the wheel for perks with mouse control instead of ZQSD loading the following gpc script.

scriptOri2:

```
main
{
	if (get_val(CEMU_EXTRA1))
	{
		swap(XB1_RX,XB1_LX);
		swap(XB1_RY,XB1_LY);
	}
}
```

5. Launcher



```
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 System.Runtime.InteropServices;
using Microsoft.VisualBasic;
using Microsoft.CSharp;
using System.CodeDom;
using System.Reflection;
namespace Launcher
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        public static bool Closinggetstate, Switchtoentercapture, getstate;
        public static bool[] _Valuechanged = new bool[5], _valuechanged = new bool[5];
        public static BackgroundWorker backgroundWorkerS = new BackgroundWorker();
        public static uint CurrentResolution = 0;
        public static int turnoffcontrollerX, turnoffcontrollerY, entercaptureX,
entercaptureY, xaimfileX, xaimfileY, xaimimportX, xaimimportY;
        string code, addedcode, finalcode;
        public bool this[int i]
        {
            get { return _valuechanged[i]; }
            set
            {
                if (_valuechanged[i] != value)
                    _Valuechanged[i] = true;
                else
                    _Valuechanged[i] = false;
                _valuechanged[i] = value;
            }
        }
        private void start()
        {
            if (!Switchtoentercapture)
            {
                System.Diagnostics.Process cmd = System.Diagnostics.Process.Start("X-
AIM.cmd");
                try
                {
                    System.Diagnostics.Process process =
System.Diagnostics.Process.Start(@"C:\Program Files (x86)\CronusPRO\Cronus.exe");
                    process.PriorityClass =
System.Diagnostics.ProcessPriorityClass.RealTime;
                    while (GetActiveWindowTitle() != "Plugin Warning")
                    {
                        try
                        {
                            var time = process.StartTime;
                            IntPtr handle;
                            handle = process.MainWindowHandle;
                            SwitchToThisWindow(handle, true);
                            if (GetActiveWindowTitle() == "Cronus Pro v1.21  [ADMIN]")
                            {
                                Interaction.AppActivate("Cronus Pro v1.21  [ADMIN]");
                                SendKeys.Send("%{F4}");
                                System.Threading.Thread.Sleep(2000);
                            }
                            else
                                if (GetActiveWindowTitle() == "X-AIM Plugin v1.21")
                                {
                                    SetCursorPos(xaimfileX, xaimfileY);
                                    LeftClick();
                                    LeftClickF();
                                    System.Threading.Thread.Sleep(300);
                                    SetCursorPos(xaimimportX, xaimimportY);
                                    LeftClick();
                                    LeftClickF();
                                    while (GetActiveWindowTitle() != "Import Layout")
                                        System.Threading.Thread.Sleep(500);
                                    if (GetActiveWindowTitle() == "Import Layout")
                                    {
                                        System.Threading.Thread.Sleep(1000);
                                        string scriptname = comboBox1.Text + ".glf";
                                        SendKeys.Send(scriptname);
                                        SendKeys.Send("{ENTER}");
                                        break;
                                    }
                                }
                            }
                            catch { }
                            System.Threading.Thread.Sleep(1);
                        }
                    }
                    catch { }
                    if (GetActiveWindowTitle() != "Plugin Warning")
                    {
                        try
                        {
                            System.Diagnostics.Process process =
System.Diagnostics.Process.Start(comboBox1.Text + ".exe");
                            process.PriorityClass =
System.Diagnostics.ProcessPriorityClass.RealTime;
                        }
                        catch { }
                        getstate = false;
                        Switchtoentercapture = true;
                    }
                }
                else
                {
                    try
                    {
                        IntPtr handle = FindWindow(null, "X-AIM Plugin v1.21");
                        SwitchToThisWindow(handle, true);
                        System.Threading.Thread.Sleep(300);
                        SendKeys.Send("{F4}");
                        System.Threading.Thread.Sleep(300);
                        SetCursorPos(turnoffcontrollerX, turnoffcontrollerY);
                        LeftClick();
                        LeftClickF();
                        System.Threading.Thread.Sleep(300);
                        SendKeys.Send("%{F4}");
                        System.Threading.Thread.Sleep(300);
                        SetCursorPos(entercaptureX, entercaptureY);
                        LeftClick();
                        LeftClickF();
                        System.Threading.Thread.Sleep(100);
                        uint dec = MapVirtualKey(0x6E, 0);
                        SimulateKeyDown(0x6E, (UInt16)dec);
                        System.Threading.Thread.Sleep(200);
                        SimulateKeyUp(0x6E, (UInt16)dec);
                        getstate = true;
                    }
                    catch { }
                }
            }
            private void stop()
            {
                IntPtr handle;
                System.Diagnostics.Process[] workers =
System.Diagnostics.Process.GetProcessesByName("Cronus");
                foreach (System.Diagnostics.Process worker in workers)
                {
                    try
                    {
                        handle = worker.MainWindowHandle;
                        SwitchToThisWindow(handle, true);
                        System.Threading.Thread.Sleep(500);
                        SendKeys.Send("%{F4}");
                        System.Threading.Thread.Sleep(300);
                        if (GetActiveWindowTitle() == "X-AIM Plugin v1.21")
                        {
```

```
                    SendKeys.Send("%{F4}");
                    System.Threading.Thread.Sleep(300);
                }
                worker.Kill();
                worker.WaitForExit();
                worker.Dispose();
            }
            catch { }
        }
        workers = System.Diagnostics.Process.GetProcessesByName(comboBox1.Text);
        foreach (System.Diagnostics.Process worker in workers)
        {
            try
            {
                handle = worker.MainWindowHandle;
                SwitchToThisWindow(handle, true);
                System.Threading.Thread.Sleep(500);
                SendKeys.Send("%{F4}");
            }
            catch { }
        }
        getstate = false;
        Switchtoentercapture = false;
    }
    private void button1_Click(object sender, EventArgs e)
    {
        start();
    }
    private void button2_Click(object sender, EventArgs e)
    {
        stop();
    }
    private void button3_Click(object sender, EventArgs e)
    {
        stop();
        System.Threading.Thread.Sleep(1500);
        LockWorkStation();
    }
    private void initConf()
    {
        string last;
        System.IO.StreamReader file = new System.IO.StreamReader("initconf.txt");
        file.ReadLine();
        last = file.ReadLine();
        file.ReadLine();
        turnoffcontrollerX = Convert.ToInt16(file.ReadLine());
        turnoffcontrollerY = Convert.ToInt16(file.ReadLine());
        file.ReadLine();
        entercaptureX = Convert.ToInt16(file.ReadLine());
        entercaptureY = Convert.ToInt16(file.ReadLine());
        file.ReadLine();
        xaimfileX = Convert.ToInt16(file.ReadLine());
        xaimfileY = Convert.ToInt16(file.ReadLine());
        file.ReadLine();
        xaimimportX = Convert.ToInt16(file.ReadLine());
        xaimimportY = Convert.ToInt16(file.ReadLine());
        file.ReadLine();
        string itemtoadd;
        int i = 0;
        do
        {
            itemtoadd = file.ReadLine();
            if (itemtoadd != "")
            {
                comboBox1.Items.Insert(i, itemtoadd);
                i++;
                }
                else
                    break;
            }
            while (itemtoadd != "" | i == 0);
            file.Close();
            comboBox1.Text = last;
        }
        private void Form1_Shown(object sender, EventArgs e)
        {
            Clipboard.Clear();
            TimeBeginPeriod(1);
            NtSetTimerResolution(1, true, ref CurrentResolution);
            this.Location = new System.Drawing.Point(300, 200);
            try { initConf(); }
            catch
            {
                using (System.IO.StreamWriter createdfile =
System.IO.File.AppendText("initconf.txt"))
                {
                    createdfile.WriteLine("//Last");
                    createdfile.WriteLine("WiimoteOnFortnite");
                    createdfile.WriteLine("//turn off controller");
                    createdfile.WriteLine("1270");
                    createdfile.WriteLine("375");
                    createdfile.WriteLine("//enter capture");
                    createdfile.WriteLine("1040");
                    createdfile.WriteLine("180");
                    createdfile.WriteLine("//xaim file");
                    createdfile.WriteLine("806");
                    createdfile.WriteLine("150");
                    createdfile.WriteLine("//xaim import");
                    createdfile.WriteLine("850");
                    createdfile.WriteLine("240");
                    createdfile.WriteLine("//List");
                    createdfile.WriteLine("WiiJoy4Fortnite");
                    createdfile.WriteLine("");
                    createdfile.Close();
                }
                initConf();
            }
            comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;
            System.IO.StreamReader file = new System.IO.StreamReader("initconf.txt");
            comboBox1.Text = file.ReadLine();
            file.Close();
            backgroundWorkerS.DoWork += new DoWorkEventHandler(FormStart);
            backgroundWorkerS.RunWorkerAsync();
        }
        private void Form1_FormClosed(object sender, FormClosedEventArgs e)
        {
            TimeEndPeriod(1);
            System.IO.StreamWriter initfile = new System.IO.StreamWriter("initconf.txt");
            initfile.WriteLine("//Last");
            initfile.WriteLine(comboBox1.Text);
            initfile.WriteLine("//turn off controller");
            initfile.WriteLine(turnoffcontrollerX.ToString());
            initfile.WriteLine(turnoffcontrollerY.ToString());
            initfile.WriteLine("//enter capture");
            initfile.WriteLine(entercaptureX.ToString());
            initfile.WriteLine(entercaptureY.ToString());
            initfile.WriteLine("//xaim file");
            initfile.WriteLine(xaimfileX.ToString());
            initfile.WriteLine(xaimfileY.ToString());
            initfile.WriteLine("//xaim import");
            initfile.WriteLine(xaimimportX.ToString());
            initfile.WriteLine(xaimimportY.ToString());
            initfile.WriteLine("//List");
            int numberofitems = comboBox1.Items.Count;
            for (int i = 0; i < numberofitems; i++)
                initfile.WriteLine(comboBox1.Items[i].ToString());
            initfile.WriteLine("");
            initfile.Close();
            Closinggetstate = true;
        }
        private string GetActiveWindowTitle()
        {
            const int nChars = 256;
            StringBuilder Buff = new StringBuilder(nChars);
            IntPtr handle = GetForegroundWindow();
            if (GetWindowText(handle, Buff, nChars) > 0)
                return Buff.ToString();
            return null;
        }
        private void FormStart(object sender, DoWorkEventArgs e)
        {
            for (; ; )
            {
                if (Closinggetstate)
                    return;
                this[1] = GetAsyncKeyState(System.Windows.Forms.Keys.NumPad1);
                if (_Valuechanged[1] & !this[1])
                    performClickStart(408, 300);
                this[2] = GetAsyncKeyState(System.Windows.Forms.Keys.NumPad2);
                if (_Valuechanged[2] & !this[2])
                    performClickStop(408, 333);
                this[3] = GetAsyncKeyState(System.Windows.Forms.Keys.NumPad3);
                if (_Valuechanged[3] & !this[3])
                    performClickStop(408, 366);
                System.Threading.Thread.Sleep(100);
            }
        }
        private void performClickStart(int X, int Y)
        {
            if (!Switchtoentercapture | !getstate)
            {
                System.Diagnostics.Process[] workers =
System.Diagnostics.Process.GetProcessesByName("Launcher");
                foreach (System.Diagnostics.Process worker in workers)
                {
                    try
                    {
                        IntPtr handle;
                        handle = worker.MainWindowHandle;
                        SwitchToThisWindow(handle, true);
                        System.Threading.Thread.Sleep(500);
                        SetCursorPos(X, Y);
                        LeftClick();
                        LeftClickF();
                    }
                    catch { }
                }
                getstate = true;
            }
            else
            {
                if (getstate)
                {
                    uint dec = MapVirtualKey(0x6E, 0);
                    SimulateKeyDown(0x6E, (UInt16)dec);
                    System.Threading.Thread.Sleep(200);
                    SimulateKeyUp(0x6E, (UInt16)dec);
                    System.Threading.Thread.Sleep(200);
                    System.Diagnostics.Process[] workers =
System.Diagnostics.Process.GetProcessesByName("Launcher");
                    foreach (System.Diagnostics.Process worker in workers)
                    {
                        try
                        {
                            IntPtr handle;
                            handle = worker.MainWindowHandle;
                            SwitchToThisWindow(handle, true);
                        }
                        catch { }
                    }
                    getstate = false;
                }
            }
        }
        private void performClickStop(int X, int Y)
        {
            if (getstate)
            {
                uint dec = MapVirtualKey(0x6E, 0);
                SimulateKeyDown(0x6E, (UInt16)dec);
                System.Threading.Thread.Sleep(200);
                SimulateKeyUp(0x6E, (UInt16)dec);
                System.Threading.Thread.Sleep(200);
            }
            System.Diagnostics.Process[] workers =
System.Diagnostics.Process.GetProcessesByName("Launcher");
            foreach (System.Diagnostics.Process worker in workers)
            {
                try
                {
                    IntPtr handle;
                    handle = worker.MainWindowHandle;
                    SwitchToThisWindow(handle, true);
                    System.Threading.Thread.Sleep(500);
                    SetCursorPos(X, Y);
                    LeftClick();
                    LeftClickF();
                    System.Threading.Thread.Sleep(200);
                }
                catch { }
            }
        }
        static readonly IntPtr HWND_TOPMOST = new IntPtr(-1);
        const UInt32 SWP_NOSIZE = 0x0001;
        const UInt32 SWP_NOMOVE = 0x0002;
        const UInt32 SWP_SHOWWINDOW = 0x0040;
        [DllImport("user32.dll")]
        static extern IntPtr FindWindow(string className, string windowName);
        [DllImport("user32.dll")]
        static extern bool SetWindowPos(IntPtr hWnd, IntPtr hWndInsertAfter, int X, int Y,
int cX, int cY, uint uFlags);
        [DllImport("user32.dll")]
        static extern bool GetAsyncKeyState(System.Windows.Forms.Keys vKey);
        [DllImport("user32.dll")]
        static extern IntPtr GetForegroundWindow();
        [DllImport("user32.dll")]
        static extern int GetWindowText(IntPtr hWnd, StringBuilder text, int count);
        [DllImport("system32/user32.dll")]
        public static extern bool SwitchToThisWindow(IntPtr handle, bool fAltTab);
        [DllImport("system32/user32.dll")]
        public static extern uint SendInput(uint numberOfInputs, INPUT[] inputs, int
sizeOfInputStructure);
        [DllImport("system32/user32.dll")]
        public static extern uint MapVirtualKey(uint uCode, uint uMapType);
        [DllImport("system32/user32.dll")]
        public static extern bool LockWorkStation();
        [DllImport("User32.dll")]
        private static extern bool GetCursorPos(out int x, out int y);
        [DllImport("User32.dll")]
        private static extern bool SetCursorPos(int x, int y);
        [DllImport("winmm.dll", EntryPoint = "timeBeginPeriod")]
        public static extern uint TimeBeginPeriod(uint ms);
        [DllImport("winmm.dll", EntryPoint = "timeEndPeriod")]
        public static extern uint TimeEndPeriod(uint ms);
        [DllImport("ntdll.dll", EntryPoint = "NtSetTimerResolution")]
        public static extern void NtSetTimerResolution(uint DesiredResolution, bool
SetResolution, ref uint CurrentResolution);
        [StructLayout(System.Runtime.InteropServices.LayoutKind.Explicit)]
        public struct MOUSEKEYBDHARDWAREINPUT
        {
            [FieldOffset(0)]
            public MOUSEINPUT Mouse;
            [FieldOffset(0)]
            public KEYBDINPUT Keyboard;
        }
        public struct INPUT
        {
            public UInt32 Type;
            public MOUSEKEYBDHARDWAREINPUT Data;
        }
        public enum InputType : uint
        {
            MOUSE = 0, KEYBOARD = 1, HARDWARE = 2,
        }
        public struct KEYBDINPUT
        {
            public UInt16 Vk;
            public UInt16 Scan;
            public uint Flags;
            public uint Time;
            public IntPtr ExtraInfo;
        }
        public struct MOUSEINPUT
        {
            public int X;
            public int Y;
            public int MouseData;
            public uint Flags;
            public uint Time;
            public IntPtr ExtraInfo;
        }
        public static INPUT[] MiceI = new INPUT[1], MiceIf = new INPUT[1], down = new
INPUT[1], up = new INPUT[1];
        public static int size =
System.Runtime.InteropServices.Marshal.SizeOf(typeof(INPUT));
        public static void LeftClick()
        {
            MiceI[0].Type = (UInt32)InputType.MOUSE;
            MiceI[0].Data.Mouse = new MOUSEINPUT();
            MiceI[0].Data.Mouse.MouseData = 0;
            MiceI[0].Data.Mouse.Flags = 0x0002;
            MiceI[0].Data.Mouse.Time = 0;
            MiceI[0].Data.Mouse.ExtraInfo = IntPtr.Zero;
            SendInput(1, MiceI, size);
        }
        public static void LeftClickF()
        {
            MiceIf[0].Type = (UInt32)InputType.MOUSE;
            MiceIf[0].Data.Mouse = new MOUSEINPUT();
            MiceIf[0].Data.Mouse.MouseData = 0;
            Micelf[0].Data.Mouse.Flags = 0x0004;
            Micelf[0].Data.Mouse.Time = 0;
            Micelf[0].Data.Mouse.ExtraInfo = IntPtr.Zero;
            SendInput(1, Micelf, size);
        }
        public static void SimulateKeyDown(UInt16 keyCode, UInt16 bScan)
        {
            down[0].Type = (UInt32)InputType.KEYBOARD;
            down[0].Data.Keyboard = new KEYBDINPUT();
            down[0].Data.Keyboard.Time = 0;
            down[0].Data.Keyboard.ExtraInfo = IntPtr.Zero;
            down[0].Data.Keyboard.Flags = 0;
            down[0].Data.Keyboard.Vk = keyCode;
            down[0].Data.Keyboard.Scan = bScan;
            SendInput(1, down, size);
        }
        public static void SimulateKeyUp(UInt16 keyCode, UInt16 bScan)
        {
            up[0].Type = (UInt32)InputType.KEYBOARD;
            up[0].Data.Keyboard = new KEYBDINPUT();
            up[0].Data.Keyboard.Time = 0;
            up[0].Data.Keyboard.ExtraInfo = IntPtr.Zero;
            up[0].Data.Keyboard.Flags = 0x0002;
            up[0].Data.Keyboard.Vk = keyCode;
            up[0].Data.Keyboard.Scan = bScan;
            SendInput(1, up, size);
        }
        private void initCode()
        {
            code = @"
                using System;
                using System.Runtime.InteropServices;
                namespace StringToCode
                {
                    public class FooClass
                    {
                        Input input = new Input();
                        funct_driver
                        public bool this[int i]
                        {
                            get { return _valuechanged[i]; }
                            set
                            {
                                if (_valuechanged[i] != value)
                                    _Valuechanged[i] = true;
                                else
                                    _Valuechanged[i] = false;
                                _valuechanged[i] = value;
                            }
                        }
                        public void Open(double irx, double iry, double watchM)
                        {
                            input.KeyboardFilterMode = KeyboardFilterMode.All;
                            input.MouseFilterMode = MouseFilterMode.All;
                            input.Load();
                        }
                        public void Close(double irx, double iry, double watchM)
                        {
                            input.Unload();
                        }
                        [DllImport(""user32.dll"")]
                        public static extern bool
GetAsyncKeyState(System.Windows.Forms.Keys vKey);
                        [DllImport(""InputSending.dll"", EntryPoint = ""MoveMouseTo"",
CallingConvention = CallingConvention.Cdecl)]
                        public static extern void MoveMouseTo(int x, int y);
                        [DllImport(""InputSending.dll"", EntryPoint = ""MoveMouseBy"",
CallingConvention = CallingConvention.Cdecl)]
                        public static extern void MoveMouseBy(int x, int y);
                        [DllImport(""InputSending.dll"", EntryPoint = ""SendKey"",
CallingConvention = CallingConvention.Cdecl)]
                        public static extern void SendKey(UInt16 bVk, UInt16 bScan);
                        [DllImport(""InputSending.dll"", EntryPoint = ""SendKeyF"",
CallingConvention = CallingConvention.Cdecl)]
                        public static extern void SendKeyF(UInt16 bVk, UInt16 bScan);
                        [DllImport(""InputSending.dll"", EntryPoint = ""SendKeyArrows"",
CallingConvention = CallingConvention.Cdecl)]
                        public static extern void SendKeyArrows(UInt16 bVk, UInt16 bScan);
                        [DllImport(""InputSending.dll"", EntryPoint = ""SendKeyArrowsF"",
CallingConvention = CallingConvention.Cdecl)]
                        public static extern void SendKeyArrowsF(UInt16 bVk, UInt16 bScan);
                        [DllImport(""InputSending.dll"", EntryPoint =
""SendMouseEventButtonLeft"", CallingConvention = CallingConvention.Cdecl)]
                        public static extern void SendMouseEventButtonLeft();
                        [DllImport(""InputSending.dll"", EntryPoint =
""SendMouseEventButtonLeftF"", CallingConvention = CallingConvention.Cdecl)]
                        public static extern void SendMouseEventButtonLeftF();
                        [DllImport(""InputSending.dll"", EntryPoint =
""SendMouseEventButtonRight"", CallingConvention = CallingConvention.Cdecl)]
                        public static extern void SendMouseEventButtonRight();
                        [DllImport(""InputSending.dll"", EntryPoint =
""SendMouseEventButtonRightF"", CallingConvention = CallingConvention.Cdecl)]
                        public static extern void SendMouseEventButtonRightF();
                        [DllImport(""InputSending.dll"", EntryPoint =
""SendMouseEventButtonMiddle"", CallingConvention = CallingConvention.Cdecl)]
                        public static extern void SendMouseEventButtonMiddle();
                        [DllImport(""InputSending.dll"", EntryPoint =
""SendMouseEventButtonMiddleF"", CallingConvention = CallingConvention.Cdecl)]
                        public static extern void SendMouseEventButtonMiddleF();
                        [DllImport(""InputSending.dll"", EntryPoint =
""SendMouseEventButtonWheelUp"", CallingConvention = CallingConvention.Cdecl)]
                        public static extern void SendMouseEventButtonWheelUp();
                        [DllImport(""InputSending.dll"", EntryPoint =
""SendMouseEventButtonWheelDown"", CallingConvention = CallingConvention.Cdecl)]
                        public static extern void SendMouseEventButtonWheelDown();
                        [DllImport(""SendInputLibrary.dll"", EntryPoint =
""SimulateKeyDown"", CallingConvention = CallingConvention.Cdecl)]
                        public static extern void SimulateKeyDown(UInt16 keyCode, UInt16
bScan);
                        [DllImport(""SendInputLibrary.dll"", EntryPoint =
""SimulateKeyUp"", CallingConvention = CallingConvention.Cdecl)]
                        public static extern void SimulateKeyUp(UInt16 keyCode, UInt16
bScan);
                        [DllImport(""SendInputLibrary.dll"", EntryPoint =
""SimulateKeyDownArrows"", CallingConvention = CallingConvention.Cdecl)]
                        public static extern void SimulateKeyDownArrows(UInt16 keyCode,
UInt16 bScan);
                        [DllImport(""SendInputLibrary.dll"", EntryPoint =
""SimulateKeyUpArrows"", CallingConvention = CallingConvention.Cdecl)]
                        public static extern void SimulateKeyUpArrows(UInt16 keyCode,
UInt16 bScan);
                        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""MouseMW3"",
CallingConvention = CallingConvention.Cdecl)]
                        public static extern void MouseMW3(int x, int y);
                        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""MouseBrink"",
CallingConvention = CallingConvention.Cdecl)]
                        public static extern void MouseBrink(int x, int y);
                        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""LeftClick"",
CallingConvention = CallingConvention.Cdecl)]
                        public static extern void LeftClick();
                        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""LeftClickF"",
CallingConvention = CallingConvention.Cdecl)]
```

```
                        public static extern void LeftClickF();
                        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""RightClick"",
CallingConvention = CallingConvention.Cdecl)]
                        public static extern void RightClick();
                        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""RightClickF"",
CallingConvention = CallingConvention.Cdecl)]
                        public static extern void RightClickF();
                        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""MiddleClick"",
CallingConvention = CallingConvention.Cdecl)]
                        public static extern void MiddleClick();
                        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""MiddleClickF"",
CallingConvention = CallingConvention.Cdecl)]
                        public static extern void MiddleClickF();
                        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""WheelDownF"",
CallingConvention = CallingConvention.Cdecl)]
                        public static extern void WheelDownF();
                        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""WheelUpF"",
CallingConvention = CallingConvention.Cdecl)]
                        public static extern void WheelUpF();
                        [DllImport(""user32.dll"")]
                        public static extern void SetPhysicalCursorPos(int X, int Y);
                        [DllImport(""user32.dll"")]
                        public static extern void SetCaretPos(int X, int Y);
                        [DllImport(""user32.dll"")]
                        public static extern void SetCursorPos(int X, int Y);
                        [DllImport(""system32/user32.dll"")]
                        public static extern uint MapVirtualKey(uint uCode, uint uMapType);
                        public static ushort VK_LBUTTON = (ushort)0x01;
                        public static ushort VK_RBUTTON = (ushort)0x02;
                        public static ushort VK_CANCEL = (ushort)0x03;
                        public static ushort VK_MBUTTON = (ushort)0x04;
                        public static ushort VK_XBUTTON1 = (ushort)0x05;
                        public static ushort VK_XBUTTON2 = (ushort)0x06;
                        public static ushort VK_BACK = (ushort)0x08;
                        public static ushort VK_Tab = (ushort)0x09;
                        public static ushort VK_CLEAR = (ushort)0x0C;
                        public static ushort VK_Return = (ushort)0x0D;
                        public static ushort VK_SHIFT = (ushort)0x10;
                        public static ushort VK_CONTROL = (ushort)0x11;
                        public static ushort VK_MENU = (ushort)0x12;
                        public static ushort VK_PAUSE = (ushort)0x13;
                        public static ushort VK_CAPITAL = (ushort)0x14;
                        public static ushort VK_KANA = (ushort)0x15;
                        public static ushort VK_HANGEUL = (ushort)0x15;
                        public static ushort VK_HANGUL = (ushort)0x15;
                        public static ushort VK_JUNJA = (ushort)0x17;
                        public static ushort VK_FINAL = (ushort)0x18;
                        public static ushort VK_HANJA = (ushort)0x19;
                        public static ushort VK_KANJI = (ushort)0x19;
                        public static ushort VK_Escape = (ushort)0x1B;
                        public static ushort VK_CONVERT = (ushort)0x1C;
                        public static ushort VK_NONCONVERT = (ushort)0x1D;
                        public static ushort VK_ACCEPT = (ushort)0x1E;
                        public static ushort VK_MODECHANGE = (ushort)0x1F;
                        public static ushort VK_Space = (ushort)0x20;
                        public static ushort VK_PRIOR = (ushort)0x21;
                        public static ushort VK_NEXT = (ushort)0x22;
                        public static ushort VK_END = (ushort)0x23;
                        public static ushort VK_HOME = (ushort)0x24;
                        public static ushort VK_LEFT = (ushort)0x25;
                        public static ushort VK_UP = (ushort)0x26;
                        public static ushort VK_RIGHT = (ushort)0x27;
                        public static ushort VK_DOWN = (ushort)0x28;
                        public static ushort VK_SELECT = (ushort)0x29;
                        public static ushort VK_PRINT = (ushort)0x2A;
                        public static ushort VK_EXECUTE = (ushort)0x2B;
                        public static ushort VK_SNAPSHOT = (ushort)0x2C;
                        public static ushort VK_INSERT = (ushort)0x2D;
                        public static ushort VK_DELETE = (ushort)0x2E;
                        public static ushort VK_HELP = (ushort)0x2F;
                        public static ushort VK_APOSTROPHE = (ushort)0xDE;
                        public static ushort VK_0 = (ushort)0x30;
                        public static ushort VK_1 = (ushort)0x31;
                        public static ushort VK_2 = (ushort)0x32;
                        public static ushort VK_3 = (ushort)0x33;
                        public static ushort VK_4 = (ushort)0x34;
                        public static ushort VK_5 = (ushort)0x35;
                        public static ushort VK_6 = (ushort)0x36;
                        public static ushort VK_7 = (ushort)0x37;
                        public static ushort VK_8 = (ushort)0x38;
                        public static ushort VK_9 = (ushort)0x39;
                        public static ushort VK_A = (ushort)0x41;
                        public static ushort VK_B = (ushort)0x42;
                        public static ushort VK_C = (ushort)0x43;
                        public static ushort VK_D = (ushort)0x44;
                        public static ushort VK_E = (ushort)0x45;
                        public static ushort VK_F = (ushort)0x46;
                        public static ushort VK_G = (ushort)0x47;
                        public static ushort VK_H = (ushort)0x48;
                        public static ushort VK_I = (ushort)0x49;
                        public static ushort VK_J = (ushort)0x4A;
                        public static ushort VK_K = (ushort)0x4B;
                        public static ushort VK_L = (ushort)0x4C;
                        public static ushort VK_M = (ushort)0x4D;
                        public static ushort VK_N = (ushort)0x4E;
                        public static ushort VK_O = (ushort)0x4F;
                        public static ushort VK_P = (ushort)0x50;
                        public static ushort VK_Q = (ushort)0x51;
                        public static ushort VK_R = (ushort)0x52;
                        public static ushort VK_S = (ushort)0x53;
                        public static ushort VK_T = (ushort)0x54;
                        public static ushort VK_U = (ushort)0x55;
                        public static ushort VK_V = (ushort)0x56;
                        public static ushort VK_W = (ushort)0x57;
                        public static ushort VK_X = (ushort)0x58;
                        public static ushort VK_Y = (ushort)0x59;
                        public static ushort VK_Z = (ushort)0x5A;
                        public static ushort VK_LWIN = (ushort)0x5B;
                        public static ushort VK_RWIN = (ushort)0x5C;
                        public static ushort VK_APPS = (ushort)0x5D;
                        public static ushort VK_SLEEP = (ushort)0x5F;
                        public static ushort VK_NUMPAD0 = (ushort)0x60;
                        public static ushort VK_NUMPAD1 = (ushort)0x61;
                        public static ushort VK_NUMPAD2 = (ushort)0x62;
                        public static ushort VK_NUMPAD3 = (ushort)0x63;
                        public static ushort VK_NUMPAD4 = (ushort)0x64;
                        public static ushort VK_NUMPAD5 = (ushort)0x65;
                        public static ushort VK_NUMPAD6 = (ushort)0x66;
                        public static ushort VK_NUMPAD7 = (ushort)0x67;
                        public static ushort VK_NUMPAD8 = (ushort)0x68;
                        public static ushort VK_NUMPAD9 = (ushort)0x69;
                        public static ushort VK_MULTIPLY = (ushort)0x6A;
                        public static ushort VK_ADD = (ushort)0x6B;
                        public static ushort VK_SEPARATOR = (ushort)0x6C;
                        public static ushort VK_SUBTRACT = (ushort)0x6D;
                        public static ushort VK_DECIMAL = (ushort)0x6E;
                        public static ushort VK_DIVIDE = (ushort)0x6F;
                        public static ushort VK_F1 = (ushort)0x70;
                        public static ushort VK_F2 = (ushort)0x71;
                        public static ushort VK_F3 = (ushort)0x72;
                        public static ushort VK_F4 = (ushort)0x73;
                        public static ushort VK_F5 = (ushort)0x74;
                        public static ushort VK_F6 = (ushort)0x75;
                        public static ushort VK_F7 = (ushort)0x76;
                        public static ushort VK_F8 = (ushort)0x77;
                        public static ushort VK_F9 = (ushort)0x78;
                        public static ushort VK_F10 = (ushort)0x79;
                        public static ushort VK_F11 = (ushort)0x7A;
                        public static ushort VK_F12 = (ushort)0x7B;
                        public static ushort VK_F13 = (ushort)0x7C;
                        public static ushort VK_F14 = (ushort)0x7D;
                        public static ushort VK_F15 = (ushort)0x7E;
                        public static ushort VK_F16 = (ushort)0x7F;
                        public static ushort VK_F17 = (ushort)0x80;
                        public static ushort VK_F18 = (ushort)0x81;
                        public static ushort VK_F19 = (ushort)0x82;
                        public static ushort VK_F20 = (ushort)0x83;
                        public static ushort VK_F21 = (ushort)0x84;
                        public static ushort VK_F22 = (ushort)0x85;
                        public static ushort VK_F23 = (ushort)0x86;
                        public static ushort VK_F24 = (ushort)0x87;
                        public static ushort VK_NUMLOCK = (ushort)0x90;
                        public static ushort VK_SCROLL = (ushort)0x91;
                        public static ushort VK_LeftShift = (ushort)0xA0;
                        public static ushort VK_RightShift = (ushort)0xA1;
                        public static ushort VK_LeftControl = (ushort)0xA2;
                        public static ushort VK_RightControl = (ushort)0xA3;
                        public static ushort VK_LMENU = (ushort)0xA4;
                        public static ushort VK_RMENU = (ushort)0xA5;
                        public static ushort VK_BROWSER_BACK = (ushort)0xA6;
                        public static ushort VK_BROWSER_FORWARD = (ushort)0xA7;
                        public static ushort VK_BROWSER_REFRESH = (ushort)0xA8;
                        public static ushort VK_BROWSER_STOP = (ushort)0xA9;
                        public static ushort VK_BROWSER_SEARCH = (ushort)0xAA;
                        public static ushort VK_BROWSER_FAVORITES = (ushort)0xAB;
                        public static ushort VK_BROWSER_HOME = (ushort)0xAC;
                        public static ushort VK_VOLUME_MUTE = (ushort)0xAD;
                        public static ushort VK_VOLUME_DOWN = (ushort)0xAE;
                        public static ushort VK_VOLUME_UP = (ushort)0xAF;
                        public static ushort VK_MEDIA_NEXT_TRACK = (ushort)0xB0;
                        public static ushort VK_MEDIA_PREV_TRACK = (ushort)0xB1;
                        public static ushort VK_MEDIA_STOP = (ushort)0xB2;
                        public static ushort VK_MEDIA_PLAY_PAUSE = (ushort)0xB3;
                        public static ushort VK_LAUNCH_MAIL = (ushort)0xB4;
                        public static ushort VK_LAUNCH_MEDIA_SELECT = (ushort)0xB5;
                        public static ushort VK_LAUNCH_APP1 = (ushort)0xB6;
                        public static ushort VK_LAUNCH_APP2 = (ushort)0xB7;
                        public static ushort VK_OEM_1 = (ushort)0xBA;
                        public static ushort VK_OEM_PLUS = (ushort)0xBB;
                        public static ushort VK_OEM_COMMA = (ushort)0xBC;
                        public static ushort VK_OEM_MINUS = (ushort)0xBD;
                        public static ushort VK_OEM_PERIOD = (ushort)0xBE;
                        public static ushort VK_OEM_2 = (ushort)0xBF;
                        public static ushort VK_OEM_3 = (ushort)0xC0;
                        public static ushort VK_OEM_4 = (ushort)0xDB;
                        public static ushort VK_OEM_5 = (ushort)0xDC;
                        public static ushort VK_OEM_6 = (ushort)0xDD;
                        public static ushort VK_OEM_7 = (ushort)0xDE;
                        public static ushort VK_OEM_8 = (ushort)0xDF;
                        public static ushort VK_OEM_102 = (ushort)0xE2;
                        public static ushort VK_PROCESSKEY = (ushort)0xE5;
                        public static ushort VK_PACKET = (ushort)0xE7;
                        public static ushort VK_ATTN = (ushort)0xF6;
                        public static ushort VK_CRSEL = (ushort)0xF7;
                        public static ushort VK_EXSEL = (ushort)0xF8;
                        public static ushort VK_EREOF = (ushort)0xF9;
                        public static ushort VK_PLAY = (ushort)0xFA;
                        public static ushort VK_ZOOM = (ushort)0xFB;
                        public static ushort VK_NONAME = (ushort)0xFC;
                        public static ushort VK_PA1 = (ushort)0xFD;
                        public static ushort VK_OEM_CLEAR = (ushort)0xFE;
                        public static ushort S_LBUTTON = (ushort)MapVirtualKey(0x01, 0);
                        public static ushort S_RBUTTON = (ushort)MapVirtualKey(0x02, 0);
                        public static ushort S_CANCEL = (ushort)MapVirtualKey(0x03, 0);
                        public static ushort S_MBUTTON = (ushort)MapVirtualKey(0x04, 0);
                        public static ushort S_XBUTTON1 = (ushort)MapVirtualKey(0x05, 0);
                        public static ushort S_XBUTTON2 = (ushort)MapVirtualKey(0x06, 0);
                        public static ushort S_BACK = (ushort)MapVirtualKey(0x08, 0);
                        public static ushort S_Tab = (ushort)MapVirtualKey(0x09, 0);
                        public static ushort S_CLEAR = (ushort)MapVirtualKey(0x0C, 0);
                        public static ushort S_Return = (ushort)MapVirtualKey(0x0D, 0);
                        public static ushort S_SHIFT = (ushort)MapVirtualKey(0x10, 0);
                        public static ushort S_CONTROL = (ushort)MapVirtualKey(0x11, 0);
                        public static ushort S_MENU = (ushort)MapVirtualKey(0x12, 0);
                        public static ushort S_PAUSE = (ushort)MapVirtualKey(0x13, 0);
                        public static ushort S_CAPITAL = (ushort)MapVirtualKey(0x14, 0);
                        public static ushort S_KANA = (ushort)MapVirtualKey(0x15 , 0);
                        public static ushort S_HANGEUL = (ushort)MapVirtualKey(0x15, 0);
                        public static ushort S_HANGUL = (ushort)MapVirtualKey(0x15, 0);
                        public static ushort S_JUNJA = (ushort)MapVirtualKey(0x17, 0);
                        public static ushort S_FINAL = (ushort)MapVirtualKey(0x18, 0);
                        public static ushort S_HANJA = (ushort)MapVirtualKey(0x19, 0);
                        public static ushort S_KANJI = (ushort)MapVirtualKey(0x19, 0);
                        public static ushort S_Escape = (ushort)MapVirtualKey(0x1B, 0);
                        public static ushort S_CONVERT = (ushort)MapVirtualKey(0x1C, 0);
                        public static ushort S_NONCONVERT = (ushort)MapVirtualKey(0x1D, 0);
                        public static ushort S_ACCEPT = (ushort)MapVirtualKey(0x1E, 0);
                        public static ushort S_MODECHANGE = (ushort)MapVirtualKey(0x1F, 0);
                        public static ushort S_Space = (ushort)MapVirtualKey(0x20, 0);
                        public static ushort S_PRIOR = (ushort)MapVirtualKey(0x21, 0);
                        public static ushort S_NEXT = (ushort)MapVirtualKey(0x22, 0);
                        public static ushort S_END = (ushort)MapVirtualKey(0x23, 0);
                        public static ushort S_HOME = (ushort)MapVirtualKey(0x24, 0);
                        public static ushort S_LEFT = (ushort)MapVirtualKey(0x25, 0);
                        public static ushort S_UP = (ushort)MapVirtualKey(0x26, 0);
                        public static ushort S_RIGHT = (ushort)MapVirtualKey(0x27, 0);
                        public static ushort S_DOWN = (ushort)MapVirtualKey(0x28, 0);
                        public static ushort S_SELECT = (ushort)MapVirtualKey(0x29, 0);
                        public static ushort S_PRINT = (ushort)MapVirtualKey(0x2A, 0);
                        public static ushort S_EXECUTE = (ushort)MapVirtualKey(0x2B, 0);
                        public static ushort S_SNAPSHOT = (ushort)MapVirtualKey(0x2C, 0);
                        public static ushort S_INSERT = (ushort)MapVirtualKey(0x2D, 0);
                        public static ushort S_DELETE = (ushort)MapVirtualKey(0x2E, 0);
                        public static ushort S_HELP = (ushort)MapVirtualKey(0x2F, 0);
                        public static ushort S_APOSTROPHE = (ushort)MapVirtualKey(0xDE, 0);
                        public static ushort S_0 = (ushort)MapVirtualKey(0x30, 0);
                        public static ushort S_1 = (ushort)MapVirtualKey(0x31, 0);
                        public static ushort S_2 = (ushort)MapVirtualKey(0x32, 0);
                        public static ushort S_3 = (ushort)MapVirtualKey(0x33, 0);
                        public static ushort S_4 = (ushort)MapVirtualKey(0x34, 0);
                        public static ushort S_5 = (ushort)MapVirtualKey(0x35, 0);
                        public static ushort S_6 = (ushort)MapVirtualKey(0x36, 0);
                        public static ushort S_7 = (ushort)MapVirtualKey(0x37, 0);
                        public static ushort S_8 = (ushort)MapVirtualKey(0x38, 0);
                        public static ushort S_9 = (ushort)MapVirtualKey(0x39, 0);
                        public static ushort S_A = (ushort)MapVirtualKey(0x41, 0);
                        public static ushort S_B = (ushort)MapVirtualKey(0x42, 0);
                        public static ushort S_C = (ushort)MapVirtualKey(0x43, 0);
                        public static ushort S_D = (ushort)MapVirtualKey(0x44, 0);
                        public static ushort S_E = (ushort)MapVirtualKey(0x45, 0);
                        public static ushort S_F = (ushort)MapVirtualKey(0x46, 0);
                        public static ushort S_G = (ushort)MapVirtualKey(0x47, 0);
                        public static ushort S_H = (ushort)MapVirtualKey(0x48, 0);
                        public static ushort S_I = (ushort)MapVirtualKey(0x49, 0);
                        public static ushort S_J = (ushort)MapVirtualKey(0x4A, 0);
                        public static ushort S_K = (ushort)MapVirtualKey(0x4B, 0);
                        public static ushort S_L = (ushort)MapVirtualKey(0x4C, 0);
                        public static ushort S_M = (ushort)MapVirtualKey(0x4D, 0);
                        public static ushort S_N = (ushort)MapVirtualKey(0x4E, 0);
                        public static ushort S_O = (ushort)MapVirtualKey(0x4F, 0);
                        public static ushort S_P = (ushort)MapVirtualKey(0x50, 0);
                        public static ushort S_Q = (ushort)MapVirtualKey(0x51, 0);
                        public static ushort S_R = (ushort)MapVirtualKey(0x52, 0);
                        public static ushort S_S = (ushort)MapVirtualKey(0x53, 0);
                        public static ushort S_T = (ushort)MapVirtualKey(0x54, 0);
                        public static ushort S_U = (ushort)MapVirtualKey(0x55, 0);
                        public static ushort S_V = (ushort)MapVirtualKey(0x56, 0);
                        public static ushort S_W = (ushort)MapVirtualKey(0x57, 0);
                        public static ushort S_X = (ushort)MapVirtualKey(0x58, 0);
                        public static ushort S_Y = (ushort)MapVirtualKey(0x59, 0);
                        public static ushort S_Z = (ushort)MapVirtualKey(0x5A, 0);
                        public static ushort S_LWIN = (ushort)MapVirtualKey(0x5B, 0);
                        public static ushort S_RWIN = (ushort)MapVirtualKey(0x5C, 0);
                        public static ushort S_APPS = (ushort)MapVirtualKey(0x5D, 0);
                        public static ushort S_SLEEP = (ushort)MapVirtualKey(0x5F, 0);
                        public static ushort S_NUMPAD0 = (ushort)MapVirtualKey(0x60, 0);
                        public static ushort S_NUMPAD1 = (ushort)MapVirtualKey(0x61, 0);
                        public static ushort S_NUMPAD2 = (ushort)MapVirtualKey(0x62, 0);
                        public static ushort S_NUMPAD3 = (ushort)MapVirtualKey(0x63, 0);
                        public static ushort S_NUMPAD4 = (ushort)MapVirtualKey(0x64, 0);
                        public static ushort S_NUMPAD5 = (ushort)MapVirtualKey(0x65, 0);
                        public static ushort S_NUMPAD6 = (ushort)MapVirtualKey(0x66, 0);
                        public static ushort S_NUMPAD7 = (ushort)MapVirtualKey(0x67, 0);
                        public static ushort S_NUMPAD8 = (ushort)MapVirtualKey(0x68, 0);
                        public static ushort S_NUMPAD9 = (ushort)MapVirtualKey(0x69, 0);
                        public static ushort S_MULTIPLY = (ushort)MapVirtualKey(0x6A, 0);
                        public static ushort S_ADD = (ushort)MapVirtualKey(0x6B, 0);
                        public static ushort S_SEPARATOR = (ushort)MapVirtualKey(0x6C, 0);
                        public static ushort S_SUBTRACT = (ushort)MapVirtualKey(0x6D, 0);
                        public static ushort S_DECIMAL = (ushort)MapVirtualKey(0x6E, 0);
                        public static ushort S_DIVIDE = (ushort)MapVirtualKey(0x6F, 0);
                        public static ushort S_F1 = (ushort)MapVirtualKey(0x70, 0);
                        public static ushort S_F2 = (ushort)MapVirtualKey(0x71, 0);
                        public static ushort S_F3 = (ushort)MapVirtualKey(0x72, 0);
                        public static ushort S_F4 = (ushort)MapVirtualKey(0x73, 0);
                        public static ushort S_F5 = (ushort)MapVirtualKey(0x74, 0);
                        public static ushort S_F6 = (ushort)MapVirtualKey(0x75, 0);
                        public static ushort S_F7 = (ushort)MapVirtualKey(0x76, 0);
                        public static ushort S_F8 = (ushort)MapVirtualKey(0x77, 0);
                        public static ushort S_F9 = (ushort)MapVirtualKey(0x78, 0);
                        public static ushort S_F10 = (ushort)MapVirtualKey(0x79, 0);
                        public static ushort S_F11 = (ushort)MapVirtualKey(0x7A, 0);
                        public static ushort S_F12 = (ushort)MapVirtualKey(0x7B, 0);
                        public static ushort S_F13 = (ushort)MapVirtualKey(0x7C, 0);
                        public static ushort S_F14 = (ushort)MapVirtualKey(0x7D, 0);
                        public static ushort S_F15 = (ushort)MapVirtualKey(0x7E, 0);
                        public static ushort S_F16 = (ushort)MapVirtualKey(0x7F, 0);
                        public static ushort S_F17 = (ushort)MapVirtualKey(0x80, 0);
                        public static ushort S_F18 = (ushort)MapVirtualKey(0x81, 0);
                        public static ushort S_F19 = (ushort)MapVirtualKey(0x82, 0);
                        public static ushort S_F20 = (ushort)MapVirtualKey(0x83, 0);
                        public static ushort S_F21 = (ushort)MapVirtualKey(0x84, 0);
                        public static ushort S_F22 = (ushort)MapVirtualKey(0x85, 0);
                        public static ushort S_F23 = (ushort)MapVirtualKey(0x86, 0);
                        public static ushort S_F24 = (ushort)MapVirtualKey(0x87, 0);
                        public static ushort S_NUMLOCK = (ushort)MapVirtualKey(0x90, 0);
                        public static ushort S_SCROLL = (ushort)MapVirtualKey(0x91, 0);
                        public static ushort S_LeftShift = (ushort)MapVirtualKey(0xA0, 0);
                        public static ushort S_RightShift = (ushort)MapVirtualKey(0xA1, 0);
```

```
                        public static ushort S_LeftControl = (ushort)MapVirtualKey(0xA2,
0);
                        public static ushort S_RightControl = (ushort)MapVirtualKey(0xA3,
0);
                        public static ushort S_LMENU = (ushort)MapVirtualKey(0xA4, 0);
                        public static ushort S_RMENU = (ushort)MapVirtualKey(0xA5, 0);
                        public static ushort S_BROWSER_BACK = (ushort)MapVirtualKey(0xA6,
0);
                        public static ushort S_BROWSER_FORWARD =
(ushort)MapVirtualKey(0xA7, 0);
                        public static ushort S_BROWSER_REFRESH =
(ushort)MapVirtualKey(0xA8, 0);
                        public static ushort S_BROWSER_STOP = (ushort)MapVirtualKey(0xA9,
0);
                        public static ushort S_BROWSER_SEARCH = (ushort)MapVirtualKey(0xAA,
0);
                        public static ushort S_BROWSER_FAVORITES =
(ushort)MapVirtualKey(0xAB, 0);
                        public static ushort S_BROWSER_HOME = (ushort)MapVirtualKey(0xAC,
0);
                        public static ushort S_VOLUME_MUTE = (ushort)MapVirtualKey(0xAD,
0);
                        public static ushort S_VOLUME_DOWN = (ushort)MapVirtualKey(0xAE,
0);
                        public static ushort S_VOLUME_UP = (ushort)MapVirtualKey(0xAF, 0);
                        public static ushort S_MEDIA_NEXT_TRACK =
(ushort)MapVirtualKey(0xB0, 0);
                        public static ushort S_MEDIA_PREV_TRACK =
(ushort)MapVirtualKey(0xB1, 0);
                        public static ushort S_MEDIA_STOP = (ushort)MapVirtualKey(0xB2, 0);
                        public static ushort S_MEDIA_PLAY_PAUSE =
(ushort)MapVirtualKey(0xB3, 0);
                        public static ushort S_LAUNCH_MAIL = (ushort)MapVirtualKey(0xB4,
0);
                        public static ushort S_LAUNCH_MEDIA_SELECT =
(ushort)MapVirtualKey(0xB5, 0);
                        public static ushort S_LAUNCH_APP1 = (ushort)MapVirtualKey(0xB6,
0);
                        public static ushort S_LAUNCH_APP2 = (ushort)MapVirtualKey(0xB7,
0);
                        public static ushort S_OEM_1 = (ushort)MapVirtualKey(0xBA, 0);
                        public static ushort S_OEM_PLUS = (ushort)MapVirtualKey(0xBB, 0);
                        public static ushort S_OEM_COMMA = (ushort)MapVirtualKey(0xBC, 0);
                        public static ushort S_OEM_MINUS = (ushort)MapVirtualKey(0xBD, 0);
                        public static ushort S_OEM_PERIOD = (ushort)MapVirtualKey(0xBE, 0);
                        public static ushort S_OEM_2 = (ushort)MapVirtualKey(0xBF, 0);
                        public static ushort S_OEM_3 = (ushort)MapVirtualKey(0xC0, 0);
                        public static ushort S_OEM_4 = (ushort)MapVirtualKey(0xDB, 0);
                        public static ushort S_OEM_5 = (ushort)MapVirtualKey(0xDC, 0);
                        public static ushort S_OEM_6 = (ushort)MapVirtualKey(0xDD, 0);
                        public static ushort S_OEM_7 = (ushort)MapVirtualKey(0xDE, 0);
                        public static ushort S_OEM_8 = (ushort)MapVirtualKey(0xDF, 0);
                        public static ushort S_OEM_102 = (ushort)MapVirtualKey(0xE2, 0);
                        public static ushort S_PROCESSKEY = (ushort)MapVirtualKey(0xE5, 0);
                        public static ushort S_PACKET = (ushort)MapVirtualKey(0xE7, 0);
                        public static ushort S_ATTN = (ushort)MapVirtualKey(0xF6, 0);
                        public static ushort S_CRSEL = (ushort)MapVirtualKey(0xF7, 0);
                        public static ushort S_EXSEL = (ushort)MapVirtualKey(0xF8, 0);
                        public static ushort S_EREOF = (ushort)MapVirtualKey(0xF9, 0);
                        public static ushort S_PLAY = (ushort)MapVirtualKey(0xFA, 0);
                        public static ushort S_ZOOM = (ushort)MapVirtualKey(0xFB, 0);
                        public static ushort S_NONAME = (ushort)MapVirtualKey(0xFC, 0);
                        public static ushort S_PA1 = (ushort)MapVirtualKey(0xFD, 0);
                        public static ushort S_OEM_CLEAR = (ushort)MapVirtualKey(0xFE, 0);
                    }
                    public class Input
```

```
{
    private IntPtr context;
    public KeyboardFilterMode KeyboardFilterMode { get; set; }
    public MouseFilterMode MouseFilterMode { get; set; }
    public bool IsLoaded { get; set; }
    public Input()
    {
        context = IntPtr.Zero;
        KeyboardFilterMode = KeyboardFilterMode.None;
        MouseFilterMode = MouseFilterMode.None;
    }
    public bool Load()
    {
        context = InterceptionDriver.CreateContext();
        return true;
    }
    public void Unload()
    {
        InterceptionDriver.DestroyContext(context);
    }
    public void SendKey(Keys key, KeyState state, int keyboardId)
    {
        Stroke stroke = new Stroke();
        KeyStroke keyStroke = new KeyStroke();
        keyStroke.Code = key;
        keyStroke.State = state;
        stroke.Key = keyStroke;
        InterceptionDriver.Send(context, keyboardId, ref stroke, 1);
    }
    public void SendKey(Keys key, int keyboardId)
    {
        SendKey(key, KeyState.Down, keyboardId);
    }
    public void SendKeyF(Keys key, int keyboardId)
    {
        SendKey(key, KeyState.Up, keyboardId);
    }
    public void SendMouseEvent(MouseState state, int mouseId)
    {
        Stroke stroke = new Stroke();
        MouseStroke mouseStroke = new MouseStroke();
        mouseStroke.State = state;
        if (state == MouseState.ScrollUp)
        {
            mouseStroke.Rolling = 120;
        }
        else if (state == MouseState.ScrollDown)
        {
            mouseStroke.Rolling = -120;
        }
        stroke.Mouse = mouseStroke;
        InterceptionDriver.Send(context, mouseId, ref stroke, 1);
    }
    public void SendLeftClick(int mouseId)
    {
        SendMouseEvent(MouseState.LeftDown, mouseId);
    }
    public void SendRightClick(int mouseId)
    {
        SendMouseEvent(MouseState.RightDown, mouseId);
    }
    public void SendLeftClickF(int mouseId)
    {
        SendMouseEvent(MouseState.LeftUp, mouseId);
    }
    public void SendRightClickF(int mouseId)
    {
        SendMouseEvent(MouseState.RightUp, mouseId);
    }
    public void ScrollMouse(ScrollDirection direction, int mouseId)
    {
        switch (direction)
        {
            case ScrollDirection.Down:
                SendMouseEvent(MouseState.ScrollDown, mouseId);
                break;
            case ScrollDirection.Up:
                SendMouseEvent(MouseState.ScrollUp, mouseId);
                break;
        }
    }
    public void MoveMouseBy(int deltaX, int deltaY, int mouseId)
    {
        Stroke stroke = new Stroke();
        MouseStroke mouseStroke = new MouseStroke();
        mouseStroke.X = deltaX;
        mouseStroke.Y = deltaY;
        stroke.Mouse = mouseStroke;
        stroke.Mouse.Flags = MouseFlags.MoveRelative;
        InterceptionDriver.Send(context, mouseId, ref stroke, 1);
    }
    public void MoveMouseTo(int x, int y, int mouseId)
    {
        Stroke stroke = new Stroke();
        MouseStroke mouseStroke = new MouseStroke();
        mouseStroke.X = x;
        mouseStroke.Y = y;
        stroke.Mouse = mouseStroke;
        stroke.Mouse.Flags = MouseFlags.MoveAbsolute;
        InterceptionDriver.Send(context, mouseId, ref stroke, 1);
    }
}
[UnmanagedFunctionPointer(CallingConvention.Cdecl)]
public delegate int Predicate(int device);
[Flags]
public enum KeyState : ushort
{
    Down = 0x00,
    Up = 0x01,
    E0 = 0x02,
    E1 = 0x04,
    TermsrvSetLED = 0x08,
    TermsrvShadow = 0x10,
    TermsrvVKPacket = 0x20
}
[Flags]
public enum KeyboardFilterMode : ushort
{
    None = 0x0000,
    All = 0xFFFF,
    KeyDown = KeyState.Up,
    KeyUp = KeyState.Up << 1,
    KeyE0 = KeyState.E0 << 1,
    KeyE1 = KeyState.E1 << 1,
    KeyTermsrvSetLED = KeyState.TermsrvSetLED << 1,
    KeyTermsrvShadow = KeyState.TermsrvShadow << 1,
    KeyTermsrvVKPacket = KeyState.TermsrvVKPacket << 1
}
[Flags]
public enum MouseState : ushort
{
    LeftDown = 0x01,
    LeftUp = 0x02,
    RightDown = 0x04,
    RightUp = 0x08,
    MiddleDown = 0x10,
    MiddleUp = 0x20,
    LeftExtraDown = 0x40,
    LeftExtraUp = 0x80,
    RightExtraDown = 0x100,
    RightExtraUp = 0x200,
    ScrollVertical = 0x400,
    ScrollUp = 0x400,
    ScrollDown = 0x400,
    ScrollHorizontal = 0x800,
    ScrollLeft = 0x800,
    ScrollRight = 0x800,
}
[Flags]
public enum MouseFilterMode : ushort
{
    None = 0x0000,
    All = 0xFFFF,
    LeftDown = 0x01,
    LeftUp = 0x02,
    RightDown = 0x04,
    RightUp = 0x08,
    MiddleDown = 0x10,
    MiddleUp = 0x20,
    LeftExtraDown = 0x40,
    LeftExtraUp = 0x80,
    RightExtraDown = 0x100,
    RightExtraUp = 0x200,
    MouseWheelVertical = 0x400,
    MouseWheelHorizontal = 0x800,
    MouseMove = 0x1000,
}
[Flags]
public enum MouseFlags : ushort
{
    MoveRelative = 0x000,
    MoveAbsolute = 0x001,
    VirtualDesktop = 0x002,
    AttributesChanged = 0x004,
    MoveWithoutCoalescing = 0x008,
    TerminalServicesSourceShadow = 0x100
}
[StructLayout(LayoutKind.Sequential)]
public struct MouseStroke
{
    public MouseState State;
    public MouseFlags Flags;
    public Int16 Rolling;
    public Int32 X;
    public Int32 Y;
    public UInt16 Information;
}
[StructLayout(LayoutKind.Sequential)]
public struct KeyStroke
{
    public Keys Code;
    public KeyState State;
    public UInt32 Information;
}
[StructLayout(LayoutKind.Explicit)]
public struct Stroke
{
    [FieldOffset(0)]
```

```
                            public MouseStroke Mouse;
                            [FieldOffset(0)]
                            public KeyStroke Key;
                        }
                        public static class InterceptionDriver
                        {
                            [DllImport(""interception.dll"", EntryPoint =
""interception_create_context"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern IntPtr CreateContext();
                            [DllImport(""interception.dll"", EntryPoint =
""interception_destroy_context"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern void DestroyContext(IntPtr context);
                            [DllImport(""interception.dll"", EntryPoint =
""interception_get_precedence"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern void GetPrecedence(IntPtr context, Int32
device);
                            [DllImport(""interception.dll"", EntryPoint =
""interception_set_precedence"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern void SetPrecedence(IntPtr context, Int32
device, Int32 Precedence);
                            [DllImport(""interception.dll"", EntryPoint =
""interception_get_filter"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern void GetFilter(IntPtr context, Int32 device);
                            [DllImport(""interception.dll"", EntryPoint =
""interception_set_filter"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern void SetFilter(IntPtr context, Predicate
predicate, Int32 keyboardFilterMode);
                            [DllImport(""interception.dll"", EntryPoint =
""interception_wait"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern Int32 Wait(IntPtr context);
                            [DllImport(""interception.dll"", EntryPoint =
""interception_wait_with_timeout"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern Int32 WaitWithTimeout(IntPtr context, UInt64
milliseconds);
                            [DllImport(""interception.dll"", EntryPoint =
""interception_send"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern Int32 Send(IntPtr context, Int32 device, ref
Stroke stroke, UInt32 numStrokes);
                            [DllImport(""interception.dll"", EntryPoint =
""interception_receive"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern Int32 Receive(IntPtr context, Int32 device,
ref Stroke stroke, UInt32 numStrokes);
                            [DllImport(""interception.dll"", EntryPoint =
""interception_get_hardware_id"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern Int32 GetHardwareId(IntPtr context, Int32
device, String hardwareIdentifier, UInt32 sizeOfString);
                            [DllImport(""interception.dll"", EntryPoint =
""interception_is_invalid"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern Int32 IsInvalid(Int32 device);
                            [DllImport(""interception.dll"", EntryPoint =
""interception_is_keyboard"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern Int32 IsKeyboard(Int32 device);
                            [DllImport(""interception.dll"", EntryPoint =
""interception_is_mouse"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern Int32 IsMouse(Int32 device);
                        }
                        public class KeyPressedEventArgs : EventArgs
                        {
                            public Keys Key { get; set; }
                            public KeyState State { get; set; }
                            public bool Handled { get; set; }
                        }
                        public enum Keys : ushort
                        {
                            Escape = 1,
                            One = 2,
Two = 3,
Three = 4,
Four = 5,
Five = 6,
Six = 7,
Seven = 8,
Eight = 9,
Nine = 10,
Zero = 11,
DashUnderscore = 12,
PlusEquals = 13,
Backspace = 14,
Tab = 15,
A = 16,
Z = 17,
E = 18,
R = 19,
T = 20,
Y = 21,
U = 22,
I = 23,
O = 24,
P = 25,
OpenBracketBrace = 26,
CloseBracketBrace = 27,
Enter = 28,
Control = 29,
Q = 30,
S = 31,
D = 32,
F = 33,
G = 34,
H = 35,
J = 36,
K = 37,
L = 38,
SemicolonColon = 39,
SingleDoubleQuote = 40,
Tilde = 41,
LeftShift = 42,
BackslashPipe = 43,
W = 44,
X = 45,
C = 46,
V = 47,
B = 48,
N = 49,
M = 50,
CommaLeftArrow = 51,
PeriodRightArrow = 52,
ForwardSlashQuestionMark = 53,
RightShift = 54,
RightAlt = 56,
Space = 57,
CapsLock = 58,
F1 = 59,
F2 = 60,
F3 = 61,
F4 = 62,
F5 = 63,
F6 = 64,
F7 = 65,
F8 = 66,
F9 = 67,
F10 = 68,
F11 = 87,
                        F12 = 88,
                        Up = 72,
                        Down = 80,
                        Right = 77,
                        Left = 75,
                        Home = 71,
                        End = 79,
                        Delete = 83,
                        PageUp = 73,
                        PageDown = 81,
                        Insert = 82,
                        PrintScreen = 55,
                        NumLock = 69,
                        ScrollLock = 70,
                        Menu = 93,
                        WindowsKey = 91,
                        NumpadDivide = 53,
                        NumpadAsterisk = 55,
                        Numpad7 = 71,
                        Numpad8 = 72,
                        Numpad9 = 73,
                        Numpad4 = 75,
                        Numpad5 = 76,
                        Numpad6 = 77,
                        Numpad1 = 79,
                        Numpad2 = 80,
                        Numpad3 = 81,
                        Numpad0 = 82,
                        NumpadDelete = 83,
                        NumpadEnter = 28,
                        NumpadPlus = 78,
                        NumpadMinus = 74,
                    }
                    public class MousePressedEventArgs : EventArgs
                    {
                        public MouseState State { get; set; }
                        public bool Handled { get; set; }
                        public int X { get; set; }
                        public int Y { get; set; }
                        public short Rolling { get; set; }
                    }
                    public enum ScrollDirection
                    {
                        Down,
                        Up
                    }
                }
                ";
        }
        private void button4_Click(object sender, EventArgs e)
        {
            string confname = comboBox1.Text + ".txt";
            initConfigM(confname);
            initCode();
            finalcode = code.Replace("funct_driver", addedcode);
            System.CodeDom.Compiler.CompilerParameters parameters = new
System.CodeDom.Compiler.CompilerParameters();
            parameters.ReferencedAssemblies.Add("System.Windows.Forms.dll");
            parameters.ReferencedAssemblies.Add("System.Drawing.dll");
            Microsoft.CSharp.CSharpCodeProvider provider = new
Microsoft.CSharp.CSharpCodeProvider();
            System.CodeDom.Compiler.CompilerResults results =
provider.CompileAssemblyFromSource(parameters, finalcode);
            if (results.Errors.HasErrors)
            {
                StringBuilder sb = new StringBuilder();
                foreach (System.CodeDom.Compiler.CompilerError error in results.Errors)
                {
                    sb.AppendLine(String.Format("Error ({0}) : {1}", error.ErrorNumber,
error.ErrorText));
                }
                MessageBox.Show("mouse control :\n\r" + sb.ToString());
            }
            initConfigK(confname);
            initCode();
            finalcode = code.Replace("funct_driver", addedcode);
            parameters = new System.CodeDom.Compiler.CompilerParameters();
            parameters.ReferencedAssemblies.Add("System.Windows.Forms.dll");
            parameters.ReferencedAssemblies.Add("System.Drawing.dll");
            provider = new Microsoft.CSharp.CSharpCodeProvider();
            results = provider.CompileAssemblyFromSource(parameters, finalcode);
            if (results.Errors.HasErrors)
            {
                StringBuilder sb = new StringBuilder();
                foreach (System.CodeDom.Compiler.CompilerError error in results.Errors)
                {
                    sb.AppendLine(String.Format("Error ({0}) : {1}", error.ErrorNumber,
error.ErrorText));
                }
                MessageBox.Show("keyboard control :\n\r" + sb.ToString());
            }
            System.Diagnostics.Process.Start(confname);
        }
        private void initConfigM(string confname)
        {
            System.IO.StreamReader file = new System.IO.StreamReader(confname);
            string linesofcode = "";
            string emptylines = "";
            int i = 0;
            file.ReadLine();
            file.ReadLine();
            do
            {
                emptylines = file.ReadLine();
                if (emptylines != "")
                {
                    linesofcode += emptylines + " ";
                    i++;
                }
                else
                    break;
            }
            while (emptylines != "" | i == 0);
            file.Close();
            addedcode = linesofcode;
        }
        private void initConfigK(string confname)
        {
            System.IO.StreamReader file = new System.IO.StreamReader(confname);
            string linesofcode = "";
            string emptylines = "";
            int i = 0;
            file.ReadLine();
            file.ReadLine();
            do
            {
                emptylines = file.ReadLine();
                if (emptylines != "")
                {
                    linesofcode += emptylines + " ";
                    i++;
                }
                else
                    break;
            }
            while (emptylines != "" | i == 0);
            linesofcode = "";
            emptylines = "";
            i = 0;
            file.ReadLine();
            do
            {
                emptylines = file.ReadLine();
                if (emptylines != "")
                {
                    linesofcode += emptylines + " ";
                    i++;
                }
                else
                    break;
            }
            while (emptylines != "" | i == 0);
            file.Close();
            addedcode = linesofcode;
        }
    }
}
```

```
X-AIM.cmd:
XCOPY                              "C:\Users\Mic\Documents\micedition\WiimoteTheory\X-Aim.cfg"
"C:\Users\mic\AppData\Roaming\Cronus\Plugins\X-Aim.cfg" /y
```

```
X-Aim.cfg:
ExitMode=0
SelectedLayout=73
Dpi=8000
DirectInputEnabled=False
DirectInputGuid=00000000-0000-0000-0000-000000000000
DirectInputIsXInput=False
DirectInputDeviceName=
DirectInputRumble=False
DirectInputDeadzone=0
IgnoreController=True
DisableCMRumble=True
OverlayOpacity=75
; Don't change these settings unless you know what you are doing
SamplingCount=400
SamplingTime=20
ScrollWheelTiming=120
MouseUpdateFrequency=1
KeyboardUpdateFrequency=1
DirectInputUpdateFrequency=1
RumbleLimit=100
```

6. Mouse and keyboard driver

Interception is a physical mouse and keyboard driver available on github delivered by Francisco Lopez. It runs better for mouse control and where mouse event or sendinut don't run. You must launch install-interception /install with cmd dos prompt command, reboot your computer and have the DLL interception.dll under the folder where my programs are. Like with hidapi.dll given with my book with my Joycons codes. In order to have the right mouse id and keyboard id, to set in configuration files of my Wiimote/Joycons and Joycon(s) programs, you need to define it with the following program Windows Form and Class having 2 textboxes.

```
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 System.Threading;
using System.Runtime.InteropServices;
namespace InterceptionTest
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        Input input = new Input();
        private void Form1_Load(object sender, EventArgs e)
        {
            input.KeyboardFilterMode = KeyboardFilterMode.All;
            input.MouseFilterMode = MouseFilterMode.All;
            input.Load();
        }
        private void Form1_FormClosed(object sender, FormClosedEventArgs e)
        {
            input.Unload();
        }
    }
}
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 System.Threading;
using System.Runtime.InteropServices;
namespace InterceptionTest
{
    public class Input
    {
        private IntPtr context;
        public KeyboardFilterMode KeyboardFilterMode { get; set; }
        public MouseFilterMode MouseFilterMode { get; set; }
        public bool IsLoaded { get; set; }
        private Thread callbackThread;
        public event EventHandler<KeyPressedEventArgs> OnKeyPressed;
        public event EventHandler<MousePressedEventArgs> OnMousePressed;
        private int deviceId, mouseId, keyboardId;
        public Input()
        {
            context = IntPtr.Zero;
            KeyboardFilterMode = KeyboardFilterMode.None;
            MouseFilterMode = MouseFilterMode.None;
        }
        public bool Load()
        {
            if (IsLoaded)
                return false;
            context = InterceptionDriver.CreateContext();
            if (context != IntPtr.Zero)
            {
                callbackThread = new Thread(new ThreadStart(DriverCallback));
                callbackThread.Priority = ThreadPriority.Highest;
                callbackThread.IsBackground = true;
                callbackThread.Start();
                IsLoaded = true;
                return true;
            }
            else
            {
                IsLoaded = false;
                return false;
            }
        }
        public void Unload()
        {
            if (!IsLoaded)
                return;
            if (context != IntPtr.Zero)
            {
                callbackThread.Abort();
                InterceptionDriver.DestroyContext(context);
                IsLoaded = false;
            }
        }
        private void DriverCallback()
        {
            Form1 form = (Form1)Application.OpenForms["Form1"];
            InterceptionDriver.SetFilter(context, InterceptionDriver.IsKeyboard, 
(Int32)KeyboardFilterMode);
            InterceptionDriver.SetFilter(context, InterceptionDriver.IsMouse, 
(Int32)MouseFilterMode);
            Stroke stroke = new Stroke();
            while (InterceptionDriver.Receive(context, deviceId = 
InterceptionDriver.Wait(context), ref stroke, 1) > 0)
            {
                if (InterceptionDriver.IsMouse(deviceId) > 0)
                {
                    mouseId = deviceId;
                    form.textBox1.Text = mouseId.ToString();
                    if (OnMousePressed != null)
                    {
                        var args = new MousePressedEventArgs() { X = stroke.Mouse.X, Y 
= stroke.Mouse.Y, State = stroke.Mouse.State, Rolling = stroke.Mouse.Rolling };
                        OnMousePressed(this, args);
                        if (args.Handled)
                            continue;
                        stroke.Mouse.X = args.X;
                        stroke.Mouse.Y = args.Y;
                        stroke.Mouse.State = args.State;
                        stroke.Mouse.Rolling = args.Rolling;
                    }
                }
                if (InterceptionDriver.IsKeyboard(deviceId) > 0)
                {
                    keyboardId = deviceId;
                    form.textBox2.Text = keyboardId.ToString();
                    if (OnKeyPressed != null)
                    {
                        var args = new KeyPressedEventArgs() { Key = stroke.Key.Code,
State = stroke.Key.State };
                        OnKeyPressed(this, args);
                        if (args.Handled)
                            continue;
                        stroke.Key.Code = args.Key;
                        stroke.Key.State = args.State;
                    }
                }
                InterceptionDriver.Send(context, deviceId, ref stroke, 1);
            }
            Unload();
            throw new Exception("Interception.Receive() failed for an unknown reason.
The driver has been unloaded.");
        }
        public void SendKey(Keys key, KeyState state)
        {
            Stroke stroke = new Stroke();
            KeyStroke keyStroke = new KeyStroke();
            keyStroke.Code = key;
            keyStroke.State = state;
            stroke.Key = keyStroke;
            InterceptionDriver.Send(context, keyboardId, ref stroke, 1);
        }
        public void SendKey(Keys key)
        {
            SendKey(key, KeyState.Down);
        }
        public void SendKeyF(Keys key)
        {
            SendKey(key, KeyState.Up);
        }
        public void SendMouseEvent(MouseState state)
        {
            Stroke stroke = new Stroke();
            MouseStroke mouseStroke = new MouseStroke();
            mouseStroke.State = state;
            if (state == MouseState.ScrollUp)
            {
                mouseStroke.Rolling = 120;
            }
            else if (state == MouseState.ScrollDown)
            {
                mouseStroke.Rolling = -120;
            }
            stroke.Mouse = mouseStroke;
            InterceptionDriver.Send(context, mouseId, ref stroke, 1);
        }
        public void SendLeftClick()
        {
            SendMouseEvent(MouseState.LeftDown);
        }
        public void SendRightClick()
```

```
    {
        SendMouseEvent(MouseState.RightDown);
    }
    public void SendLeftClickF()
    {
        SendMouseEvent(MouseState.LeftUp);
    }
    public void SendRightClickF()
    {
        SendMouseEvent(MouseState.RightUp);
    }
    public void ScrollMouse(ScrollDirection direction)
    {
        switch (direction)
        {
            case ScrollDirection.Down:
                SendMouseEvent(MouseState.ScrollDown);
                break;
            case ScrollDirection.Up:
                SendMouseEvent(MouseState.ScrollUp);
                break;
        }
    }
    public void MoveMouseBy(int deltaX, int deltaY)
    {
        Stroke stroke = new Stroke();
        MouseStroke mouseStroke = new MouseStroke();
        mouseStroke.X = deltaX;
        mouseStroke.Y = deltaY;
        stroke.Mouse = mouseStroke;
        stroke.Mouse.Flags = MouseFlags.MoveRelative;
        InterceptionDriver.Send(context, mouseId, ref stroke, 1);
    }
    public void MoveMouseTo(int x, int y)
    {
        Stroke stroke = new Stroke();
        MouseStroke mouseStroke = new MouseStroke();
        mouseStroke.X = x;
        mouseStroke.Y = y;
        stroke.Mouse = mouseStroke;
        stroke.Mouse.Flags = MouseFlags.MoveAbsolute;
        InterceptionDriver.Send(context, mouseId, ref stroke, 1);
    }
}
[UnmanagedFunctionPointer(CallingConvention.Cdecl)]
public delegate int Predicate(int device);
[Flags]
public enum KeyState : ushort
{
    Down = 0x00,
    Up = 0x01,
    E0 = 0x02,
    E1 = 0x04,
    TermsrvSetLED = 0x08,
    TermsrvShadow = 0x10,
    TermsrvVKPacket = 0x20
}
[Flags]
public enum KeyboardFilterMode : ushort
{
    None = 0x0000,
    All = 0xFFFF,
    KeyDown = KeyState.Up,
    KeyUp = KeyState.Up << 1,
    KeyE0 = KeyState.E0 << 1,
    KeyE1 = KeyState.E1 << 1,
    KeyTermsrvSetLED = KeyState.TermsrvSetLED << 1,
    KeyTermsrvShadow = KeyState.TermsrvShadow << 1,
    KeyTermsrvVKPacket = KeyState.TermsrvVKPacket << 1
}
[Flags]
public enum MouseState : ushort
{
    LeftDown = 0x01,
    LeftUp = 0x02,
    RightDown = 0x04,
    RightUp = 0x08,
    MiddleDown = 0x10,
    MiddleUp = 0x20,
    LeftExtraDown = 0x40,
    LeftExtraUp = 0x80,
    RightExtraDown = 0x100,
    RightExtraUp = 0x200,
    ScrollVertical = 0x400,
    ScrollUp = 0x400,
    ScrollDown = 0x400,
    ScrollHorizontal = 0x800,
    ScrollLeft = 0x800,
    ScrollRight = 0x800,
}
[Flags]
public enum MouseFilterMode : ushort
{
    None = 0x0000,
    All = 0xFFFF,
    LeftDown = 0x01,
    LeftUp = 0x02,
    RightDown = 0x04,
    RightUp = 0x08,
    MiddleDown = 0x10,
    MiddleUp = 0x20,
    LeftExtraDown = 0x40,
    LeftExtraUp = 0x80,
    RightExtraDown = 0x100,
    RightExtraUp = 0x200,
    MouseWheelVertical = 0x400,
    MouseWheelHorizontal = 0x800,
    MouseMove = 0x1000,
}
[Flags]
public enum MouseFlags : ushort
{
    MoveRelative = 0x000,
    MoveAbsolute = 0x001,
    VirtualDesktop = 0x002,
    AttributesChanged = 0x004,
    MoveWithoutCoalescing = 0x008,
    TerminalServicesSourceShadow = 0x100
}
[StructLayout(LayoutKind.Sequential)]
public struct MouseStroke
{
    public MouseState State;
    public MouseFlags Flags;
```

```
        public Int16 Rolling;
        public Int32 X;
        public Int32 Y;
        public UInt16 Information;
    }
    [StructLayout(LayoutKind.Sequential)]
    public struct KeyStroke
    {
        public Keys Code;
        public KeyState State;
        public UInt32 Information;
    }
    [StructLayout(LayoutKind.Explicit)]
    public struct Stroke
    {
        [FieldOffset(0)]
        public MouseStroke Mouse;
        [FieldOffset(0)]
        public KeyStroke Key;
    }
    public static class InterceptionDriver
    {
        [DllImport("interception.dll", EntryPoint = "interception_create_context",
CallingConvention = CallingConvention.Cdecl)]
        public static extern IntPtr CreateContext();
        [DllImport("interception.dll", EntryPoint = "interception_destroy_context",
CallingConvention = CallingConvention.Cdecl)]
        public static extern void DestroyContext(IntPtr context);
        [DllImport("interception.dll", EntryPoint = "interception_get_precedence",
CallingConvention = CallingConvention.Cdecl)]
        public static extern void GetPrecedence(IntPtr context, Int32 device);
        [DllImport("interception.dll", EntryPoint = "interception_set_precedence",
CallingConvention = CallingConvention.Cdecl)]
        public static extern void SetPrecedence(IntPtr context, Int32 device, Int32
Precedence);
        [DllImport("interception.dll", EntryPoint = "interception_get_filter",
CallingConvention = CallingConvention.Cdecl)]
        public static extern void GetFilter(IntPtr context, Int32 device);
        [DllImport("interception.dll", EntryPoint = "interception_set_filter",
CallingConvention = CallingConvention.Cdecl)]
        public static extern void SetFilter(IntPtr context, Predicate predicate, Int32
keyboardFilterMode);
        [DllImport("interception.dll", EntryPoint = "interception_wait",
CallingConvention = CallingConvention.Cdecl)]
        public static extern Int32 Wait(IntPtr context);
        [DllImport("interception.dll", EntryPoint = "interception_wait_with_timeout",
CallingConvention = CallingConvention.Cdecl)]
        public static extern Int32 WaitWithTimeout(IntPtr context, UInt64
milliseconds);
        [DllImport("interception.dll", EntryPoint = "interception_send",
CallingConvention = CallingConvention.Cdecl)]
        public static extern Int32 Send(IntPtr context, Int32 device, ref Stroke
stroke, UInt32 numStrokes);
        [DllImport("interception.dll", EntryPoint = "interception_receive",
CallingConvention = CallingConvention.Cdecl)]
        public static extern Int32 Receive(IntPtr context, Int32 device, ref Stroke
stroke, UInt32 numStrokes);
        [DllImport("interception.dll", EntryPoint = "interception_get_hardware_id",
CallingConvention = CallingConvention.Cdecl)]
        public static extern Int32 GetHardwareId(IntPtr context, Int32 device, String
hardwareIdentifier, UInt32 sizeOfString);
```

```
        [DllImport("interception.dll", EntryPoint = "interception_is_invalid",
CallingConvention = CallingConvention.Cdecl)]
        public static extern Int32 IsInvalid(Int32 device);
        [DllImport("interception.dll", EntryPoint = "interception_is_keyboard",
CallingConvention = CallingConvention.Cdecl)]
        public static extern Int32 IsKeyboard(Int32 device);
        [DllImport("interception.dll", EntryPoint = "interception_is_mouse",
CallingConvention = CallingConvention.Cdecl)]
        public static extern Int32 IsMouse(Int32 device);
    }
    public class KeyPressedEventArgs : EventArgs
    {
        public Keys Key { get; set; }
        public KeyState State { get; set; }
        public bool Handled { get; set; }
    }
    public enum Keys : ushort
    {
        Escape = 1,
        One = 2,
        Two = 3,
        Three = 4,
        Four = 5,
        Five = 6,
        Six = 7,
        Seven = 8,
        Eight = 9,
        Nine = 10,
        Zero = 11,
        DashUnderscore = 12,
        PlusEquals = 13,
        Backspace = 14,
        Tab = 15,
        Q = 16,
        W = 17,
        E = 18,
        R = 19,
        T = 20,
        Y = 21,
        U = 22,
        I = 23,
        O = 24,
        P = 25,
        OpenBracketBrace = 26,
        CloseBracketBrace = 27,
        Enter = 28,
        Control = 29,
        A = 30,
        S = 31,
        D = 32,
        F = 33,
        G = 34,
        H = 35,
        J = 36,
        K = 37,
        L = 38,
        SemicolonColon = 39,
        SingleDoubleQuote = 40,
        Tilde = 41,
        LeftShift = 42,
        BackslashPipe = 43,
        Z = 44,
        X = 45,
        C = 46,
        V = 47,
        B = 48,
        N = 49,
        M = 50,
        CommaLeftArrow = 51,
        PeriodRightArrow = 52,
        ForwardSlashQuestionMark = 53,
        RightShift = 54,
        RightAlt = 56,
        Space = 57,
        CapsLock = 58,
        F1 = 59,
        F2 = 60,
        F3 = 61,
        F4 = 62,
        F5 = 63,
        F6 = 64,
        F7 = 65,
        F8 = 66,
        F9 = 67,
        F10 = 68,
        F11 = 87,
        F12 = 88,
        Up = 72,
        Down = 80,
        Right = 77,
        Left = 75,
        Home = 71,
        End = 79,
        Delete = 83,
        PageUp = 73,
        PageDown = 81,
        Insert = 82,
        PrintScreen = 55,
        NumLock = 69,
        ScrollLock = 70,
        Menu = 93,
        WindowsKey = 91,
        NumpadDivide = 53,
        NumpadAsterisk = 55,
        Numpad7 = 71,
        Numpad8 = 72,
        Numpad9 = 73,
        Numpad4 = 75,
        Numpad5 = 76,
        Numpad6 = 77,
        Numpad1 = 79,
        Numpad2 = 80,
        Numpad3 = 81,
        Numpad0 = 82,
        NumpadDelete = 83,
        NumpadEnter = 28,
        NumpadPlus = 78,
        NumpadMinus = 74,
    }
    public class MousePressedEventArgs : EventArgs
    {
        public MouseState State { get; set; }
        public bool Handled { get; set; }
        public int X { get; set; }
```

```
        public int Y { get; set; }
        public short Rolling { get; set; }
    }
    public enum ScrollDirection
    {
        Down,
        Up
    }
}
```

6. Input emulator

It’s possible to control keyboard and mouse settings with a text file. The following code give the example.

```
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 Microsoft.CSharp;
using System.CodeDom;
using System.Reflection;
namespace StringToCode
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            string code = @"
            using System;
            using System.Runtime.InteropServices;
            namespace StringToCode
            {
                public class FooClass
                {

                    Input input = new Input();
                    public static bool[] _Valuechanged = new bool[30], _valuechanged = new
bool[30];
                    public bool this[int i]
                    {
                        get { return _valuechanged[i]; }
                        set
                        {
                            if (_valuechanged[i] != value)
                                _Valuechanged[i] = true;
                            else
                                _Valuechanged[i] = false;
                            _valuechanged[i] = value;
                        }
                    }
                    public void Selection()
                    {
                        this[3] = GetAsyncKeyState(System.Windows.Forms.Keys.Decimal);
```

```
                                if (_Valuechanged[3] & this[3])
                                    LeftClick();
                                if (_Valuechanged[3] & !this[3])
                                    LeftClickF();
                            }
                            public void Main(double x, double y)
                            {
                                Selection();
                                //funct_xy
                                //funct_mouse
                                funct_driver
                            }
                            public void Open(double x, double y)
                            {
                                input.KeyboardFilterMode = KeyboardFilterMode.All;
                                input.MouseFilterMode = MouseFilterMode.All;
                                input.Load();
                            }
                            public void Close(double x, double y)
                            {
                                input.Unload();
                            }
                            [DllImport(""user32.dll"")]
                            public static extern bool GetAsyncKeyState(System.Windows.Forms.Keys
vKey);
                            [DllImport(""user32.dll"")]
                            public static extern void SetCursorPos(int X, int Y);
                            [DllImport(""InputSending.dll"", EntryPoint = ""MoveMouseTo"",
CallingConvention = CallingConvention.Cdecl)]
                            public static extern void MoveMouseTo(int x, int y);
                            [DllImport(""InputSending.dll"", EntryPoint = ""MoveMouseBy"",
CallingConvention = CallingConvention.Cdecl)]
                            public static extern void MoveMouseBy(int x, int y);
                            [DllImport(""InputSending.dll"", EntryPoint = ""SendKey"",
CallingConvention = CallingConvention.Cdecl)]
                            public static extern void SendKey(UInt16 bVk, UInt16 bScan);
                            [DllImport(""InputSending.dll"", EntryPoint = ""SendKeyF"",
CallingConvention = CallingConvention.Cdecl)]
                            public static extern void SendKeyF(UInt16 bVk, UInt16 bScan);
                            [DllImport(""InputSending.dll"", EntryPoint = ""SendKeyArrows"",
CallingConvention = CallingConvention.Cdecl)]
                            public static extern void SendKeyArrows(UInt16 bVk, UInt16 bScan);
                            [DllImport(""InputSending.dll"", EntryPoint = ""SendKeyArrowsF"",
CallingConvention = CallingConvention.Cdecl)]
                            public static extern void SendKeyArrowsF(UInt16 bVk, UInt16 bScan);
                            [DllImport(""InputSending.dll"", EntryPoint =
""SendMouseEventButtonLeft"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern void SendMouseEventButtonLeft();
                            [DllImport(""InputSending.dll"", EntryPoint =
""SendMouseEventButtonLeftF"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern void SendMouseEventButtonLeftF();
                            [DllImport(""InputSending.dll"", EntryPoint =
""SendMouseEventButtonRight"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern void SendMouseEventButtonRight();
                            [DllImport(""InputSending.dll"", EntryPoint =
""SendMouseEventButtonRightF"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern void SendMouseEventButtonRightF();
                            [DllImport(""InputSending.dll"", EntryPoint =
""SendMouseEventButtonMiddle"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern void SendMouseEventButtonMiddle();
                            [DllImport(""InputSending.dll"", EntryPoint =
""SendMouseEventButtonMiddleF"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern void SendMouseEventButtonMiddleF();
                            [DllImport(""InputSending.dll"", EntryPoint =
""SendMouseEventButtonWheelUp"", CallingConvention = CallingConvention.Cdecl)]
                            public static extern void SendMouseEventButtonWheelUp();
```

```
                    [DllImport(""InputSending.dll"", EntryPoint =
""SendMouseEventButtonWheelDown"", CallingConvention = CallingConvention.Cdecl)]
                    public static extern void SendMouseEventButtonWheelDown();
                    [DllImport(""SendInputLibrary.dll"", EntryPoint = ""SimulateKeyDown"",
CallingConvention = CallingConvention.Cdecl)]
                    public static extern void SimulateKeyDown(UInt16 keyCode, UInt16
bScan);
                    [DllImport(""SendInputLibrary.dll"", EntryPoint = ""SimulateKeyUp"",
CallingConvention = CallingConvention.Cdecl)]
                    public static extern void SimulateKeyUp(UInt16 keyCode, UInt16 bScan);
                    [DllImport(""SendInputLibrary.dll"", EntryPoint =
""SimulateKeyDownArrows"", CallingConvention = CallingConvention.Cdecl)]
                    public static extern void SimulateKeyDownArrows(UInt16 keyCode, UInt16
bScan);
                    [DllImport(""SendInputLibrary.dll"", EntryPoint =
""SimulateKeyUpArrows"", CallingConvention = CallingConvention.Cdecl)]
                    public static extern void SimulateKeyUpArrows(UInt16 keyCode, UInt16
bScan);
                    [DllImport(""SendInputLibrary.dll"", EntryPoint = ""MouseMW3"",
CallingConvention = CallingConvention.Cdecl)]
                    public static extern void MouseMW3(int x, int y);
                    [DllImport(""SendInputLibrary.dll"", EntryPoint = ""MouseBrink"",
CallingConvention = CallingConvention.Cdecl)]
                    public static extern void MouseBrink(int x, int y);
                    [DllImport(""SendInputLibrary.dll"", EntryPoint = ""LeftClick"",
CallingConvention = CallingConvention.Cdecl)]
                    public static extern void LeftClick();
                    [DllImport(""SendInputLibrary.dll"", EntryPoint = ""LeftClickF"",
CallingConvention = CallingConvention.Cdecl)]
                    public static extern void LeftClickF();
                    [DllImport(""SendInputLibrary.dll"", EntryPoint = ""RightClick"",
CallingConvention = CallingConvention.Cdecl)]
                    public static extern void RightClick();
                    [DllImport(""SendInputLibrary.dll"", EntryPoint = ""RightClickF"",
CallingConvention = CallingConvention.Cdecl)]
                    public static extern void RightClickF();
                    [DllImport(""SendInputLibrary.dll"", EntryPoint = ""MiddleClick"",
CallingConvention = CallingConvention.Cdecl)]
                    public static extern void MiddleClick();
                    [DllImport(""SendInputLibrary.dll"", EntryPoint = ""MiddleClickF"",
CallingConvention = CallingConvention.Cdecl)]
                    public static extern void MiddleClickF();
                    [DllImport(""SendInputLibrary.dll"", EntryPoint = ""WheelDownF"",
CallingConvention = CallingConvention.Cdecl)]
                    public static extern void WheelDownF();
                    [DllImport(""SendInputLibrary.dll"", EntryPoint = ""WheelUpF"",
CallingConvention = CallingConvention.Cdecl)]
                    public static extern void WheelUpF();
                    [DllImport(""system32/user32.dll"")]
                    public static extern uint MapVirtualKey(uint uCode, uint uMapType);
                    public static ushort VK_LBUTTON = (ushort)0x01;
                    public static ushort VK_RBUTTON = (ushort)0x02;
                    public static ushort VK_CANCEL = (ushort)0x03;
                    public static ushort VK_MBUTTON = (ushort)0x04;
                    public static ushort VK_XBUTTON1 = (ushort)0x05;
                    public static ushort VK_XBUTTON2 = (ushort)0x06;
                    public static ushort VK_BACK = (ushort)0x08;
                    public static ushort VK_Tab = (ushort)0x09;
                    public static ushort VK_CLEAR = (ushort)0x0C;
                    public static ushort VK_Return = (ushort)0x0D;
                    public static ushort VK_SHIFT = (ushort)0x10;
                    public static ushort VK_CONTROL = (ushort)0x11;
                    public static ushort VK_MENU = (ushort)0x12;
                    public static ushort VK_PAUSE = (ushort)0x13;
                    public static ushort VK_CAPITAL = (ushort)0x14;
                    public static ushort VK_KANA = (ushort)0x15;
                    public static ushort VK_HANGEUL = (ushort)0x15;
                    public static ushort VK_HANGUL = (ushort)0x15;
                    public static ushort VK_JUNJA = (ushort)0x17;
                    public static ushort VK_FINAL = (ushort)0x18;
                    public static ushort VK_HANJA = (ushort)0x19;
                    public static ushort VK_KANJI = (ushort)0x19;
                    public static ushort VK_Escape = (ushort)0x1B;
                    public static ushort VK_CONVERT = (ushort)0x1C;
                    public static ushort VK_NONCONVERT = (ushort)0x1D;
                    public static ushort VK_ACCEPT = (ushort)0x1E;
                    public static ushort VK_MODECHANGE = (ushort)0x1F;
                    public static ushort VK_Space = (ushort)0x20;
                    public static ushort VK_PRIOR = (ushort)0x21;
                    public static ushort VK_NEXT = (ushort)0x22;
                    public static ushort VK_END = (ushort)0x23;
                    public static ushort VK_HOME = (ushort)0x24;
                    public static ushort VK_LEFT = (ushort)0x25;
                    public static ushort VK_UP = (ushort)0x26;
                    public static ushort VK_RIGHT = (ushort)0x27;
                    public static ushort VK_DOWN = (ushort)0x28;
                    public static ushort VK_SELECT = (ushort)0x29;
                    public static ushort VK_PRINT = (ushort)0x2A;
                    public static ushort VK_EXECUTE = (ushort)0x2B;
                    public static ushort VK_SNAPSHOT = (ushort)0x2C;
                    public static ushort VK_INSERT = (ushort)0x2D;
                    public static ushort VK_DELETE = (ushort)0x2E;
                    public static ushort VK_HELP = (ushort)0x2F;
                    public static ushort VK_APOSTROPHE = (ushort)0xDE;
                    public static ushort VK_0 = (ushort)0x30;
                    public static ushort VK_1 = (ushort)0x31;
                    public static ushort VK_2 = (ushort)0x32;
                    public static ushort VK_3 = (ushort)0x33;
                    public static ushort VK_4 = (ushort)0x34;
                    public static ushort VK_5 = (ushort)0x35;
                    public static ushort VK_6 = (ushort)0x36;
                    public static ushort VK_7 = (ushort)0x37;
                    public static ushort VK_8 = (ushort)0x38;
                    public static ushort VK_9 = (ushort)0x39;
                    public static ushort VK_A = (ushort)0x41;
                    public static ushort VK_B = (ushort)0x42;
                    public static ushort VK_C = (ushort)0x43;
                    public static ushort VK_D = (ushort)0x44;
                    public static ushort VK_E = (ushort)0x45;
                    public static ushort VK_F = (ushort)0x46;
                    public static ushort VK_G = (ushort)0x47;
                    public static ushort VK_H = (ushort)0x48;
                    public static ushort VK_I = (ushort)0x49;
                    public static ushort VK_J = (ushort)0x4A;
                    public static ushort VK_K = (ushort)0x4B;
                    public static ushort VK_L = (ushort)0x4C;
                    public static ushort VK_M = (ushort)0x4D;
                    public static ushort VK_N = (ushort)0x4E;
                    public static ushort VK_O = (ushort)0x4F;
                    public static ushort VK_P = (ushort)0x50;
                    public static ushort VK_Q = (ushort)0x51;
                    public static ushort VK_R = (ushort)0x52;
                    public static ushort VK_S = (ushort)0x53;
                    public static ushort VK_T = (ushort)0x54;
                    public static ushort VK_U = (ushort)0x55;
                    public static ushort VK_V = (ushort)0x56;
                    public static ushort VK_W = (ushort)0x57;
                    public static ushort VK_X = (ushort)0x58;
                    public static ushort VK_Y = (ushort)0x59;
                    public static ushort VK_Z = (ushort)0x5A;
                    public static ushort VK_LWIN = (ushort)0x5B;
                    public static ushort VK_RWIN = (ushort)0x5C;
                    public static ushort VK_APPS = (ushort)0x5D;
                    public static ushort VK_SLEEP = (ushort)0x5F;
                    public static ushort VK_NUMPAD0 = (ushort)0x60;
                    public static ushort VK_NUMPAD1 = (ushort)0x61;
                    public static ushort VK_NUMPAD2 = (ushort)0x62;
                    public static ushort VK_NUMPAD3 = (ushort)0x63;
                    public static ushort VK_NUMPAD4 = (ushort)0x64;
                    public static ushort VK_NUMPAD5 = (ushort)0x65;
                    public static ushort VK_NUMPAD6 = (ushort)0x66;
                    public static ushort VK_NUMPAD7 = (ushort)0x67;
                    public static ushort VK_NUMPAD8 = (ushort)0x68;
                    public static ushort VK_NUMPAD9 = (ushort)0x69;
                    public static ushort VK_MULTIPLY = (ushort)0x6A;
                    public static ushort VK_ADD = (ushort)0x6B;
                    public static ushort VK_SEPARATOR = (ushort)0x6C;
                    public static ushort VK_SUBTRACT = (ushort)0x6D;
                    public static ushort VK_DECIMAL = (ushort)0x6E;
                    public static ushort VK_DIVIDE = (ushort)0x6F;
                    public static ushort VK_F1 = (ushort)0x70;
                    public static ushort VK_F2 = (ushort)0x71;
                    public static ushort VK_F3 = (ushort)0x72;
                    public static ushort VK_F4 = (ushort)0x73;
                    public static ushort VK_F5 = (ushort)0x74;
                    public static ushort VK_F6 = (ushort)0x75;
                    public static ushort VK_F7 = (ushort)0x76;
                    public static ushort VK_F8 = (ushort)0x77;
                    public static ushort VK_F9 = (ushort)0x78;
                    public static ushort VK_F10 = (ushort)0x79;
                    public static ushort VK_F11 = (ushort)0x7A;
                    public static ushort VK_F12 = (ushort)0x7B;
                    public static ushort VK_F13 = (ushort)0x7C;
                    public static ushort VK_F14 = (ushort)0x7D;
                    public static ushort VK_F15 = (ushort)0x7E;
                    public static ushort VK_F16 = (ushort)0x7F;
                    public static ushort VK_F17 = (ushort)0x80;
                    public static ushort VK_F18 = (ushort)0x81;
                    public static ushort VK_F19 = (ushort)0x82;
                    public static ushort VK_F20 = (ushort)0x83;
                    public static ushort VK_F21 = (ushort)0x84;
                    public static ushort VK_F22 = (ushort)0x85;
                    public static ushort VK_F23 = (ushort)0x86;
                    public static ushort VK_F24 = (ushort)0x87;
                    public static ushort VK_NUMLOCK = (ushort)0x90;
                    public static ushort VK_SCROLL = (ushort)0x91;
                    public static ushort VK_LeftShift = (ushort)0xA0;
                    public static ushort VK_RightShift = (ushort)0xA1;
                    public static ushort VK_LeftControl = (ushort)0xA2;
                    public static ushort VK_RightControl = (ushort)0xA3;
                    public static ushort VK_LMENU = (ushort)0xA4;
                    public static ushort VK_RMENU = (ushort)0xA5;
                    public static ushort VK_BROWSER_BACK = (ushort)0xA6;
                    public static ushort VK_BROWSER_FORWARD = (ushort)0xA7;
                    public static ushort VK_BROWSER_REFRESH = (ushort)0xA8;
                    public static ushort VK_BROWSER_STOP = (ushort)0xA9;
                    public static ushort VK_BROWSER_SEARCH = (ushort)0xAA;
                    public static ushort VK_BROWSER_FAVORITES = (ushort)0xAB;
                    public static ushort VK_BROWSER_HOME = (ushort)0xAC;
                    public static ushort VK_VOLUME_MUTE = (ushort)0xAD;
                    public static ushort VK_VOLUME_DOWN = (ushort)0xAE;
                    public static ushort VK_VOLUME_UP = (ushort)0xAF;
                    public static ushort VK_MEDIA_NEXT_TRACK = (ushort)0xB0;
                    public static ushort VK_MEDIA_PREV_TRACK = (ushort)0xB1;
                    public static ushort VK_MEDIA_STOP = (ushort)0xB2;
                    public static ushort VK_MEDIA_PLAY_PAUSE = (ushort)0xB3;
                    public static ushort VK_LAUNCH_MAIL = (ushort)0xB4;
                    public static ushort VK_LAUNCH_MEDIA_SELECT = (ushort)0xB5;
                    public static ushort VK_LAUNCH_APP1 = (ushort)0xB6;
                    public static ushort VK_LAUNCH_APP2 = (ushort)0xB7;
                    public static ushort VK_OEM_1 = (ushort)0xBA;
                    public static ushort VK_OEM_PLUS = (ushort)0xBB;
                    public static ushort VK_OEM_COMMA = (ushort)0xBC;
                    public static ushort VK_OEM_MINUS = (ushort)0xBD;
                    public static ushort VK_OEM_PERIOD = (ushort)0xBE;
                    public static ushort VK_OEM_2 = (ushort)0xBF;
                    public static ushort VK_OEM_3 = (ushort)0xC0;
                    public static ushort VK_OEM_4 = (ushort)0xDB;
                    public static ushort VK_OEM_5 = (ushort)0xDC;
                    public static ushort VK_OEM_6 = (ushort)0xDD;
                    public static ushort VK_OEM_7 = (ushort)0xDE;
                    public static ushort VK_OEM_8 = (ushort)0xDF;
                    public static ushort VK_OEM_102 = (ushort)0xE2;
                    public static ushort VK_PROCESSKEY = (ushort)0xE5;
                    public static ushort VK_PACKET = (ushort)0xE7;
                    public static ushort VK_ATTN = (ushort)0xF6;
                    public static ushort VK_CRSEL = (ushort)0xF7;
                    public static ushort VK_EXSEL = (ushort)0xF8;
                    public static ushort VK_EREOF = (ushort)0xF9;
                    public static ushort VK_PLAY = (ushort)0xFA;
                    public static ushort VK_ZOOM = (ushort)0xFB;
                    public static ushort VK_NONAME = (ushort)0xFC;
                    public static ushort VK_PA1 = (ushort)0xFD;
                    public static ushort VK_OEM_CLEAR = (ushort)0xFE;
                    public static ushort S_LBUTTON = (ushort)MapVirtualKey(0x01, 0);
                    public static ushort S_RBUTTON = (ushort)MapVirtualKey(0x02, 0);
                    public static ushort S_CANCEL = (ushort)MapVirtualKey(0x03, 0);
                    public static ushort S_MBUTTON = (ushort)MapVirtualKey(0x04, 0);
                    public static ushort S_XBUTTON1 = (ushort)MapVirtualKey(0x05, 0);
                    public static ushort S_XBUTTON2 = (ushort)MapVirtualKey(0x06, 0);
                    public static ushort S_BACK = (ushort)MapVirtualKey(0x08, 0);
                    public static ushort S_Tab = (ushort)MapVirtualKey(0x09, 0);
                    public static ushort S_CLEAR = (ushort)MapVirtualKey(0x0C, 0);
                    public static ushort S_Return = (ushort)MapVirtualKey(0x0D, 0);
                    public static ushort S_SHIFT = (ushort)MapVirtualKey(0x10, 0);
                    public static ushort S_CONTROL = (ushort)MapVirtualKey(0x11, 0);
                    public static ushort S_MENU = (ushort)MapVirtualKey(0x12, 0);
                    public static ushort S_PAUSE = (ushort)MapVirtualKey(0x13, 0);
                    public static ushort S_CAPITAL = (ushort)MapVirtualKey(0x14, 0);
                    public static ushort S_KANA = (ushort)MapVirtualKey(0x15, 0);
                    public static ushort S_HANGEUL = (ushort)MapVirtualKey(0x15, 0);
                    public static ushort S_HANGUL = (ushort)MapVirtualKey(0x15, 0);
                    public static ushort S_JUNJA = (ushort)MapVirtualKey(0x17, 0);
                    public static ushort S_FINAL = (ushort)MapVirtualKey(0x18, 0);
                    public static ushort S_HANJA = (ushort)MapVirtualKey(0x19, 0);
                    public static ushort S_KANJI = (ushort)MapVirtualKey(0x19, 0);
                    public static ushort S_Escape = (ushort)MapVirtualKey(0x1B, 0);
                    public static ushort S_CONVERT = (ushort)MapVirtualKey(0x1C, 0);
                    public static ushort S_NONCONVERT = (ushort)MapVirtualKey(0x1D, 0);
                    public static ushort S_ACCEPT = (ushort)MapVirtualKey(0x1E, 0);
                    public static ushort S_MODECHANGE = (ushort)MapVirtualKey(0x1F, 0);
                    public static ushort S_Space = (ushort)MapVirtualKey(0x20, 0);
                    public static ushort S_PRIOR = (ushort)MapVirtualKey(0x21, 0);
                    public static ushort S_NEXT = (ushort)MapVirtualKey(0x22, 0);
                    public static ushort S_END = (ushort)MapVirtualKey(0x23, 0);
                    public static ushort S_HOME = (ushort)MapVirtualKey(0x24, 0);
                    public static ushort S_LEFT = (ushort)MapVirtualKey(0x25, 0);
                    public static ushort S_UP = (ushort)MapVirtualKey(0x26, 0);
                    public static ushort S_RIGHT = (ushort)MapVirtualKey(0x27, 0);
                    public static ushort S_DOWN = (ushort)MapVirtualKey(0x28, 0);
                    public static ushort S_SELECT = (ushort)MapVirtualKey(0x29, 0);
                    public static ushort S_PRINT = (ushort)MapVirtualKey(0x2A, 0);
                    public static ushort S_EXECUTE = (ushort)MapVirtualKey(0x2B, 0);
                    public static ushort S_SNAPSHOT = (ushort)MapVirtualKey(0x2C, 0);
                    public static ushort S_INSERT = (ushort)MapVirtualKey(0x2D, 0);
                    public static ushort S_DELETE = (ushort)MapVirtualKey(0x2E, 0);
                    public static ushort S_HELP = (ushort)MapVirtualKey(0x2F, 0);
                    public static ushort S_APOSTROPHE = (ushort)MapVirtualKey(0xDE, 0);
                    public static ushort S_0 = (ushort)MapVirtualKey(0x30, 0);
                    public static ushort S_1 = (ushort)MapVirtualKey(0x31, 0);
                    public static ushort S_2 = (ushort)MapVirtualKey(0x32, 0);
                    public static ushort S_3 = (ushort)MapVirtualKey(0x33, 0);
                    public static ushort S_4 = (ushort)MapVirtualKey(0x34, 0);
                    public static ushort S_5 = (ushort)MapVirtualKey(0x35, 0);
                    public static ushort S_6 = (ushort)MapVirtualKey(0x36, 0);
                    public static ushort S_7 = (ushort)MapVirtualKey(0x37, 0);
                    public static ushort S_8 = (ushort)MapVirtualKey(0x38, 0);
                    public static ushort S_9 = (ushort)MapVirtualKey(0x39, 0);
                    public static ushort S_A = (ushort)MapVirtualKey(0x41, 0);
                    public static ushort S_B = (ushort)MapVirtualKey(0x42, 0);
                    public static ushort S_C = (ushort)MapVirtualKey(0x43, 0);
                    public static ushort S_D = (ushort)MapVirtualKey(0x44, 0);
                    public static ushort S_E = (ushort)MapVirtualKey(0x45, 0);
                    public static ushort S_F = (ushort)MapVirtualKey(0x46, 0);
                    public static ushort S_G = (ushort)MapVirtualKey(0x47, 0);
                    public static ushort S_H = (ushort)MapVirtualKey(0x48, 0);
                    public static ushort S_I = (ushort)MapVirtualKey(0x49, 0);
                    public static ushort S_J = (ushort)MapVirtualKey(0x4A, 0);
                    public static ushort S_K = (ushort)MapVirtualKey(0x4B, 0);
                    public static ushort S_L = (ushort)MapVirtualKey(0x4C, 0);
                    public static ushort S_M = (ushort)MapVirtualKey(0x4D, 0);
                    public static ushort S_N = (ushort)MapVirtualKey(0x4E, 0);
                    public static ushort S_O = (ushort)MapVirtualKey(0x4F, 0);
                    public static ushort S_P = (ushort)MapVirtualKey(0x50, 0);
                    public static ushort S_Q = (ushort)MapVirtualKey(0x51, 0);
                    public static ushort S_R = (ushort)MapVirtualKey(0x52, 0);
                    public static ushort S_S = (ushort)MapVirtualKey(0x53, 0);
                    public static ushort S_T = (ushort)MapVirtualKey(0x54, 0);
                    public static ushort S_U = (ushort)MapVirtualKey(0x55, 0);
                    public static ushort S_V = (ushort)MapVirtualKey(0x56, 0);
                    public static ushort S_W = (ushort)MapVirtualKey(0x57, 0);
                    public static ushort S_X = (ushort)MapVirtualKey(0x58, 0);
                    public static ushort S_Y = (ushort)MapVirtualKey(0x59, 0);
                    public static ushort S_Z = (ushort)MapVirtualKey(0x5A, 0);
                    public static ushort S_LWIN = (ushort)MapVirtualKey(0x5B, 0);
                    public static ushort S_RWIN = (ushort)MapVirtualKey(0x5C, 0);
                    public static ushort S_APPS = (ushort)MapVirtualKey(0x5D, 0);
                    public static ushort S_SLEEP = (ushort)MapVirtualKey(0x5F, 0);
                    public static ushort S_NUMPAD0 = (ushort)MapVirtualKey(0x60, 0);
                    public static ushort S_NUMPAD1 = (ushort)MapVirtualKey(0x61, 0);
                    public static ushort S_NUMPAD2 = (ushort)MapVirtualKey(0x62, 0);
                    public static ushort S_NUMPAD3 = (ushort)MapVirtualKey(0x63, 0);
                    public static ushort S_NUMPAD4 = (ushort)MapVirtualKey(0x64, 0);
                    public static ushort S_NUMPAD5 = (ushort)MapVirtualKey(0x65, 0);
                    public static ushort S_NUMPAD6 = (ushort)MapVirtualKey(0x66, 0);
                    public static ushort S_NUMPAD7 = (ushort)MapVirtualKey(0x67, 0);
                    public static ushort S_NUMPAD8 = (ushort)MapVirtualKey(0x68, 0);
                    public static ushort S_NUMPAD9 = (ushort)MapVirtualKey(0x69, 0);
                    public static ushort S_MULTIPLY = (ushort)MapVirtualKey(0x6A, 0);
                    public static ushort S_ADD = (ushort)MapVirtualKey(0x6B, 0);
                    public static ushort S_SEPARATOR = (ushort)MapVirtualKey(0x6C, 0);
                    public static ushort S_SUBTRACT = (ushort)MapVirtualKey(0x6D, 0);
                    public static ushort S_DECIMAL = (ushort)MapVirtualKey(0x6E, 0);
                    public static ushort S_DIVIDE = (ushort)MapVirtualKey(0x6F, 0);
                    public static ushort S_F1 = (ushort)MapVirtualKey(0x70, 0);
                    public static ushort S_F2 = (ushort)MapVirtualKey(0x71, 0);
                    public static ushort S_F3 = (ushort)MapVirtualKey(0x72, 0);
                    public static ushort S_F4 = (ushort)MapVirtualKey(0x73, 0);
                    public static ushort S_F5 = (ushort)MapVirtualKey(0x74, 0);
                    public static ushort S_F6 = (ushort)MapVirtualKey(0x75, 0);
```

```
                    public static ushort S_F7 = (ushort)MapVirtualKey(0x76, 0);
                    public static ushort S_F8 = (ushort)MapVirtualKey(0x77, 0);
                    public static ushort S_F9 = (ushort)MapVirtualKey(0x78, 0);
                    public static ushort S_F10 = (ushort)MapVirtualKey(0x79, 0);
                    public static ushort S_F11 = (ushort)MapVirtualKey(0x7A, 0);
                    public static ushort S_F12 = (ushort)MapVirtualKey(0x7B, 0);
                    public static ushort S_F13 = (ushort)MapVirtualKey(0x7C, 0);
                    public static ushort S_F14 = (ushort)MapVirtualKey(0x7D, 0);
                    public static ushort S_F15 = (ushort)MapVirtualKey(0x7E, 0);
                    public static ushort S_F16 = (ushort)MapVirtualKey(0x7F, 0);
                    public static ushort S_F17 = (ushort)MapVirtualKey(0x80, 0);
                    public static ushort S_F18 = (ushort)MapVirtualKey(0x81, 0);
                    public static ushort S_F19 = (ushort)MapVirtualKey(0x82, 0);
                    public static ushort S_F20 = (ushort)MapVirtualKey(0x83, 0);
                    public static ushort S_F21 = (ushort)MapVirtualKey(0x84, 0);
                    public static ushort S_F22 = (ushort)MapVirtualKey(0x85, 0);
                    public static ushort S_F23 = (ushort)MapVirtualKey(0x86, 0);
                    public static ushort S_F24 = (ushort)MapVirtualKey(0x87, 0);
                    public static ushort S_NUMLOCK = (ushort)MapVirtualKey(0x90, 0);
                    public static ushort S_SCROLL = (ushort)MapVirtualKey(0x91, 0);
                    public static ushort S_LeftShift = (ushort)MapVirtualKey(0xA0, 0);
                    public static ushort S_RightShift = (ushort)MapVirtualKey(0xA1, 0);
                    public static ushort S_LeftControl = (ushort)MapVirtualKey(0xA2, 0);
                    public static ushort S_RightControl = (ushort)MapVirtualKey(0xA3, 0);
                    public static ushort S_LMENU = (ushort)MapVirtualKey(0xA4, 0);
                    public static ushort S_RMENU = (ushort)MapVirtualKey(0xA5, 0);
                    public static ushort S_BROWSER_BACK = (ushort)MapVirtualKey(0xA6, 0);
                    public static ushort S_BROWSER_FORWARD = (ushort)MapVirtualKey(0xA7,
0);
                    public static ushort S_BROWSER_REFRESH = (ushort)MapVirtualKey(0xA8,
0);
                    public static ushort S_BROWSER_STOP = (ushort)MapVirtualKey(0xA9, 0);
                    public static ushort S_BROWSER_SEARCH = (ushort)MapVirtualKey(0xAA, 0);
                    public static ushort S_BROWSER_FAVORITES = (ushort)MapVirtualKey(0xAB,
0);
                    public static ushort S_BROWSER_HOME = (ushort)MapVirtualKey(0xAC, 0);
                    public static ushort S_VOLUME_MUTE = (ushort)MapVirtualKey(0xAD, 0);
                    public static ushort S_VOLUME_DOWN = (ushort)MapVirtualKey(0xAE, 0);
                    public static ushort S_VOLUME_UP = (ushort)MapVirtualKey(0xAF, 0);
                    public static ushort S_MEDIA_NEXT_TRACK = (ushort)MapVirtualKey(0xB0,
0);
                    public static ushort S_MEDIA_PREV_TRACK = (ushort)MapVirtualKey(0xB1,
0);
                    public static ushort S_MEDIA_STOP = (ushort)MapVirtualKey(0xB2, 0);
                    public static ushort S_MEDIA_PLAY_PAUSE = (ushort)MapVirtualKey(0xB3,
0);
                    public static ushort S_LAUNCH_MAIL = (ushort)MapVirtualKey(0xB4, 0);
                    public static ushort S_LAUNCH_MEDIA_SELECT =
(ushort)MapVirtualKey(0xB5, 0);
                    public static ushort S_LAUNCH_APP1 = (ushort)MapVirtualKey(0xB6, 0);
                    public static ushort S_LAUNCH_APP2 = (ushort)MapVirtualKey(0xB7, 0);
                    public static ushort S_OEM_1 = (ushort)MapVirtualKey(0xBA, 0);
                    public static ushort S_OEM_PLUS = (ushort)MapVirtualKey(0xBB, 0);
                    public static ushort S_OEM_COMMA = (ushort)MapVirtualKey(0xBC, 0);
                    public static ushort S_OEM_MINUS = (ushort)MapVirtualKey(0xBD, 0);
                    public static ushort S_OEM_PERIOD = (ushort)MapVirtualKey(0xBE, 0);
                    public static ushort S_OEM_2 = (ushort)MapVirtualKey(0xBF, 0);
                    public static ushort S_OEM_3 = (ushort)MapVirtualKey(0xC0, 0);
                    public static ushort S_OEM_4 = (ushort)MapVirtualKey(0xDB, 0);
                    public static ushort S_OEM_5 = (ushort)MapVirtualKey(0xDC, 0);
                    public static ushort S_OEM_6 = (ushort)MapVirtualKey(0xDD, 0);
                    public static ushort S_OEM_7 = (ushort)MapVirtualKey(0xDE, 0);
                    public static ushort S_OEM_8 = (ushort)MapVirtualKey(0xDF, 0);
                    public static ushort S_OEM_102 = (ushort)MapVirtualKey(0xE2, 0);
                    public static ushort S_PROCESSKEY = (ushort)MapVirtualKey(0xE5, 0);
                    public static ushort S_PACKET = (ushort)MapVirtualKey(0xE7, 0);
```

```
    public static ushort S_ATTN = (ushort)MapVirtualKey(0xF6, 0);
    public static ushort S_CRSEL = (ushort)MapVirtualKey(0xF7, 0);
    public static ushort S_EXSEL = (ushort)MapVirtualKey(0xF8, 0);
    public static ushort S_EREOF = (ushort)MapVirtualKey(0xF9, 0);
    public static ushort S_PLAY = (ushort)MapVirtualKey(0xFA, 0);
    public static ushort S_ZOOM = (ushort)MapVirtualKey(0xFB, 0);
    public static ushort S_NONAME = (ushort)MapVirtualKey(0xFC, 0);
    public static ushort S_PA1 = (ushort)MapVirtualKey(0xFD, 0);
    public static ushort S_OEM_CLEAR = (ushort)MapVirtualKey(0xFE, 0);
}
public class Input
{
    private IntPtr context;
    public KeyboardFilterMode KeyboardFilterMode { get; set; }
    public MouseFilterMode MouseFilterMode { get; set; }
    public bool IsLoaded { get; set; }
    public Input()
    {
        context = IntPtr.Zero;
        KeyboardFilterMode = KeyboardFilterMode.None;
        MouseFilterMode = MouseFilterMode.None;
    }
    public bool Load()
    {
        context = InterceptionDriver.CreateContext();
        return true;
    }
    public void Unload()
    {
        InterceptionDriver.DestroyContext(context);
    }
    public void SendKey(Keys key, KeyState state, int keyboardId)
    {
        Stroke stroke = new Stroke();
        KeyStroke keyStroke = new KeyStroke();
        keyStroke.Code = key;
        keyStroke.State = state;
        stroke.Key = keyStroke;
        InterceptionDriver.Send(context, keyboardId, ref stroke, 1);
    }
    public void SendKey(Keys key, int keyboardId)
    {
        SendKey(key, KeyState.Down, keyboardId);
    }
    public void SendKeyF(Keys key, int keyboardId)
    {
        SendKey(key, KeyState.Up, keyboardId);
    }
    public void SendMouseEvent(MouseState state, int mouseId)
    {
        Stroke stroke = new Stroke();
        MouseStroke mouseStroke = new MouseStroke();
        mouseStroke.State = state;
        if (state == MouseState.ScrollUp)
        {
            mouseStroke.Rolling = 120;
        }
        else if (state == MouseState.ScrollDown)
        {
            mouseStroke.Rolling = -120;
        }
        stroke.Mouse = mouseStroke;
        InterceptionDriver.Send(context, mouseId, ref stroke, 1);
    }
    public void SendLeftClick(int mouseId)
    {
        SendMouseEvent(MouseState.LeftDown, mouseId);
    }
    public void SendRightClick(int mouseId)
    {
        SendMouseEvent(MouseState.RightDown, mouseId);
    }
    public void SendLeftClickF(int mouseId)
    {
        SendMouseEvent(MouseState.LeftUp, mouseId);
    }
    public void SendRightClickF(int mouseId)
    {
        SendMouseEvent(MouseState.RightUp, mouseId);
    }
    public void ScrollMouse(ScrollDirection direction, int mouseId)
    {
        switch (direction)
        {
            case ScrollDirection.Down:
                SendMouseEvent(MouseState.ScrollDown, mouseId);
                break;
            case ScrollDirection.Up:
                SendMouseEvent(MouseState.ScrollUp, mouseId);
                break;
        }
    }
    public void MoveMouseBy(int deltaX, int deltaY, int mouseId)
    {
        Stroke stroke = new Stroke();
        MouseStroke mouseStroke = new MouseStroke();
        mouseStroke.X = deltaX;
        mouseStroke.Y = deltaY;
        stroke.Mouse = mouseStroke;
        stroke.Mouse.Flags = MouseFlags.MoveRelative;
        InterceptionDriver.Send(context, mouseId, ref stroke, 1);
    }
    public void MoveMouseTo(int x, int y, int mouseId)
    {
        Stroke stroke = new Stroke();
        MouseStroke mouseStroke = new MouseStroke();
        mouseStroke.X = x;
        mouseStroke.Y = y;
        stroke.Mouse = mouseStroke;
        stroke.Mouse.Flags = MouseFlags.MoveAbsolute;
        InterceptionDriver.Send(context, mouseId, ref stroke, 1);
    }
}
[UnmanagedFunctionPointer(CallingConvention.Cdecl)]
public delegate int Predicate(int device);
[Flags]
public enum KeyState : ushort
{
    Down = 0x00,
    Up = 0x01,
    E0 = 0x02,
    E1 = 0x04,
    TermsrvSetLED = 0x08,
    TermsrvShadow = 0x10,
    TermsrvVKPacket = 0x20
}
[Flags]
public enum KeyboardFilterMode : ushort
{
    None = 0x0000,
    All = 0xFFFF,
    KeyDown = KeyState.Up,
    KeyUp = KeyState.Up << 1,
    KeyE0 = KeyState.E0 << 1,
    KeyE1 = KeyState.E1 << 1,
    KeyTermsrvSetLED = KeyState.TermsrvSetLED << 1,
    KeyTermsrvShadow = KeyState.TermsrvShadow << 1,
    KeyTermsrvVKPacket = KeyState.TermsrvVKPacket << 1
}
[Flags]
public enum MouseState : ushort
{
    LeftDown = 0x01,
    LeftUp = 0x02,
    RightDown = 0x04,
    RightUp = 0x08,
    MiddleDown = 0x10,
    MiddleUp = 0x20,
    LeftExtraDown = 0x40,
    LeftExtraUp = 0x80,
    RightExtraDown = 0x100,
    RightExtraUp = 0x200,
    ScrollVertical = 0x400,
    ScrollUp = 0x400,
    ScrollDown = 0x400,
    ScrollHorizontal = 0x800,
    ScrollLeft = 0x800,
    ScrollRight = 0x800,
}
[Flags]
public enum MouseFilterMode : ushort
{
    None = 0x0000,
    All = 0xFFFF,
    LeftDown = 0x01,
    LeftUp = 0x02,
    RightDown = 0x04,
    RightUp = 0x08,
    MiddleDown = 0x10,
    MiddleUp = 0x20,
    LeftExtraDown = 0x40,
    LeftExtraUp = 0x80,
    RightExtraDown = 0x100,
    RightExtraUp = 0x200,
    MouseWheelVertical = 0x400,
    MouseWheelHorizontal = 0x800,
    MouseMove = 0x1000,
}
[Flags]
public enum MouseFlags : ushort
{
    MoveRelative = 0x000,
    MoveAbsolute = 0x001,
    VirtualDesktop = 0x002,
    AttributesChanged = 0x004,
    MoveWithoutCoalescing = 0x008,
    TerminalServicesSourceShadow = 0x100
}
[StructLayout(LayoutKind.Sequential)]
public struct MouseStroke
{
    public MouseState State;
    public MouseFlags Flags;
    public Int16 Rolling;
    public Int32 X;
    public Int32 Y;
    public UInt16 Information;
}
```

```
                [StructLayout(LayoutKind.Sequential)]
                public struct KeyStroke
                {
                    public Keys Code;
                    public KeyState State;
                    public UInt32 Information;
                }
                [StructLayout(LayoutKind.Explicit)]
                public struct Stroke
                {
                    [FieldOffset(0)]
                    public MouseStroke Mouse;
                    [FieldOffset(0)]
                    public KeyStroke Key;
                }
                public static class InterceptionDriver
                {
                    [DllImport(""interception.dll"", EntryPoint =
""interception_create_context"", CallingConvention = CallingConvention.Cdecl)]
                    public static extern IntPtr CreateContext();
                    [DllImport(""interception.dll"", EntryPoint =
""interception_destroy_context"", CallingConvention = CallingConvention.Cdecl)]
                    public static extern void DestroyContext(IntPtr context);
                    [DllImport(""interception.dll"", EntryPoint =
""interception_get_precedence"", CallingConvention = CallingConvention.Cdecl)]
                    public static extern void GetPrecedence(IntPtr context, Int32 device);
                    [DllImport(""interception.dll"", EntryPoint =
""interception_set_precedence"", CallingConvention = CallingConvention.Cdecl)]
                    public static extern void SetPrecedence(IntPtr context, Int32 device,
Int32 Precedence);
                    [DllImport(""interception.dll"", EntryPoint =
""interception_get_filter"", CallingConvention = CallingConvention.Cdecl)]
                    public static extern void GetFilter(IntPtr context, Int32 device);
                    [DllImport(""interception.dll"", EntryPoint =
""interception_set_filter"", CallingConvention = CallingConvention.Cdecl)]
                    public static extern void SetFilter(IntPtr context, Predicate
predicate, Int32 keyboardFilterMode);
                    [DllImport(""interception.dll"", EntryPoint = ""interception_wait"",
CallingConvention = CallingConvention.Cdecl)]
                    public static extern Int32 Wait(IntPtr context);
                    [DllImport(""interception.dll"", EntryPoint =
""interception_wait_with_timeout"", CallingConvention = CallingConvention.Cdecl)]
                    public static extern Int32 WaitWithTimeout(IntPtr context, UInt64
milliseconds);
                    [DllImport(""interception.dll"", EntryPoint = ""interception_send"",
CallingConvention = CallingConvention.Cdecl)]
                    public static extern Int32 Send(IntPtr context, Int32 device, ref
Stroke stroke, UInt32 numStrokes);
                    [DllImport(""interception.dll"", EntryPoint = ""interception_receive"",
CallingConvention = CallingConvention.Cdecl)]
                    public static extern Int32 Receive(IntPtr context, Int32 device, ref
Stroke stroke, UInt32 numStrokes);
                    [DllImport(""interception.dll"", EntryPoint =
""interception_get_hardware_id"", CallingConvention = CallingConvention.Cdecl)]
                    public static extern Int32 GetHardwareId(IntPtr context, Int32 device,
String hardwareIdentifier, UInt32 sizeOfString);
                    [DllImport(""interception.dll"", EntryPoint =
""interception_is_invalid"", CallingConvention = CallingConvention.Cdecl)]
                    public static extern Int32 IsInvalid(Int32 device);
                    [DllImport(""interception.dll"", EntryPoint =
""interception_is_keyboard"", CallingConvention = CallingConvention.Cdecl)]
                    public static extern Int32 IsKeyboard(Int32 device);
                    [DllImport(""interception.dll"", EntryPoint =
""interception_is_mouse"", CallingConvention = CallingConvention.Cdecl)]
                    public static extern Int32 IsMouse(Int32 device);
                }
```

```
public class KeyPressedEventArgs : EventArgs
{
    public Keys Key { get; set; }
    public KeyState State { get; set; }
    public bool Handled { get; set; }
}
public enum Keys : ushort
{
    Escape = 1,
    One = 2,
    Two = 3,
    Three = 4,
    Four = 5,
    Five = 6,
    Six = 7,
    Seven = 8,
    Eight = 9,
    Nine = 10,
    Zero = 11,
    DashUnderscore = 12,
    PlusEquals = 13,
    Backspace = 14,
    Tab = 15,
    Q = 16,
    W = 17,
    E = 18,
    R = 19,
    T = 20,
    Y = 21,
    U = 22,
    I = 23,
    O = 24,
    P = 25,
    OpenBracketBrace = 26,
    CloseBracketBrace = 27,
    Enter = 28,
    Control = 29,
    A = 30,
    S = 31,
    D = 32,
    F = 33,
    G = 34,
    H = 35,
    J = 36,
    K = 37,
    L = 38,
    SemicolonColon = 39,
    SingleDoubleQuote = 40,
    Tilde = 41,
    LeftShift = 42,
    BackslashPipe = 43,
    Z = 44,
    X = 45,
    C = 46,
    V = 47,
    B = 48,
    N = 49,
    M = 50,
    CommaLeftArrow = 51,
    PeriodRightArrow = 52,
    ForwardSlashQuestionMark = 53,
    RightShift = 54,
    RightAlt = 56,
    Space = 57,
    CapsLock = 58,
    F1 = 59,
                    F2 = 60,
                    F3 = 61,
                    F4 = 62,
                    F5 = 63,
                    F6 = 64,
                    F7 = 65,
                    F8 = 66,
                    F9 = 67,
                    F10 = 68,
                    F11 = 87,
                    F12 = 88,
                    Up = 72,
                    Down = 80,
                    Right = 77,
                    Left = 75,
                    Home = 71,
                    End = 79,
                    Delete = 83,
                    PageUp = 73,
                    PageDown = 81,
                    Insert = 82,
                    PrintScreen = 55,
                    NumLock = 69,
                    ScrollLock = 70,
                    Menu = 93,
                    WindowsKey = 91,
                    NumpadDivide = 53,
                    NumpadAsterisk = 55,
                    Numpad7 = 71,
                    Numpad8 = 72,
                    Numpad9 = 73,
                    Numpad4 = 75,
                    Numpad5 = 76,
                    Numpad6 = 77,
                    Numpad1 = 79,
                    Numpad2 = 80,
                    Numpad3 = 81,
                    Numpad0 = 82,
                    NumpadDelete = 83,
                    NumpadEnter = 28,
                    NumpadPlus = 78,
                    NumpadMinus = 74,
                }
                public class MousePressedEventArgs : EventArgs
                {
                    public MouseState State { get; set; }
                    public bool Handled { get; set; }
                    public int X { get; set; }
                    public int Y { get; set; }
                    public short Rolling { get; set; }
                }
                public enum ScrollDirection
                {
                    Down,
                    Up
                }
            }
            ";
            string irx = "10", iry = "10";
            string line1 = "int mousex = (int)irx * 10; int mousey = (int)iry * 10;
SetCursorPos(mousex, mousey);";
            string finalline1 = line1.Replace("irx", "Int32.Parse(\"" + irx + "\")");
            finalline1 = finalline1.Replace("iry", "Int32.Parse(\"" + iry + "\")");
            string finalcode = code.Replace("funct_xy", finalline1);
            finalcode = code.Replace("funct_mouse", "int mousex = (int)x * 10; int mousey =
(int)y * 10; SetCursorPos(mousex, mousey);");
```

```
            finalcode = code.Replace("funct_driver", "int mousex = (int)x * 1500; int
mousey = (int)y * 1500; input.MoveMouseTo(mousex, mousey, 11);");
            System.CodeDom.Compiler.CompilerParameters parameters = new
System.CodeDom.Compiler.CompilerParameters();
            parameters.ReferencedAssemblies.Add("System.Windows.Forms.dll");
            Microsoft.CSharp.CSharpCodeProvider provider = new
Microsoft.CSharp.CSharpCodeProvider();
            System.CodeDom.Compiler.CompilerResults results =
provider.CompileAssemblyFromSource(parameters, finalcode);
            if (results.Errors.HasErrors)
            {
                StringBuilder sb = new StringBuilder();
                foreach (System.CodeDom.Compiler.CompilerError error in results.Errors)
                {
                    sb.AppendLine(String.Format("Error ({0}) : {1}", error.ErrorNumber,
error.ErrorText));
                }
                MessageBox.Show(sb.ToString());
                return;
            }
            Assembly assembly = results.CompiledAssembly;
            Type program = assembly.GetType("StringToCode.FooClass");
            object obj = Activator.CreateInstance(program);
            double x = 10, y = 10;
            program.InvokeMember("Open", BindingFlags.Default | BindingFlags.InvokeMethod,
null, obj, new object[] { x, y });
            for (int i = 0; i <= 1000; i++)
            {
                program.InvokeMember("Main", BindingFlags.Default |
BindingFlags.InvokeMethod, null, obj, new object[] { x, y });
                System.Threading.Thread.Sleep(10);
            }
            program.InvokeMember("Close", BindingFlags.Default | BindingFlags.InvokeMethod,
null, obj, new object[] { x, y });
        }
    }
}
```

7. Use and Agreement Contract

Owner: Michael Andre Franiatte.

Contact: michael.franiatte@gmail.com.

Owning: All works from scratch of the owner.

Proof of Owning: Works published, and writings/speakings all over.

Requirements of Use: Pay the owner, quote the owner, agreement of the owner.

Availability of Works: Only under the shapes of the owner built, only for personal use.

Subjects of Claims: Works published by the owner on Google Play and Google Books.

Concerning Author Rights: Equations and codes from scratch of the owner, softwares built from it, all things of people arising from it.

End User License Agreement: A commercial license is required to use in personal manner. Do not redistributing in any manner, including by computer media, a file server, an email attachment, etc. Do not embedding in or linking it to another programs, source codes and