

Politechnika Wrocławska  
Wydział Elektroniki  
Urządzenia Peryferyjne

---

# OBSŁUGA JOYSTICKA USB Z WYKORZYSTANIEM DIRECTINPUT

---

*Termin zajęć:*

WTOREK TN 7:30

*Autorzy:*

JAKUB CHMIEL 235028

TOMASZ CIEŚLAR 235652

*Prowadzący:*

dr inż. Tomasz Walkowiak

14 stycznia 2019

# Spis treści

1	Cel ćwiczenia . . . . .	2
2	Wstęp . . . . .	2
3	Założenia projektowe . . . . .	2
4	Wykorzystane narzędzia . . . . .	2
5	Implementacja programu . . . . .	3
	5.1 Interfejs aplikacji . . . . .	3
	5.2 Kod źródłowy . . . . .	3
6	Wnioski . . . . .	8
7	Bibliografia . . . . .	8

# 1 Cel ćwiczenia

1. Korzystając z przykładowej aplikacji stwierdzić obecność joysticka podłączonego do portu USB komputera
2. Napisać program, odczytujący nazwę zainstalowanego joysticka, wykorzystując w tym celu API DirectInput
3. Napisać program ilustrujący działanie joysticka: stwierdzający naciśnięcie myszy, zmianę położenia drążka (w przestrzeni 2D) oraz suwaka.
4. Napisać program zastępujący działanie myszy. Program ma umożliwiać sterowanie kursorem za pomocą joysticka oraz obsługę przycisków fire jako kliknięć myszy.
5. Napisać program realizujący prosty edytor graficzny - rysowanie przy pomocy Joysticka

# 2 Wstęp

DirectInput jest to biblioteka obsługująca urządzenia wejściowe tj. klawiatura, mysz czy joystick, będąca częścią DirectX firmy Microsoft. Zawiera w sobie funkcje odczytujące dane z tych urządzeń w różny sposób (bezpośredni lub buforowany). Ma też możliwość przyporządkowania określonych akcji do konkretnych przycisków (Action Mapping). DirectInput nie przyniesie żadnych dodatkowych korzyści aplikacjom, które używają klawiatury do wprowadzania tekstu i myszy do nawigacji. API jest przeznaczone głównie do tworzenia gier komputerowych, symulacji i wszystkich innych interaktywnych aplikacji dla Windows.

# 3 Założenia projektowe

- Program był pisany w języku C#.
- Na komputerze, na którym uruchamiany był program zainstalowano system operacyjny Windows 10 w wersji 64-bitowej.

# 4 Wykorzystane narzędzia

- Windows Forms - API do implementacji interfejsu graficznego dla platformy .NET.
- SlimDX - API dla programowania z użyciem DirectX na platformie .NET.

## 5 Implementacja programu

### 5.1 Interfejs aplikacji



*Rysunek 1. Interfejs aplikacji*

### 5.2 Kod źródłowy

```
using [...]  
  
namespace USB  
{  
    public partial class Form1 : Form  
    {  
        Graphics g;  
        private int x = -1;  
        private int y = -1;  
        private bool moving = false;  
        private Pen pen;  
        private Color currentColor = Color.Black;  
        private Color previousColor = Color.Black;  
        private Color currColor = Color.Black;  
        private bool prev = false;  
  
        public Form1()  
        {  
            InitializeComponent();  
            GetSticks();  
            Sticks = GetSticks();  
            timer1.Enabled = true;  
            //Przygotowanie panelu do rysowania  
            g = panel1.CreateGraphics();  
            g.SmoothingMode =
```

```

        System.Drawing.Drawing2D.SmoothingMode.
        AntiAlias;
    pen = new Pen(Color.Black, 5);
    pen.StartCap = pen.EndCap =
        System.Drawing.Drawing2D.LineCap.Round;
}

//
DirectInput Input = new DirectInput();
SlimDX.DirectInput.Joystick stick;
Joystick[] Sticks;

bool mouseClicked = false;

int yValue = 0;
int xValue = 0;
int zValue = 0;

[DllImport("user32.dll", CharSet = CharSet.Auto)]
public static extern void mouse_event(uint flag, uint
    _x, uint _y, uint btn, uint exInfo);
private const int MOUSEEVENT_LEFTDOWN = 0x02;
private const int MOUSEEVENT_LEFTUP = 0x04;

public Joystick[] GetSticks()
{
    //Lista podlaczonych urzadzen typu joystick
    List<SlimDX.DirectInput.Joystick> sticks = new
    List<SlimDX.DirectInput.Joystick>();
    foreach (DeviceInstance device in
        Input.GetDevices(DeviceClass.GameController,
        DeviceEnumerationFlags.AttachedOnly))
    {
        try
        {
            //Pobranie informacji o urzadzeniu
            stick = new
                SlimDX.DirectInput.Joystick(Input,
                device.InstanceGuid);
            stick.Acquire();

            foreach(DeviceObjectInstance deviceObject
                in stick.GetObjects())
            {
                if((deviceObject.ObjectType &
                    ObjectDeviceType.Axis) != 0)
                {
                    stick.GetObjectPropertiesById
                        ((int)deviceObject.ObjectType).
                        SetRange(-100, 100);
                }
            }
        }
        catch { }
    }
}

```

```

        }
    }
    sticks.Add(stick);
}
catch (DirectInputException)
{
}
}
return sticks.ToArray();
}

void stickHandle(Joystick stick, int id)
{
    //Obsługa joysticka
    JoystickState state = new JoystickState();
    state = stick.GetCurrentState();

    //Wychylenia osi X, Y, Z
    yValue = state.Y;
    xValue = state.X;
    zValue = state.Z;
    pen.Width = (float)((zValue + 101) * 0.2);
    MouseMove(xValue, yValue);

    //Pobranie stanów przycisków
    bool[] buttons = state.GetButtons();

    if(id == 0)
    {
        if (buttons[1])
        {
            //Czyszczenie panelu
            panel1.BackColor = Color.White;
            panel1.Refresh();
        }
        if (buttons[2])
        {
            //Przelaczanie miedzy ostatnimi dwoma
            kolorami
            if (prev)
            {
                pen.Color = currentColor;
                prev = false;
            }
            else
            {
                pen.Color = previousColor;
                prev = true;
            }
        }
    }
}

```

```

        currColor = pen.Color;
    }
    if (buttons[0])
    {
        //Rysowanie (wcisniecie lewego
        przycisku myszy)
        if (mouseClicked == false)
        {
            mouse_event(MOUSEEVENT_LEFTDOWN, 0, 0,
                0, 0);
            mouseClicked = true;
        }
    }
    else
    {
        //Zwolnienie LPM
        if (mouseClicked == true)
        {
            mouse_event(MOUSEEVENT_LEFTUP, 0, 0, 0,
                0);
            mouseClicked = false;
        }
    }
}

public void MouseMove (int posX, int posY)
{
    //Poruszanie kursorem
    Cursor.Position = new Point(Cursor.Position.X +
        posX/3, Cursor.Position.Y + posY/3);
    Cursor.Clip = new Rectangle(this.Location,
        this.Size);
}

private void timer1_Tick(object sender, EventArgs e)
{
    //Obsluga timera, ktory wywoluje metode
    zbierajaca informacje o aktualnym stanie
    joysticka
    for (int i = 0; i < Sticks.Length; i++)
    {
        stickHandle(Sticks[i], i);
    }
}

private void Form1_Load(object sender, EventArgs e)
{

```

```

        Joystick[] joystick = GetSticks();
    }

    private void pictureBox1_Click(object sender, EventArgs
    e)
    {
        //Przelaczanie kolorow
        pictureBox1.BorderStyle = BorderStyle.None;
        pictureBox2.BorderStyle = BorderStyle.None;
        pictureBox3.BorderStyle = BorderStyle.None;
        pictureBox4.BorderStyle = BorderStyle.None;

        PictureBox p = (PictureBox) sender;

        pen.Color = p.BackColor;
        previousColor = currColor;
        currentColor = p.BackColor;

        if (p.BorderStyle == BorderStyle.FixedSingle)
        {
            p.BorderStyle = BorderStyle.None;
            pen.Color = Color.Black;
        }
        else
        {
            p.BorderStyle = BorderStyle.FixedSingle;
        }
    }

    private void panel1_MouseUp(object sender,
    MouseEventArgs e)
    {
        //Zwolnienie LPM - koniec rysowania
        moving = false;
        x = -1;
        y = -1;
        panel1.Cursor = Cursors.Default;
    }

    private void panel1_MouseDown(object sender,
    MouseEventArgs e)
    {
        //Wcisniecie LPM - rozpoczecie rysowania
        moving = true;
        x = e.X;
        y = e.Y;
        g.DrawLine(pen, new Point(x, y), new Point(x + 1,
        y));
        panel1.Cursor = Cursors.Cross;
    }

```



```

private void panel1_MouseMove(object sender,
    MouseEventArgs e)
{
    //Poruszanie mysza - rysowanie linii
    if (moving && x != -1 && y != -1)
    {
        g.DrawLine(pen, new Point(x, y), e.Location);
        x = e.X;
        y = e.Y;
    }
}
}

```

## 6 Wnioski

Udało nam się wykonać zadanie w całości. Joystick przejmował operację kursora myszki i umożliwiał rysowanie na panelu w oknie aplikacji. API SlimDX znacząco ułatwiło obsługę urządzenia.

## 7 Bibliografia

- DirectInput:  
*<https://en.wikipedia.org/wiki/DirectInput>*
- Dokumentacja SlimDX:  
*<https://slimdx.org/docs/>*