Konrad Kieda 250997
Dmytro Matviichuk 240545

Termin: pt TN 17:15-20:15

Urządzenia peryferyjne
Temat: Obsługa joysticka USB z wykorzystaniem DirectInput

Kod źródłowy(GitHub): https://github.com/dmatviichuk/Urzadzenia_Peryferyjne

1. Cel ćwiczenia

Celem ćwiczeń było zapoznanie się z obsługą joysticka USB. Zadaniem do wykonania w trakcie zajęć było napisanie programu odczytującego nazwę zainstalowanego joysticka, wykorzystując w tym celu API DirectInput. Kolejnym zadaniem było napisanie programu stwierdzającego naciśnięcie przycisku, zmianę położenia drążka (w przestrzeni 2D) oraz suwaka. Następnie program miał umożliwiać sterowanie kursorem za pomocą joysticka. Ostatnie zadanie, jakie było do wykonania, to napisanie programu realizującego prosty edytor graficzny, umożliwiający rysowanie przy pomocy joysticka.

2. Wstęp teoretyczny

Dżojstik lub **manipulator drążkowy** – urządzenie wejścia komputera, manipulator służący do sterowania ruchem obiektów na ekranie. W podstawowej wersji składa się z wychylnego drążka zamocowanego na podstawce, którego przechylenie w odpowiednim kierunku powoduje stosowną reakcję sterowanego obiektu, oraz z umieszczonych na drążku i podstawce przycisków uruchamiających przypisane im działania i dodatkowe funkcje sterujące.

3. Infromancja o programie oraz fragmenty kodu

Program został napisany w technologii C#, z użyciem biblioteki DirectInput.

3.1 Biblioteki które zostały użyte w naszej aplikacji:

```
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 SharpDX.DirectInput;
using System.Threading;
```

3.2 Emulacja myszy:

```
    public void Emulation()
    {
        while (true)
        {
            int inputOffset = (1 << 15) - 1;
            int x = joystick.GetCurrentState().X - inputOffset;
            int y = joystick.GetCurrentState().Y - inputOffset;

            float xOffset = (float)x / inputOffset;
            float yOffset = (float)y / inputOffset;

            int slider = joystick.GetCurrentState().Z;

            MouseSpeed = 50 * slider / ((1 << 16) - 1);

            mouseX += xOffset * MouseSpeed;
            mouseY += yOffset * MouseSpeed;
```

```
            if (mouseX / 50 < 0)
                mouseX = 0;
            if (mouseX / 50 > 1300)
                mouseX = 1300 * 50;
            if (mouseY / 50 < 0)
                mouseY = 0;
            if (mouseY / 50 > 1300)
                mouseY = 1300 * 50;


            uint flags = (uint)(MouseEventFlags.ABSOLUTE | MouseEventFlags.MOVE);

            if (joystick.GetCurrentState().Buttons[0])
            {
                flags |= (uint)MouseEventFlags.LEFTDOWN;
            }
            else
            {
                flags |= (uint)MouseEventFlags.LEFTUP;
            }

            if (joystick.GetCurrentState().Buttons[2])
            {
                System.Environment.Exit(0);
            }

            if (ifJoystickInUse)
                mouse_event(flags, (uint)mouseX, (uint)mouseY, 0, 0);


        }

    }
```

3.3 Prosty edytor graficzny**:**

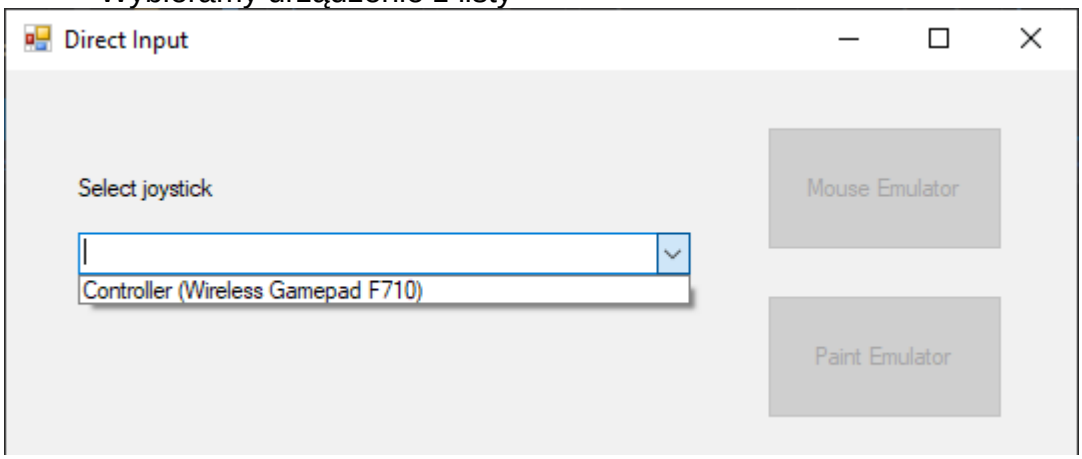
```
public partial class PaintEmulator : Form {
    private System.Threading.Thread myThread;
    private Joystick joystick;

    float x = 0, y = 0;
    public bool showCross = false;
    private PaintEmulator() {
        InitializeComponent();
    }

    public void dodajJoystick(Joystick joystick) {
        this.joystick = joystick;
    }


    public static PaintEmulator CreateCanvas() {
        PaintEmulator canvas = null;
```

```
    System.Threading.Thread thread = new System.Threading.Thread(() => {
        canvas = new PaintEmulator();
        canvas.ShowDialog();
    });
    thread.Start();
    while(canvas == null) {
        System.Threading.Thread.Sleep(10);
    }
    canvas.myThread = thread;

    return canvas;
}

public int WindowWidth { get { return pictureBox1.Width; } }
public int WindowHeight { get { return pictureBox1.Height; } }
private void Timer1_Tick(object sender, EventArgs e) {

    if(updated) {
        updated = false;
        Refresh();
    }


}
private void PictureBox1_Paint(object sender, PaintEventArgs e) {
    lock(bmp) {
        e.Graphics.DrawImage(bmp, 0, 0);
    }
    using(SolidBrush pedzel = new SolidBrush(Color.Black)) {
        e.Graphics.FillEllipse(pedzel, x - 4, y - 4, 2 * 4, 2 * 4);
    }
}

public void ClearScreen() {
    bmp = new Bitmap(this.pictureBox1.Width, this.pictureBox1.Height);
    lock(bmp) {
        using(Graphics g = Graphics.FromImage(bmp))
        using(SolidBrush pedzel = new SolidBrush(Color.White)) {
            g.FillRectangle(pedzel, 0, 0, WindowWidth, WindowHeight);
        }
        updated = true;
    }
}
public void Draw(float x, float y, float r) {
    lock(bmp) {
        using(Graphics g = Graphics.FromImage(bmp))
        using(SolidBrush pedzel = new SolidBrush(Color.Green)) {
            g.FillEllipse(pedzel, x - r, y - r, 2 * r, 2 * r);
        }
        updated = true;
    }
}
```

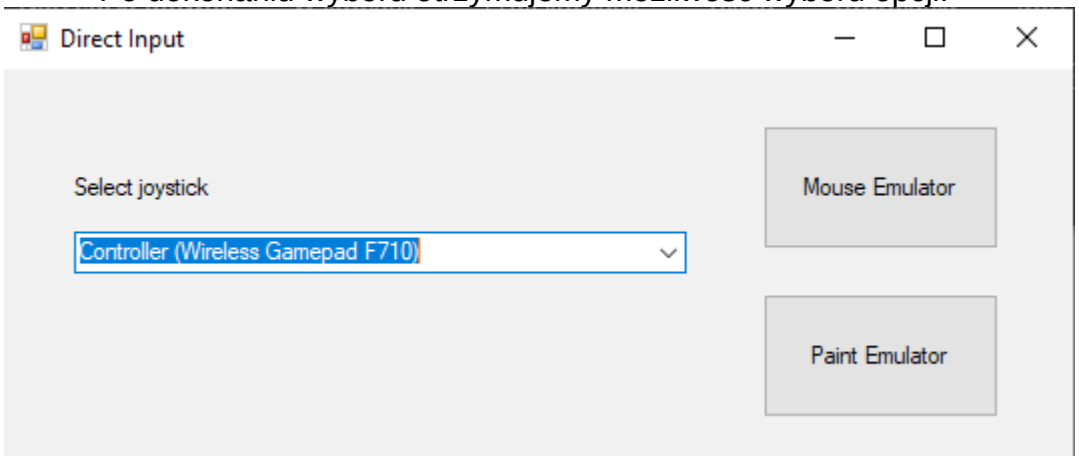
4. Działanie aplikacji

4.1 Menu główne:

Wybieramy urządzenie z listy

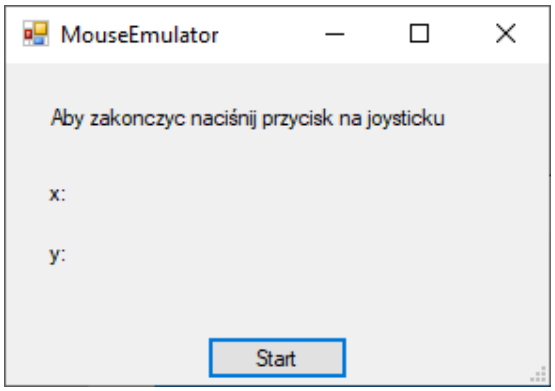


Po dokonaniu wyboru otrzymujemy możliwość wyboru opcji:

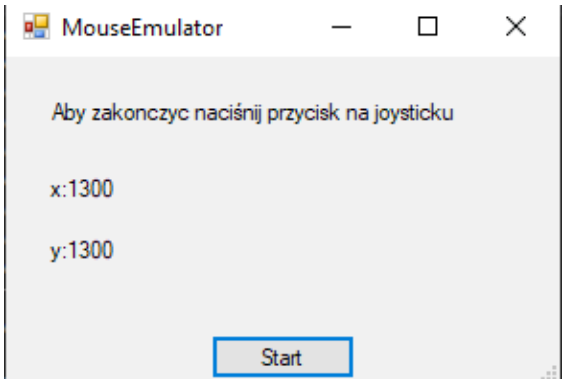


4.2 Opcja *Mouse Emulator*:

Po naciśnięciu przycisku *Start* otrzymujemy możliwość kontroli wskaźnika myszy za pomocą lewego analoga pada.

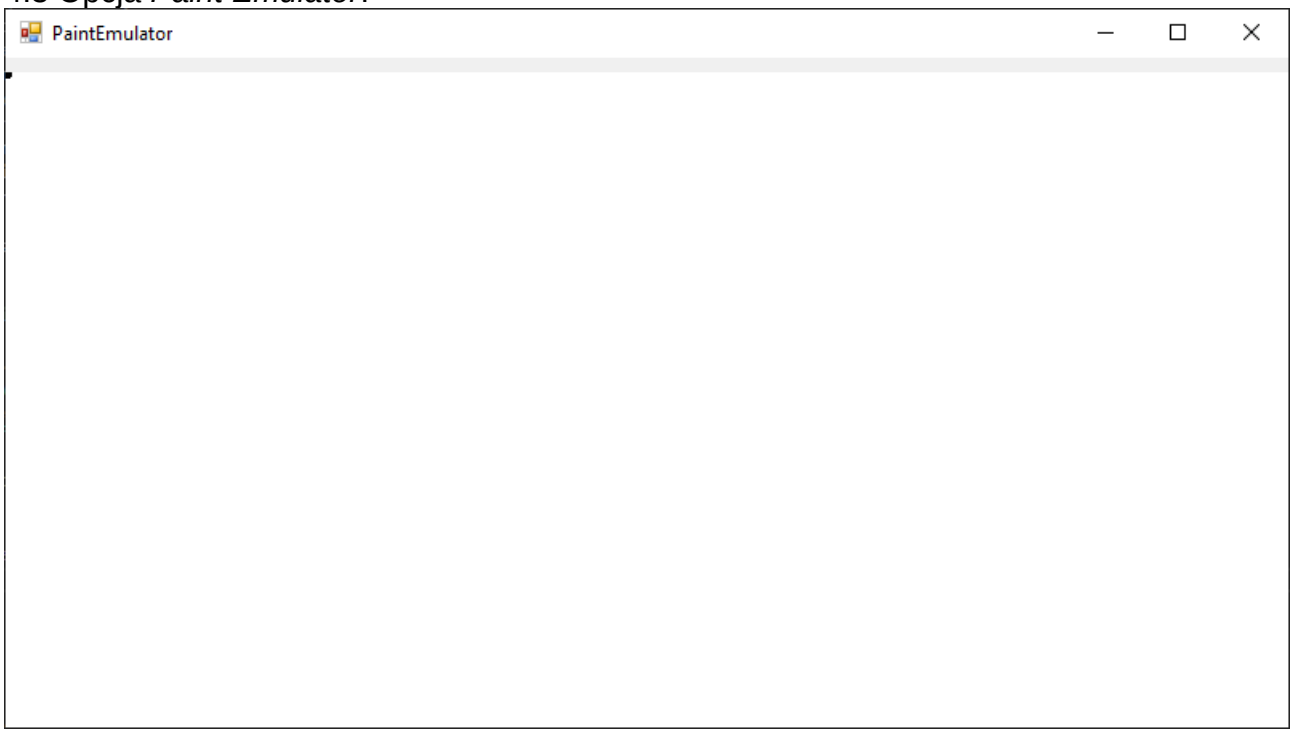


Przed rozpoczęciem emulacji



Ustawienie myszy w dolnym prawym rogu ekranu za pomocą pada

4.3 Opcja *Paint Emulator*:



Ekran pokazujący się po naciśnięciu opcji Paint Emulator

Rysunek wykonany za pomocą pada

5. Wnioski

Podczas laboratoriów nauczyliśmy się podstawowej obsługi joysticka/gamepda. Aplikacja działa poprawnie, zgodnie ze wszystkimi założeniami.