Politechnika Wrocławska

Wydział Informatyki i Telekomunikacji

Urządzenia peryferyjne

Obsługa joysticka USB z wykorzystaniem DirectInput

Ćwiczenie nr 10

Prowadzący:

dr inż. Jan Nikodem

Grupa:

Piątek 13:15 TP

Autorzy:

Daniel Glazer, 252743

Paweł Helisz, 252779

Wrocław 22.10.2021

1 Wstęp

Celem ćwiczenia było zapoznanie się z obsługą joysticka wykorzystującego port USB. Zadania jakie miały zostać wykonane w trakcie ćwiczeń opierały się na napisaniu programu, który na sam początek miał odczytywać nazwę naszego kontrolera przy użyciu API DirectInput. W kolejnej części ćwiczenia było napisanie programu, ilustrującego działanie joysticka poprzez pokazanie zmiany położenia drążka, wciśnięcia przycisku „fire" oraz zmiany wartości suwaka. Następnie do programu trzeba było dodać funkcję odpowiadającą za emulację myszki za pomocą kontrolera. Ostatnim zadaniem było stworzenie prostego edytora graficznego, w którym powinna istnieć możliwość rysowania za pomocą joysticka. Aby poradzić sobie z tym ćwiczeniem i związanym z nim zadaniami otrzymaliśmy na zajęciach joystick firmy Logitech oraz w domu mieliśmy za zadanie zapoznać się z pojęciami związanymi z USB i biblioteką DirectInput.

Pierwsze dwa pojęcia, które były do opracowania w domu to HAL i HEL, które są kolejno warstwą abstrakcji sprzętu i warstwą emulacji sprzętu. Zadaniem HAL jest ukrycie różnych architektur sprzętowych, które znajdują się pod systemem operacyjnym poprzez zapewnienie jednolitego interfejsu dla urządzeń peryferyjnych. Ma to umożliwić aplikacjom wydobycie jak największej wydajności z urządzeń oraz ułatwić przenoszenia tych urządzeń pomiędzy różnymi architekturami sprzętowymi. Za to zadaniem warstwy emulacji sprzętowej - HEL jest tak jak nazwa mówi, emulowanie funkcji. W momencie, którym HAL nie posiada wystarczającego oprogramowania albo sterowników urządzenia, HEL ma za zadanie emulować programowo daną funkcję, na podstawie tych, które posiada obecnie. W momencie, w którym HEL nie jest w stanie emulować funkcji, która nie jest wspierana poinformuje użytkownika o błędzie.

Następnymi pojęciami, które warto omówić w przypadku połączenia komputera z joystickiem są USB i HID. USB jest to typ uniwersalnej magistrali szeregowej, który zastępuje złącza PS/2. W ramach swojej uniwersalności umożliwia podłączenie do komputera wielu różnych urządzeń typu joysticki, myszki, klawiatury, drukarki itp. Kolejną ważną cechą USB jest jego wsteczna kompatybilność, pozwala ona nam podłączyć do portów nowszej generacji, urządzenia posiadające porty starszej generacji. Przydaje się to zwłaszcza w tym ćwiczeniu, ponieważ komputer do, którego podłączaliśmy pada posiadał gniazda USB 3.0, a nasz joystick posiadał złączę w technologi USB 2.0. Jeśli mówimy już o połączeniu komputera z joystickiem nie możemy nie wspomnieć o HID. Z ang. Human Interface Devices jest to rodzaj urządzenia komputerowego, który pobiera dane od ludzi i w zamian uzyskuje jakieś dane wyjściowe. Koncepcja HID opiera się na tym, że mamy hosta i użytkownika. Host odbiera od użytkownika dane, które w przypadku HID składają się z dwóch typów: z deskryptora raportu i raportów. Deskryptor raportów to tabela zawierająca informacje o tym jak aplikacja ma odczytywać dane z raportów, a same raporty zawierająca informacje o akcjach użytkownika wykonanych na urządzeniu np. tak jak w naszym przypadku - joystick, lub poprzez raporty, aplikacja może wysyłać dane do urządzenia np. do drukarki.

W ostatnim akapicie warto wspomnieć o samym oprogramowaniu, które zostało użyte do napisania programu. W tym celu wykorzystaliśmy bibliotekę DirectInput wchodzącą w skład DirectX. Biblioteka ta zapewnia interfejs umożliwiający odbieranie niezliczonych danych wejściowych urządzenia (ograniczonych przez przepustowość łącza) w sposób bezpośredni lub buforowany. Również ma możliwość przyporządkowania określonych akcji do konkretnych przycisków urządzenia, chociażby przypisanie pod przycisk „fire" lewego klawisza mysz. Dzięki DirectInput również omijamy system wiadomości Windows dzięki bezpośredniej pracy ze sterownikiem urządzenia, a co za tym idzie pozwala nam to na szybszy przesył danych potrzebny w grach komputerowych.

2 Opis działania programu

Program został napisany w języku C#. Po uruchomieniu programu, pojawia się okno naszej aplikacji. Po lewej stronie okna znajduje się lista z nazwami podłączonych kontrolerów. Aby nazwa podłączonego kontrolera pojawiła się w aplikacji, najpierw trzeba podłączyć joystick, a potem uruchomić aplikację. Po prawej stronie naszego okna programu znajdują się trzy przyciski i odpowiadają one za ilustrowania działania joysticka, emulowania myszki oraz uruchomienie edytora graficznego. Aby uruchomić jedną z podanych funkcji należy wybrać z listy kontrolerów podany kontroler, a następnie nacisnąć jeden z przycisków. W przypadku edytora graficznego, sterowania pędzlem odbywa się za pomocą drążka joysticka, przycisk „fire" odpowiada za malowanie pędzlem, przycisk 2 (względem instrukcji, albo Button[1] względem kodu) odpowiada za czyszczenie ekranu, a suwak zmienia grubości pędzla.

3 Odczytywanie nazwy joysticka

Aby odczytać nazwę urządzenia należało wykorzystać w tym celu bibliotekę SharpDX.DirectInput. Dodano w ten sposób wszystkie wykryte gamepady oraz joysticki na listę urządzeń. Następnie lista ta została wyświetlona na ekranie.

```
public OknoListyUrzadzen()
        {
            InitializeComponent();

            //wykorzystanie DirectInput do komunikacji z urzadzeniami
            directInput = new SharpDX.DirectInput.DirectInput();
            listaUrzadzen = new List<DeviceInstance>();
            var listaJoystickow = directInput.GetDevices(DeviceType.Joystick, DeviceEnumeratio
            var listaGamepadow = directInput.GetDevices(DeviceType.Gamepad, DeviceEnumerationF

            //dodanie urzadzen do listy
            foreach (DeviceInstance joystick in listaJoystickow)
                listaUrzadzen.Add(joystick);



            foreach (DeviceInstance gamepad in listaGamepadow)
                listaUrzadzen.Add(gamepad);

            //dodanie urzadzen z listy do listBoxa
            foreach (DeviceInstance urzadzenie in listaUrzadzen)
                listBox1.Items.Add(urzadzenie.InstanceName);
        }
```

4 Ilustrowanie działania joysticka

Do testowania urządzenia służy klasa Pozycje. Konstruktor inicjalizuje komponenty graficzne, przypisuje obiekt joystick oraz tworzy timer potrzebny do prawidłowego odczytania wartości urządzenia przy pomocy wątku.

```
public Pozycje(Joystick joystick)
        {
            InitializeComponent();
            this.joystick = joystick;
            timer = new System.Threading.Timer(obj => Update());
        }
```

Metoda Update pobiera aktualne wartości kontrolera (joystick.GetCurrentState()) za pomocą biblioteki SharpDX.DirectInput i pokazuje je na ekranie w polu tekstowym.

```
private new void Update()
        {
            this.textBox1.Invoke((MethodInvoker)delegate
           {
               this.textBox1.Text = String.Format("{0}", joystick.GetCurrentState().X);
           });
            this.textBox2.Invoke((MethodInvoker)delegate
            {
                this.textBox2.Text = String.Format("{0}", joystick.GetCurrentState().Y);
            });
            this.textBox3.Invoke((MethodInvoker)delegate
            {
                this.textBox3.Text = "" + joystick.GetCurrentState().Z;
            });
            this.textBox4.Invoke((MethodInvoker)delegate
            {
                this.textBox4.Text = "" + joystick.GetCurrentState().Buttons[0];
            });
        }
```

5 Emulowanie myszki za pomocą joysticka

Do emulacji myszy za pomocą joysticka została użyta biblioteka WindowsInput, która pozwala na emulowanie zachowań myszy poprzez interfejs IMouseSimulator. W podobny sposób jak w klasie Pozycje wykorzystany został Timer do obsługi wątku.

```
public EmulacjaMyszy(Joystick joystick)
        {
            this.joystick = joystick;
            joystick.Acquire();
            mouseSimulator = new InputSimulator().Mouse;
            timer = new System.Threading.Timer(obj => Update());
            timer.Change(0, 1000 / 60);
        }
```

Metoda Update pobiera aktualny stan urządzenia (X,Y,FIRE), oblicza położenie kursora aby następnie przekazać pobrane wartości do metody mouseSimulator.MoveMouseBy(x, y). Aby obliczyć poprawną zmianę położenia kursora należy odjąć od odebranej wartości połowę maksymalnego wychylenia drążka (32767), a następnie podzielić tą wartość przez zmienną MovementDivider, która odpowiada za szybkość poruszania się kursora.

```
private void Update()
        {
                //obliczenie polozenia wspolrzednej x
                var x = (joystick.GetCurrentState().X - 32767) / MovementDivider;
                //obliczenie polozenia wspolrzednej y
                var y = (joystick.GetCurrentState().Y - 32767) / MovementDivider;
                //przekazanie x,y do emulatora myszy
                mouseSimulator.MoveMouseBy(x, y);
                //przycisk Fire jako lewy przycisk myszy
                if (joystick.GetCurrentState().Buttons[0]) mouseSimulator.LeftButtonDown();
                else mouseSimulator.LeftButtonUp();
                if (joystick.GetCurrentState().Buttons[1])mouseSimulator.RightButtonDown();
                 else mouseSimulator.RightButtonUp();


        }
```

6 Prosty edytor graficzny

Klasa Draw odpowiada za zarządzanie edytorem graficznym reprezentowanym jako bitmapa.

```
private void LoadDraw(object sender, EventArgs e)
        {
            bmp = new Bitmap(this.pictureBox1.Width, this.pictureBox1.Height);
        }
```

Metoda PictureBox1_Paint rysuje w edytorze czarny kursor do poruszania się za pomocą Graphics.DrawImage(...) oraz Graphics.FillEllipse(...)

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

Obsługa czyszczenia okna znajduje się w metodzie ClearScreen().

```
public void ClearScreen()
        {
            bmp = new Bitmap(this.pictureBox1.Width, this.pictureBox1.Height);
            lock (bmp)
            {
                using (Graphics g = Graphics.FromImage(bmp))
                using (SolidBrush brush = new SolidBrush(Color.White))
                {
                    g.FillRectangle(brush, 0, 0, szerokosc, wysokosc);
                }
                updated = true;
            }
        }
```

Zmiana poleżenia kursora w edytorze jest realizowana przez metodę ChangeXY(...).

```
//zmiana po o enia x,y wraz ze sprawdzeniem aby nie znalaz  si  poza oknem programu
        public void ChangeXY(int x, int y)
        {

            if (this.x + x > 0 && this.x + x < szerokosc)
                this.x = this.x + x;
            if (this.y + y > 0 && this.y + y < wysokosc)
                this.y = this.y + y;

            updated = true;
        }
```

Rysowanie w edytorze obsługuje metoda Drawing(...).

```
public void Drawing(int x, int y, int r)
        {
            lock (bmp)
            {
                using (Graphics g = Graphics.FromImage(bmp))
                using (SolidBrush brush = new SolidBrush(Color.Red))
                {
                    g.FillEllipse(brush, this.x - r, this.y - r, 2 * r, 2 * r);
                }
                updated = true;
            }
        }
```

Klasą pomocniczą dla Draw jest Draw_Brush, która odpowiada za pobranie stanu wychylenia drążka, slidera, przycisków 1,2.

```
public void InputThread()
        {
            draw.ClearScreen();
            while (draw.Visible)
            {
                //pozycja x
                x = (joystick.GetCurrentState().X - 32767) / MovementDivider;
                //pozycja y
                y = (joystick.GetCurrentState().Y - 32767) / MovementDivider;
                //pozycja slidera
                z = joystick.GetCurrentState().Z;
                //wcisniecie przycisku fire rozpoczyna rysowanie
                if (joystick.GetCurrentState().Buttons[0])
                {
                    z /= 5000;
                    draw.Drawing(x, y, z);
                }
                //wcisniecie przycisku 2 (button[1]) czysci edytor graficzny
                if (joystick.GetCurrentState().Buttons[1])
                {
                    draw.ClearScreen();
                }
                //zmiana polozenia kursora w edytorze graficznym
                draw.ChangeXY(x, y);
                Thread.Sleep(10);
            }
        }
```

7 Wnioski

Wykorzystując dostępne sterowniki Windows, biblioteki DirectInput oraz WindowsInput, udało się poprawnie wykryć dostępne urządzenia, co pozwoliło na realizację zadań zawartych w instrukcji laboratorium. Program został stworzony w języku C# i poprawnie realizuje wcześniej przedstawione funkcje.