Urządzenia Peryferyjne

**Ćwiczenie 10 – Obsługa joysticka USB z wykorzystaniem DirectInput**

Prowadzący: Dr Inż. Jan Nikodem

Grupa: Poniedziałek tydzień parzysty godz. 10:15

Data wykonania ćwiczenia: 13.11.2017r.

Wykonali : Paweł Biel 225949

Oskar Szubert 213624

# Wstęp

Na pierwszych zajęciach laboratoryjnych otrzymaliśmy zadanie polegające na napisaniu programu pozwalającego na prostą obsługę joysticka podłączonego do komputera  
za pomocą portu USB. Wymagane było, aby program:

* Stwierdzał obecność joysticka podłączonego do komputera
* Wyświetlał wszystkie podłączone joysticki w postaci MessageBox’ów

Kod naszego programu został napisany w języku C#.

# Zagadnienia

HAL / HEL :

## DirectX wykorzystuje dwa sterowniki, do sterowania sprzętem:

## -warstwę abstrakcji sprzętu Hardware Abstraction Layer (HAL)

## -warstwę emulacji sprzętowej Hardware Emulation Layer (HEL)

## Po zainicjowaniu DirectX sprawdza sprzęt, czy obsługuje on określone funkcje. Jeżeli je posiada wtedy zostanie użyta warstwa HAL, żeby uzyskać tą funkcjonalność. W przeciwnym przypadku warstwy HEL użyta będzie do emulacji potrzebnych funkcjonalności. Innymi słowy :

## HAL - wykorzystuje funkcjonalność sprzętową

## HEL - implementuje własną funkcjonalność. Pozwala to na łatwiejsze rozwijanie na kolejnych generacjach sprzętu.

## Port USB:

Uniwersalna magistrala szeregowa (Universal Serial Bus) to powszechny port umożliwiający komunikację pomiędzy komputerem, a dowolnym urządzeniem peryferyjnym. Opracowany został między innymi przez Microsoft, IBM i Intel. Urządzenia podłączone za pomocą portu USB mogą być automatycznie wykryte i rozpoznane przez system. Z upływem lat powstały kolejne generacje portu USB zwiększając prędkość transmisji danych przez ten port.

* + - * USB 1.1: do 1,5 MB / s
      * USB 2.0: do 60 MB / s
      * USB 3.1 gen. 1: do 625 MB / s
      * USB 3.1 gen. 2: do 1,25 GB / s

## HID (Human Interface Device):

Są to urządzenia podłączane do komputera, które wysyłają do niego informacje generowane przez człowieka np. mysz, klawiatura, joystick. Urządzenia HID obecnie produkowane są z myślą, aby móc je podłączać   
za pomocą portu USB.

## COM (Component Object Model):

Jest to standard definiowania i tworzenia interfejsów opracowany przez firmę Microsoft oraz w oparciu o zestaw adekwatnych bibliotek. Na podstawie standardu COM zostało opracowanych dużo niskopoziomowych API dla produktów firmy Microsoft – między innymi DirectX.

## Biblioteka DirectInput:

Powyższa biblioteka umożliwia obsługę urządzeń wejściowych takich jak mysz, klawiatura czy joystick. Wchodzi ona w skład zestawu funkcji API DirectX firmy Microsoft. Za jej pomocą możliwe jest odczytywanie informacji   
o podłączanych urządzeniach oraz przypisywanie im określonych akcji (przykładowo po wciśnięciu określonego przycisku). Biblioteka ta najczęściej stosowana jest podczas tworzenia gier komputerowych oraz aplikacji interaktywnych.

1. Kod programu
2. namespace JoyStrick
3. {
4. /// <summary>
5. /// Interaction logic for MainWindow.xaml
6. /// </summary>
7. public partial class MainWindow : Window
8. {
9. public MainWindow()
10. {
11. InitializeComponent();
12. CheckJoy();
13. Sticks = GetSticks();
14. System.Windows.Threading.DispatcherTimer dispatcherTimer = new System.Windows.Threading.DispatcherTimer();
15. dispatcherTimer.Tick += dispatcherTimer\_Tick;
16. dispatcherTimer.Interval = new TimeSpan(0, 0, 1);
17. dispatcherTimer.Start();
18. }
19. DirectInput Input = new DirectInput();
20. private SlimDX.DirectInput.Joystick stick;
21. private Joystick[] Sticks;
23. private bool mouseClicked = false;
24. private int yvalue = 0;
25. private int xvalue = 0;
26. private int zvalue = 0;
27. [DllImport("User32.dll")]
28. private static extern bool SetCursorPos(int X, int Y);
29. [DllImport("User32.dll")]
30. public static extern void mouse\_event(uint flag, uint \_x, uint \_y, uint btn, uint exInfo);
31. private const int MOUSEEVENT\_LEFTDOWN = 0x02;
32. private const int MOUSEEVENT\_LEFTUP = 0x04;
34. private void Window\_Loaded(object sender, RoutedEventArgs e)
35. {
36. SetCursor(200, 200);
37. }
38. private static void SetCursor(int x, int y)
39. {
40. // Powrócenie kursora myszy do początkowego położenia
41. var xL = (int)App.Current.MainWindow.Left;
42. var yT = (int)App.Current.MainWindow.Top;
43. SetCursorPos(x + xL, y + yT); //
44. }
45. public void CheckJoy()
46. {
48. var directInput = new DirectInput();
50. var joystickGuid = Guid.Empty;
51. List<string> names = new List<string>();
53. foreach (var deviceInstance in directInput.GetDevices(DeviceType.Gamepad,
54. DeviceEnumerationFlags.AllDevices))
55. {
56. joystickGuid = deviceInstance.InstanceGuid;
57. names.Add( deviceInstance.InstanceName); // dodanie Gamepad'a do listy urządzeń
58. }
60. foreach (var deviceInstance in directInput.GetDevices(DeviceType.Joystick,
61. DeviceEnumerationFlags.AllDevices))
62. {
63. joystickGuid = deviceInstance.InstanceGuid;
64. names.Add(deviceInstance.InstanceName); // dodanie Joysticka do listy urządzeń
65. }
66. // Jeśli Joystick lub Gamepad nie jest podłączony wyświetl powiadomienie o braku podłączonych urządzeń
67. if (joystickGuid == Guid.Empty)
68. {
70. MessageBox.Show("Joystick nie jest podlaczony");
71. Environment.Exit(1);
72. }
73. else
74. {
75. // wypisz wszystkie Gamepady i Joysticki
76. foreach (string i in names)
77. {
78. MessageBox.Show(i);
79. }
81. }
83. }
84. // funkcja pobierająca informacje o przyciskach
85. public Joystick[] GetSticks()
86. {
87. var sticks = new List<Joystick>();
88. foreach (DeviceInstance device in Input.GetDevices(DeviceClass.GameController, DeviceEnumerationFlags.AttachedOnly))
89. {
90. try
91. {
92. stick = new SlimDX.DirectInput.Joystick(Input, device.InstanceGuid);
93. stick.Acquire();
94. foreach (DeviceObjectInstance deviceObject in stick.GetObjects())
95. {
96. if ((deviceObject.ObjectType & ObjectDeviceType.Axis) != 0)
97. {
98. stick.GetObjectPropertiesById((int)deviceObject.ObjectType).SetRange(-100,100);
99. }
100. }
101. sticks.Add(stick);
102. }
103. catch (DirectInputException)
104. {
106. }
107. }
108. return sticks.ToArray();
109. }
110. void stickHandle(Joystick stick, int id)
111. {
112. JoystickState state = new JoystickState();
113. state = stick.GetCurrentState();
114. yvalue = state.Y;
115. xvalue = state.X;
116. zvalue = state.Z;
117. SetCursor(xvalue,yvalue);
118. bool[] buttons = state.GetButtons();
119. if (id == 0)
120. {
121. if (buttons[0])
122. {
123. if (mouseClicked == false)
124. {
125. mouse\_event(MOUSEEVENT\_LEFTDOWN,0,0,0,0);
126. mouseClicked = true;
128. }
129. }
130. else
131. {
132. if (mouseClicked)
133. {
134. mouse\_event(MOUSEEVENT\_LEFTUP, 0, 0, 0, 0);
135. mouseClicked = false;
137. }
139. }
140. }
141. }
142. private void dispatcherTimer\_Tick(object sender, EventArgs e)
143. {
144. for (int i = 0; i < Sticks.Length; i++)
145. {
146. stickHandle(Sticks[i], i);
147. }
148. }
149. }
150. }

4.Podsumowanie

Podczas pierwszych zajęć laboratoryjnych odświeżyliśmy naszą wiedzę na temat portów szeregowych oraz o tym jak one działają. Zapoznaliśmy się również z podstawami DirectX oraz o tym w jaki komputer rozpoznaje urządzenia, które są do niego podłączane za pomocą portu USB. Napisanie kodu programu oraz analiza przykładów zawartych w różnych źródłach pozwoliła nam lepiej zrozumieć sposób komunikacji pomiędzy komputerem a urządzeniami peryferyjnymi. Niestety nie udało nam zrealizować wszystkich poleceń ze względu na pierwszą styczność z tego typu ćwiczeniami oraz   
z językiem C#.