Baraniecki Karol Byczko Maciej	Prowadzący: Dr inż. Dominik Żelazny	Numer ćwiczenia laboratoria 10
PT 16:30 TP	Temat ćwiczenia: USB, DirectInput	Ocena:
Grupa: D	Data wykonania: 25 października 2021	

1 Zadania do opracowania

Teoretyczny opis elementów używanych do bezpośredniej kontroli (Direct Input)

1.1 USB w Windows

1.1.1 Warstwa HAL i HEL

HAL - Hardware Abstraction Layer, "nieprzenośna" część jądra NT.
HEL - Hardware Emulation Library, jak sama nazwa mówi, służy do emulacji urządzeń, problem ze znalezieniem dokumentacji, raczej już nie używana.

1.1.2 Standard USB oraz USB 2.0

USB - Universal Serial Bus, po polsku Uniwersalna Magistrala Szeregowa, uniwersalny interfejs komunikacyjny typu plug and play. Obecnie najnowszą wersją jest USB 3.2 wyprodukowany w 2017 roku.

1.1.3 Interface HID

HID - Human Interface Devices, urządzenia do wprowadzania danych przez człowieka. nazwa kodowa dla urządzeń peryferyjnych służących do wprowadzania informacji do komputera, takich jak dżojstik, mysz, trackball czy klawiatura.

1.2 DirectInput

1.2.1 DirectX

zestaw funkcji API wspomagających generowanie grafiki (dwuwymiarowej i trójwymiarowej), dźwięku oraz innych zadań związanych zwykle z grami komputerowymi i innymi aplikacjami multimedialnymi.

1.2.2 Model COM

COM - Component Object Model, standard definiowania i tworzenia interfejsów programistycznych na poziomie binarnym dla komponentów oprogramowania wprowadzony przez firmę Microsoft wraz z bibliotekami zapewniającymi podstawowe ramy i usługi dla współdziałania komponentów COM i aplikacji.

2 Zadania do wykonania

1. Napisać program, odczytujący nazwę zainstalowanego joysticka
2. Napisać program ilustrujący działanie joysticka: stwierdzający naciśnięcie myszy, zmianę położenia drążka (w przestrzeni 2D) oraz suwaka.
3. Napisać program zastępujący działanie myszy. Program ma umożliwiać sterowanie kursorem za pomocą joysticka oraz obsługę przycisków fire jako kliknięć myszy.
4. Napisać program realizujący prosty edytor graficzny - rysowanie przy pomocy Joysticka

2.1 Odczytywanie nazwy joysticka

```
pygame.joystick.init()
pad = pygame.joystick.Joystick(0)
pad_name = pad.get_name()
```

2.2 Kontrola myszki za pomocą joysticka

Do wykonania tego zadania wykorzystaliśmy biblioteki *Mouse* oraz *PyGame*

```
import time
import mouse
import pygame
import sys
pygame.init()

joystick_count = pygame.joystick.get_count()
print("Number of joysticks: {}".format(joystick_count) )

if joystick_count == 0:
  print("No joysticks found")
  sys.exit(0)

joystick = pygame.joystick.Joystick(0)

is_pressed = mouse.is_pressed()
is_pressed_right = mouse.is_pressed(button='right')

while True:
  pygame.event.pump()
  time.sleep(1.0 / 30.0)
  mult = 3
  mouse.move((mult * joystick.get_axis(0)) ** 3, (mult * joystick.
     get_axis(1)) ** 3, absolute=False)

  if joystick.get_button(0) and not is_pressed:
    is_pressed = True
    mouse.press()

  if not joystick.get_button(0) and is_pressed:
```

```
30      is_pressed = False
31      mouse.release()
32
33    if joystick.get_button(2) and not is_pressed_right:
34      is_pressed_right = True
35      mouse.press(button='right')
36
37    if not joystick.get_button(0) and is_pressed_right:
38      is_pressed_right = False
39      mouse.release(button='right')
```

2.3 Paint

2.4 odczytanie nazwy zainstalowanego joysticka

```
import pydirectinput
import pygame
import random

# class for drawing circles
class Circle:
    def __init__(self, x, y, radius, color):
        self.x = x
        self.y = y
        self.radius = radius
        self.color = color

    def draw(self, surface):
        pygame.draw.circle(surface, self.color, (self.x, self.y),
                           self.radius)


pygame.init() # initialize
pad = pygame.joystick.Joystick(0) # get joystick
width, height = 1980, 900 # set window size
screen = pygame.display # display screen
surface = screen.set_mode((width, height)) # set surface

pad_name = pad.get_name() # ged pad name
pygame.display.set_caption(f"{pad_name} - drawing") # display pad
   name

# cursor position, default center
position_x = width / 2
position_y = height / 2

speed = 0.5 # cursor speed
running = True # while false, run program
drawing = False # if true, draw

color = (0, 0, 0) # default color

radius = 25 # cursor radius

circles = [] # drawed circles

#main loop
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        if event.type == pygame.JOYBUTTONDOWN:
```

```
47              if pad.get_button(0):
48                  drawing = True
49              if pad.get_button(1):
50                  speed *= 0.5 # set speed to slower
51              if pad.get_button(2):
52                  speed *= 2 # set speed to higher
53              if pad.get_button(3): # set random color
54                  color = (random.randrange(0, 255),
55                           random.randrange(0, 255),
56                           random.randrange(0, 255))
57              if pad.get_button(5): # change cursor size
58                  if radius != 1:
59                      radius -= 1
60              if pad.get_button(6):
61                  radius += 1
62          if event.type == pygame.JOYBUTTONUP: # if fire clicked then
                draw
63              if not pad.get_button(0):
64                  drawing = False
65
66      surface.fill((255, 255, 255)) # fill cursor
67
68      # draw all circles
69      for circle in circles:
70          circle.draw(surface)
71
72      # change cursor position
73      position_x += pad.get_axis(0) * speed
74      position_y += pad.get_axis(1) * speed
75
76      # display cursor
77      pygame.draw.circle(surface, (0, 0, 0), (position_x, position_y),
78                         radius)
79
80      # add new circles
81      if drawing:
82          circles.append(Circle(position_x, position_y, radius, color)
            )
83
84      screen.update()
```

2.4.1 Dowód działania programu graficznego