Réalisation d'une application de détection d'HID

Noms : HAMAN OUMAROU
Classe : AIA4
Matricule : 22p124

Année académique : 2023-2024

♣ Table des matières ♣

★ ★ CHAPITRE UN ★ ★

INTRODUCTION GÉNÉRALE

L'avènement de la technologie informatique a profondément influencé notre façon d'interagir avec les ordinateurs. Les périphériques d'entrée tels que les souris, les claviers ou les joysticks jouent un rôle essentiel dans cette interaction, permettant aux utilisateurs de contrôler et de communiquer avec les systèmes informatiques. Dans le domaine du développement logiciel, la détection précise de ces périphériques HID revêt une importance capitale pour assurer une expérience utilisateur fluide et efficace. Le présent rapport se concentre sur la conception et l'implémentation d'une application de détection des HID. L'objectif principal de cette application est d'identifier les claviers, souris et joysticks connectés à un système informatique et de fournir des informations détaillées sur ceux-ci. Au cours de ce rapport, nous décrirons en détail notre approche de conception et d'architecture pour l'application de détection de ces HID. Nous aborderons également les étapes d'implémentation et de développement, en mettant en évidence les technologies et les outils utilisés. De plus, nous présenterons les résultats de nos tests et les performances de l'application, ainsi que les perspectives d'amélioration et d'extension pour les projets futurs.

★ ★ CHAPITRE DEUX ★ ★

CONCEPTION ET RÉALISATION

Introduction

Notre travail a pour ambition de répondre à un besoin croissant dans le domaine de l'interaction homme-machine. Avec la prolifération des périphériques HID et leur utilisation généralisée dans divers domaines, il est devenu essentiel de disposer d'une application permettant de détecter et de gérer ces périphériques de manière transparente.

2.1 Analyse des besoins

Les principaux objectifs de notre projet sont donc les suivants :

1. Concevoir une application capable de détecter automatique des périphériques HID :
   — L'application doit être capable de détecter automatiquement les périphériques HID connectés à l'ordinateur.
   — Elle devrait pouvoir identifier les différents types de périphériques HID, tels que les souris, les claviers et les joysticks.
2. Assurer le suivi des événements générés par ces périphériques :
   — L'application doit être en mesure de suivre les événements générés par les périphériques HID, tels que les clics de souris, les frappes de clavier et les mouvements et frappes de touches des joystick.
   — Elle devrait pouvoir enregistrer et traiter ces événements de manière appropriée.
3. Fournir une interface utilisateur conviviale :
   — L'application devrait fournir une interface utilisateur permettant aux utilisateurs de visualiser les périphériques détectés et les événements associés.

2.2 Conception et architecture

2.2.1 Identification des composants et de leurs responsabilités

La conception de l'application repose sur quatre principaux composants :

— **La logique de détection des périphériques** : Ce composant est chargé de détecter tous les périphériques connectés.
— **L'identificateur de périphériques** : Ce composant est responsable de l'identification des périphériques détectés. Il détermine s'il s'agit d'une souris, d'un clavier, d'un joystick, etc.
— **L'interface utilisateur** : C'est l'endroit où les périphériques détectés ainsi que les informations correspondantes seront affichés.
— **La gestion des évènements des périphériques** : Ce composant écoute les périphériques connectés et traduit tous les évènements survenus en une forme visuelle compréhensible par l'utilisateur.

2.2.2 Flux de données

À partir de l'interface utilisateur, une demande de périphériques connectés est initiée vers la logique de détection de périphériques. Cette dernière vérifie si des périphériques sont connectés. Si tel est le cas, elle les transmet à l'identificateur de périphériques. À son tour, l'identificateur de périphériques identifie les périphériques reçus et envoie à l'interface utilisateur les images et les informations de chaque périphérique qui lui a été transmis. Il retransmet également les périphériques qu'il a identifiés au gestionnaire des périphériques. Ce dernier, à intervalles réguliers, envoie l'état des périphériques à l'interface utilisateur.

La figure ci-dessous illustre cette transmission des données entre les différents composants :

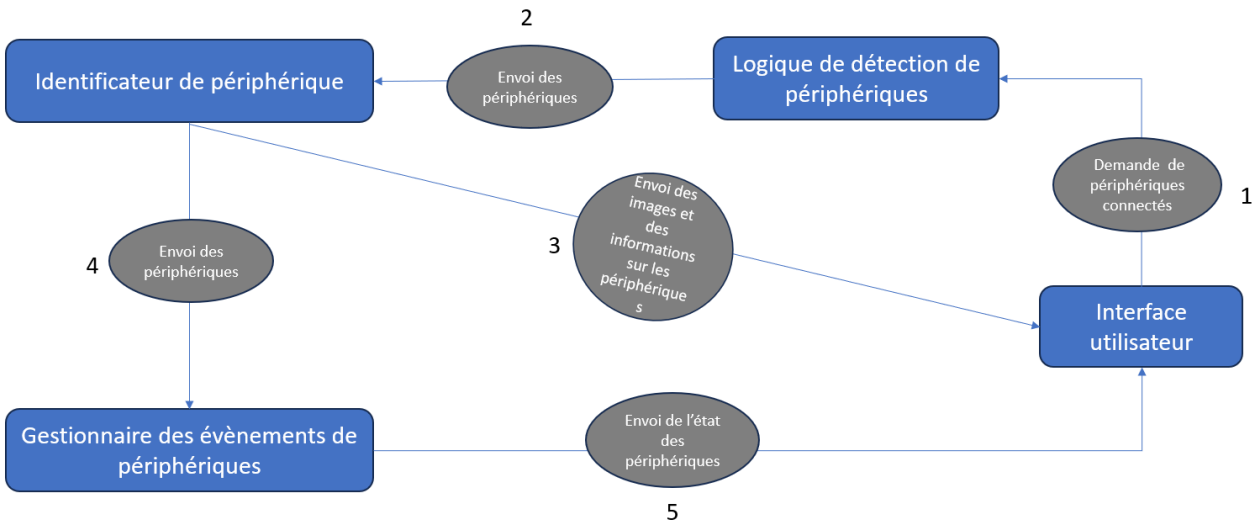


FIGURE 2.1 – Flux de données

2.3 Implémentation et développement

2.3.1 Choix de technologies

1. Pour ce projet, nous avons choisi d'utiliser le de programmation langage **Python** en raison de ses nombreux avantages. Voici quelques raisons qui ont motivé ce choix :
   - **Simplicité et lisibilité** : Python est réputé pour sa syntaxe claire et sa facilité de compréhension. Cela permet de développer du code propre et maintenable, ce qui est essentiel pour un projet de détection des périphériques HID.
   - **Large écosystème de bibliothèques** : Python dispose d'une vaste gamme de bibliothèques spécialisées, telles que PyUSB, Pygame, PyWin32 et pywinusb, qui facilitent l'accès aux périphériques HID et simplifient la mise en œuvre de la détection. Ces bibliothèques offrent des fonctionnalités prêtes à l'emploi et permettent d'économiser du temps et des efforts de développement.
   - **Portabilité** : Python est un langage multiplateforme, ce qui signifie que le code développé peut être exécuté sur différents systèmes d'exploitation tels que Windows, macOS et Linux. Cela permet une plus grande flexibilité et facilite le déploiement sur différentes machines.
   - **Support de la communauté** : Python bénéficie d'une large communauté de développeurs actifs, ce qui signifie qu'il est facile de trouver des ressources, des tutoriels et des exemples de code pour résoudre les problèmes ou obtenir de l'aide en cas de besoin. Cela facilite le développement et assure une progression plus rapide du projet.

2. Pour l'aspect graphique, nous avons utilisé **pygame** qui est une bibliothèque Python populaire et largement utilisée pour le développement de jeux et d'applications graphiques. Elle offre des fonctionnalités puissantes pour créer des interfaces interactives et visuellement attrayantes. En utilisant pygame, nous avons pu concevoir une interface utilisateur fluide et agréable, qui répond aux besoins des utilisateurs et améliore leur expérience globale.
3. Pour la détection des périphériques nous avons utilisé la bibliothèque **pywinusb**.

2.3.2 Fonctionnalités implémentées

> **Interface utilisateur conviviale** : Notre application dispose d'une interface utilisateur conviviale et intuitive. Les utilisateurs peuvent facilement visualiser les informations sur les périphériques détectés, ainsi que les événements HID enregistrés. L'interface a été conçue pour offrir une expérience utilisateur fluide et agréable. Cet interface à été réalisé en utilisant le module pygame.

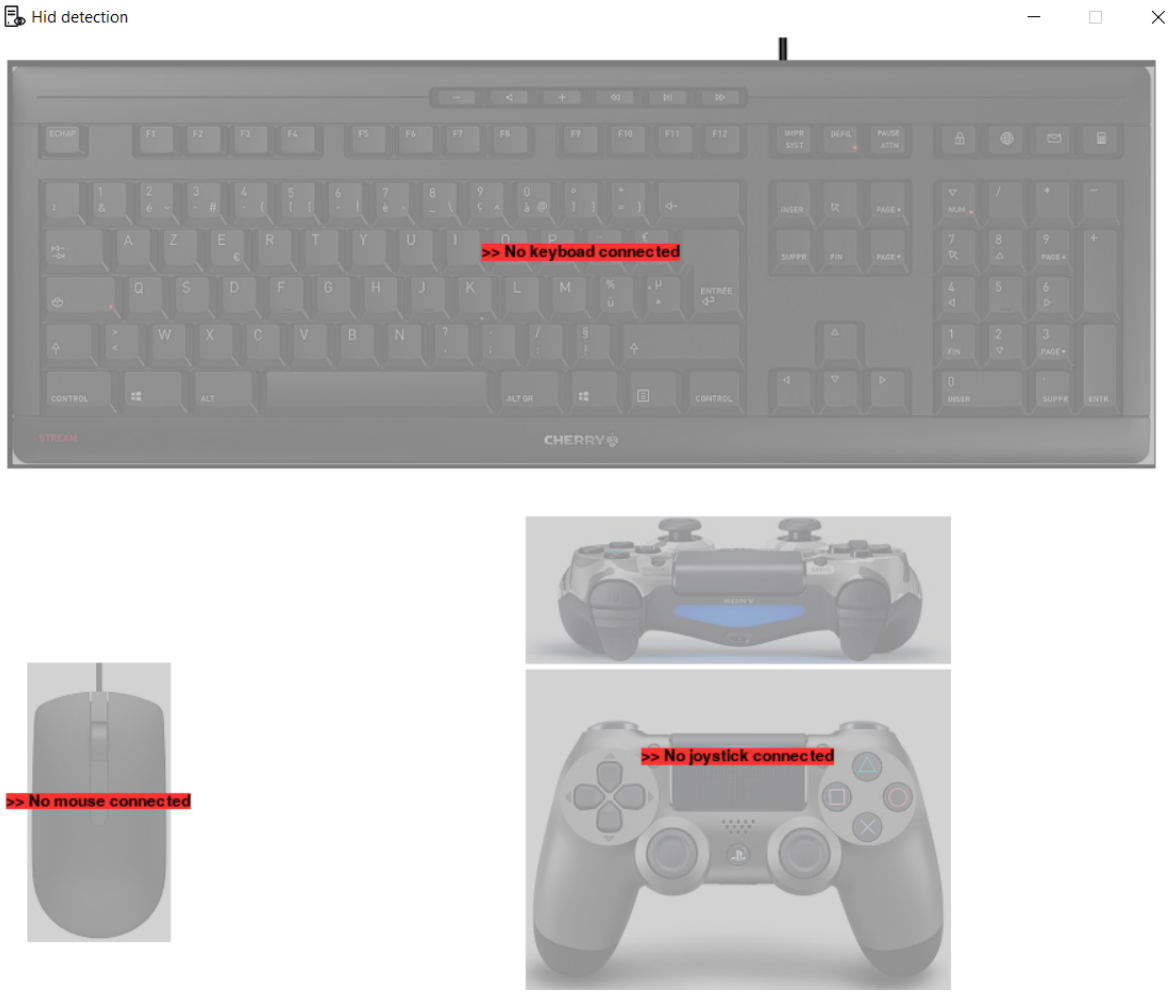


FIGURE 2.2 – Interface utilisateur

> **Détection des périphériques HID connectés** : Notre application est capable de détecter automatiquement les périphériques HID connectés à l'ordinateur. Lorsqu'un périphérique est branché, il est immédiatement identifié et ajouté à la liste des périphériques détectés.
> **Affichage des informations sur les périphériques** : Une fois que les périphériques HID sont détectés, notre application affiche des informations détaillées sur chaque périphérique. Les utilisateurs peuvent consulter le nom du périphérique, le type, le fournisseur et d'autres informations pertinentes pour identifier facilement les périphériques connectés.

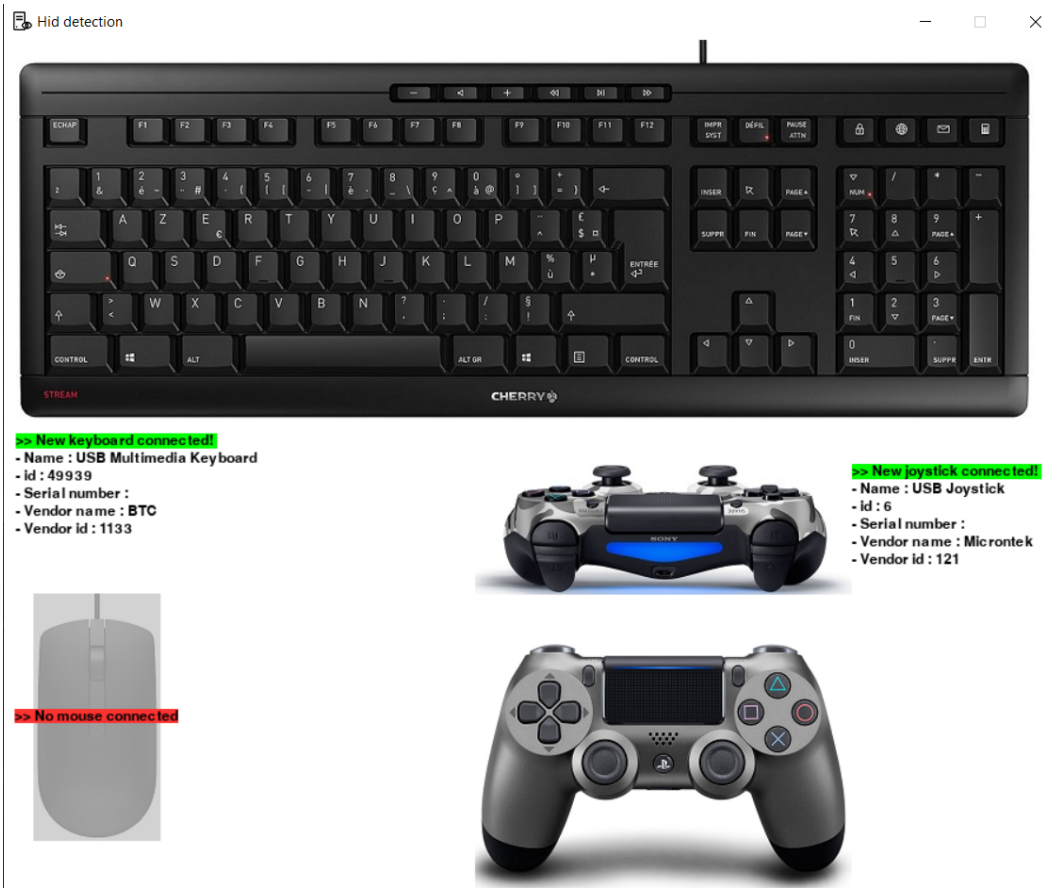
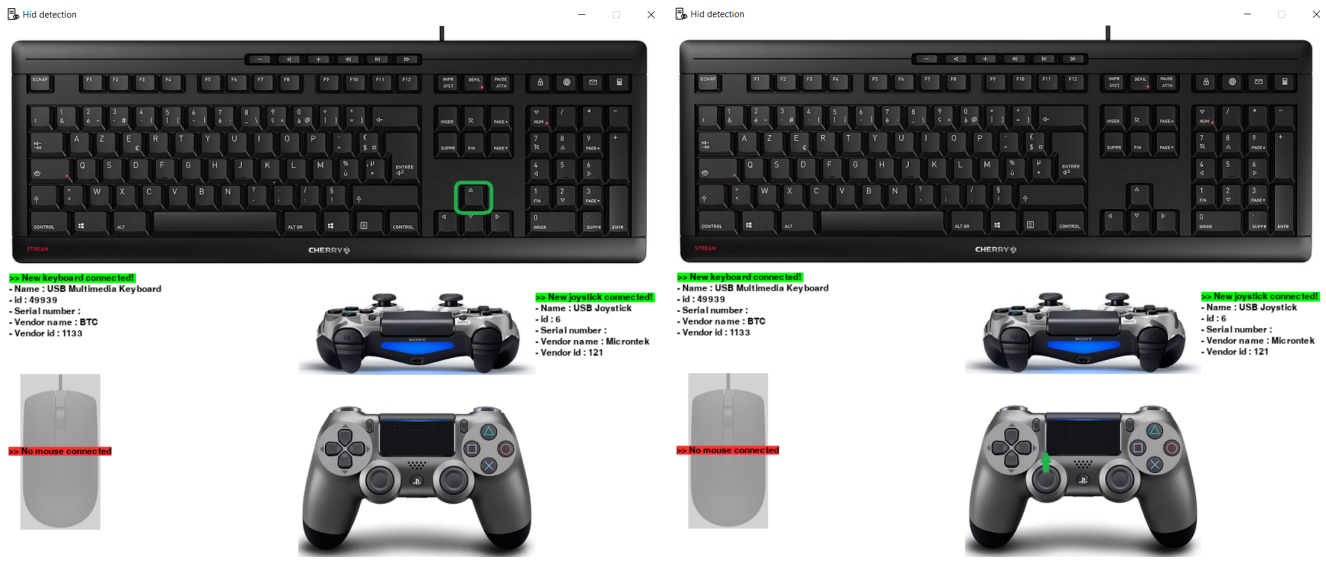


FIGURE 2.3 – Périphériques détectés

> **Suivi des événements HID** : Nous avons mis en place un système de suivi des événements générés par les périphériques HID. Notre application capture les clics de souris, les pressions de touches et d'autres types d'événements émis par les périphériques. Ces événements sont enregistrés et utilisés pour des analyses ultérieures.



(a) Événement du clavier

(b) Événement du joystick

FIGURE 2.4 – Gestion des évènement de périphériques

2.4 Problèmes rencontrés

Au niveau de la gestion des événements, nous avons rencontré plusieurs problèmes. Les principaux problèmes rencontrés sont les suivants :

1. **Différences entre les périphériques** : En raison de la diversité des claviers, des souris et des joysticks disponibles sur le marché, il est possible qu'une touche soit présente sur un périphérique et absente sur d'autres du même type. Par exemple, certaines souris peuvent avoir des boutons supplémentaires qui ne sont pas présents sur d'autres modèles. Cette variation entre les périphériques peut entraîner des incohérences dans la gestion des événements et rendre difficile l'écriture de code générique qui fonctionne avec tous les périphériques.
2. **Différences de configuration** : Les claviers, les souris et les joysticks peuvent avoir des configurations différentes en termes de touches et de boutons. Par exemple, la disposition des touches sur les claviers peut varier en fonction du type de clavier. Cela peut entraîner des erreurs dans la détection des touches et des boutons, ainsi que dans la gestion des événements correspondants. Il est important de prendre en compte ces différences de configuration pour assurer un fonctionnement correct de l'application sur différents périphériques.

2.5 Perspectives

Comme perspective, nous pouvons ajouter à notre application les composantes suivants :

1. **Analyse des données HID** : L'application peut effectuer une analyse plus approfondie des données HID reçues. Par exemple, elle peut détecter des motifs de frappe spécifiques sur un clavier, des mouvements de souris inhabituels, ou des commandes spéciales sur un joystick. Cela peut être utile pour des tâches telles que la détection d'intrusion ou la surveillance de l'utilisation des périphériques.
2. **Options de configuration** : Il peut être intéressant d'inclure des options de configuration, telles que la possibilité d'activer ou de désactiver la détection de certains types de périphériques, de filtrer les événements spécifiques, ou de définir des seuils pour les alertes ou les déclencheurs d'action.
3. **Tests et rétroaction des utilisateurs** : Effectuez des tests approfondis avec différents types de périphériques pour vous assurer que votre application fonctionne correctement avec chaque configuration. Encouragez les utilisateurs à fournir des commentaires sur leur expérience d'utilisation avec différents périphériques. Cela vous permettra d'identifier les problèmes potentiels et d'apporter des améliorations continues à votre application.

★ ★ CHAPITRE TROIS ★ ★

CONCLUSION

En conclusion, notre projet de détection HID a été une expérience enrichissante qui nous a permis de comprendre et de résoudre les défis liés à la gestion des événements des périphériques. Nous avons rencontré des problèmes liés aux différences de configuration des claviers, des souris et des joysticks. Pour surmonter ces problèmes, nous avons mis en place une logique de détection des périphériques qui nous a permis d'identifier les types de périphériques connectés et de gérer leurs configurations spécifiques. En fin de compte, grâce à nos efforts de développement, nous avons réussi à créer une application de détection HID qui offre une expérience utilisateur fluide, en prenant en compte les variations entre les périphériques. Nous sommes fiers du travail accompli et nous sommes confiants que notre application sera utile pour les utilisateurs souhaitant interagir avec différents périphériques HID.

★ ★ CHAPITRE QUATRE ★ ★

ANNEXE(CODE SOURCE)

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Mar  7 05:37:33 2024

@author: Leon-Onetype
"""

import pygame
import pywinusb.hid as hid

# Taille de la fenetre
WIN_WIDTH   =  900
WIN_HEIGHT  = 720

# Les couleurs
BACKGROUND_COLOR   = (255, 255, 255)
WHITE_COLOR        = (255, 255, 255)
BLACK_COLOR        = (0,   0,   0)
GREEN_COLOR        = (0,   255, 0)
BLUE_COLOR         = (0,   0,   180)
RED_COLOR          = (255, 50,   50)

# Les positions et les dimentions des object sur l'écran
MOUSE_DEST         = (25, 470)
IMG_MOUSE_SIZE     = (108, 209)

KEYBOARD_DEST      = (10, 0)
IMG_KEYBOARD_SIZE  = (864, 324)

JOYSTICK_FRONT_DEST= (400, 475)
JOYSTICK_BACK_DEST=  (400 , 360)
IMG_JOYSTICK_FRONT_SIZE  = (320, 250)
IMG_JOYSTICK_BACK_SIZE  = (320, 111)

FPS = 30


class Device():
    """

    Device class

    """
    def __init__(self, product_name, product_id, vendor_name, vendor_id, serial_number ):
        self.product_name = product_name.strip()
        self.product_id = int(product_id)
        self.vendor_name = vendor_name.strip()
        self.vendor_id = int(vendor_id)
        self.serial_number = serial_number

    def __eq__(self, other_device):

        # On a supposer que que deux devices sont égaux s'ils ont le meme id
        return  self.product_id == other_device.product_id

def remove_duplicate(iterable):
    """

    Returns a new iterable  with non duplicated values
    """
```

```
    new_iterable = [iterable[0]]

    for element in iterable:
        if element not in new_iterable:
            new_iterable.append(element)

    return new_iterable

def parse_devices():
    """
    Detect new connected devices and devices deconnected

    Returns
    -------
     booleans values that determinate if keyboards, mouses and joysticks is connected or not

    """

    # On recupère la liste des devices connectés
    tmp_devices = hid.HidDeviceFilter().get_devices()

    # On converti ces devices en instances de la notre class Device
    tmp_devices = [Device(hid_device.product_name, hid_device.product_id, hid_device.vendor_name
                  for hid_device in tmp_devices]

    tmp_devices = remove_duplicate(tmp_devices)

    return tmp_devices

def display_text(text, pos, screen, color = WHITE_COLOR, font_size=12, center= False):
    """
    Display text on the Surface object passed as parameter
    """
    # text setting
    font_obj = pygame.font.Font('freesansbold.ttf', font_size)
    text_surface = font_obj.render(text, True, BLACK_COLOR, color)
    text_rect = text_surface.get_rect()
    position = pos
    if center == True:
        position = pos[0] - text_rect.width/2,  pos[1] - text_rect.height/2
    text_rect = pygame.Rect(*position, text_rect.width, text_rect.height)
    screen.blit(text_surface, position)

def display_device_informations(device, pos, screen, decalage):
    pos = pos[0], pos[1] + decalage
    display_text(f"- Name : {device.product_name[:50]}", pos, screen)
    pos = pos[0], pos[1] + decalage
    display_text(f"- id : {device.product_id}", pos, screen)
    pos = pos[0], pos[1] + decalage
    display_text(f"- Serial number : {device.serial_number }", pos, screen)
    pos = pos[0], pos[1] + decalage
    display_text(f"- Vendor name : {device.vendor_name}", pos, screen)
    pos = pos[0], pos[1] + decalage
    display_text(f"- Vendor id : {device.vendor_id}", pos, screen)

def display_devices_informations(devices, screen):
    """

    Parameters
    ----------
    devices : list
```

```
    list of devices(instances of the Device Classe).
screen :

Returns
-------
keyboard_connected : bool
    His value will be True if a keyboard is connected False if not.
mouse_connected : bool
    His value will be True if a mouse is connected False if not..
joystick_connected : bool
    His value will be True if a joystick is connected False if not..

"""
# Affichage des informations sur les devices détectés
pos = None
keyboard_connected = False
mouse_connected = False
joystick_connected = False
num_keyboard = 0
num_mouse = 0
num_joystick = 0

# Si on trouve au moins un périphérique
if devices:

    # Le décalage en ordonnées ou en abscices quand on va afficher les informations sur le p
    decalage=15

    for i, device in enumerate(devices):

        # Gestion selon le type de périphérique

        # Le cas d'une clavier
        if "keyboard" in device.product_name.lower():
            pos = KEYBOARD_DEST[0] + num_keyboard*25 , IMG_KEYBOARD_SIZE[1] + 10
            display_text(">> New keyboard connected! ", pos, screen, color=GREEN_COLOR)
            display_device_informations(device, pos, screen, decalage)
            keyboard_connected = True
            num_keyboard += 1

        # Le cas d'une souris
        elif "mouse" in device.product_name.lower():
            pos = MOUSE_DEST[0] + IMG_MOUSE_SIZE[0], MOUSE_DEST[1] + num_mouse*decalage
            display_text(">> New mouse connected! ", pos, screen, color=GREEN_COLOR)
            display_device_informations(device, pos, screen, decalage)
            mouse_connected = True
            num_mouse += 1

        # Le  d'une manette de jeu
        elif "joystick" in device.product_name.lower():
            pos = JOYSTICK_BACK_DEST[0] + IMG_JOYSTICK_BACK_SIZE[0], JOYSTICK_BACK_DEST[1] +
            display_text(">> New joystick connected! ", pos, screen, color=GREEN_COLOR)
            display_device_informations(device, pos, screen, decalage)
            joystick_connected = True
            num_joystick += 1

        else:
            pass

# Dans le cas contraire.
if keyboard_connected == False:
```

```
        pos = KEYBOARD_DEST[0] + IMG_KEYBOARD_SIZE[0]/2, KEYBOARD_DEST[1] + IMG_KEYBOARD_SIZE[1]
        display_text(">> No keyboad connected", pos, screen, RED_COLOR, center=True)

    if mouse_connected == False:
        pos = MOUSE_DEST[0] + IMG_MOUSE_SIZE[0]/2, MOUSE_DEST[1] + IMG_MOUSE_SIZE[1]/2
        display_text(">> No mouse connected", pos, screen, RED_COLOR, center=True)

    if joystick_connected == False:
        pos = JOYSTICK_BACK_DEST[0] + IMG_JOYSTICK_BACK_SIZE[0]/2, JOYSTICK_BACK_DEST[1]\
                            + (IMG_JOYSTICK_FRONT_SIZE[1] + IMG_JOYSTICK_BACK_SIZE[1])/2
        display_text(">> No joystick connected", pos, screen, RED_COLOR, center=True)

    # Actualisation de la surface
    pygame.display.update()

    return keyboard_connected,  mouse_connected, joystick_connected

def on_key_press(event, screen):
    """

    function to call when a key is pressed
    Parameters
    ----------
    event : pygame.event.Event
    screen : pygame.surface.Surface

    Returns
    -------
    None.

    """
    image_key = None

    try:
        texte = "clavier_images/" + '_'.join(pygame.key.name(event.key).lower().split(' ')) + "
        image_key = pygame.image.load(texte).convert_alpha()
    except FileNotFoundError:
        # On regarde d'abord si c'est une touche particulière
        particuliers =  {':':"deux_points", '\\':"antislash",'[/]':"slash", '[*]':"etoile", '?':
        if pygame.key.name(event.key) in particuliers.keys():
             texte = "clavier_images/" + particuliers[pygame.key.name(event.key)] + ".png"
             image_key = pygame.image.load(texte).convert_alpha()

        # Sinon, on ne fais rien
        else:
            image_key = pygame.image.load("clavier_images/connected.png").convert_alpha()

    screen.blit(image_key, KEYBOARD_DEST)

    pygame.display.update()

def on_click(event, screen):
    """

    function to call when a key is pressed
    Parameters
    ----------
    event : pygame.event.Event
    screen : pygame.surface.Surface

    Returns
```

```
    -------
    None.

    """

    path = "souris_images/connected.png"

    if event.button == 1:
        path = "souris_images/Button.left.png"
    elif event.button == 2:
        path = "souris_images/Button.middle.png"
    elif event.button == 3:
        path = "souris_images/Button.right.png"

    image_key = pygame.image.load(path).convert_alpha()
    screen.blit(image_key, MOUSE_DEST)
    pygame.display.update()

def on_joystick_pres(event, screen):

    button = event.button

    try:
        image = pygame.image.load(f"joystick_images/{button}.png").convert_alpha()
        if button in [4, 5, 6, 7]:
            screen.blit(image, JOYSTICK_BACK_DEST)
        else:
            screen.blit(image, JOYSTICK_FRONT_DEST)
    except:
        pass

    pygame.display.update()

def on_joystick_motion(event, screen):
    axis = event.axis
    value = event.value
    path = "joystick_images/connected_front.png"

    if axis == 0:
        if value <= -1.0:
            path = "joystick_images/l3_gauche.png"
        elif value >=1.0:
            path = "joystick_images/l3_droite.png"

    elif axis == 1:
        if value <= -1.0:
            path = "joystick_images/l3_haut.png"
        elif value >=1.0:
            path = "joystick_images/l3_bas.png"

    elif axis == 2:
        if value <= -1.0:
            path = "joystick_images/r3_gauche.png"
        elif value >=1.0:
            path = "joystick_images/r3_droite.png"

    elif axis == 3:
        if value <= -1.0:
            path = "joystick_images/r3_haut.png"
        elif value >=1.0:
            path = "joystick_images/r3_bas.png"
```

```
    image = pygame.image.load(path).convert_alpha()
    screen.blit(image, JOYSTICK_FRONT_DEST)


def main():
    # Initialisation du module pygame
    pygame.init()

    # Création de la fénètre et définition du titre
    screen = pygame.display.set_mode(size=(WIN_WIDTH, WIN_HEIGHT))
    pygame.display.set_caption("Hid detection")
    screen.fill(BACKGROUND_COLOR)
    # Icone
    icon = pygame.image.load('icone.png')
    pygame.display.set_icon(icon)

    # affichage des images de base
    clavier = pygame.image.load("clavier_images/deconnected.png").convert_alpha()
    screen.blit(clavier, KEYBOARD_DEST)

    image = pygame.image.load("souris_images/deconnected.png").convert_alpha()
    screen.blit(image, MOUSE_DEST)

    image_front = pygame.image.load("joystick_images/deconnected_front.png").convert_alpha()
    image_back = pygame.image.load("joystick_images/deconnected_back.png").convert_alpha()
    screen.blit(image_front, JOYSTICK_FRONT_DEST)
    # On la met juste à coté de l'image front
    screen.blit(image_back, JOYSTICK_BACK_DEST)

    pygame.display.update()

    # Quelques variables
    connected_devices = []
    clock = pygame.time.Clock()  # pour la gestion du temps dans la boucle principale
    running = True
    keyboard_connected = False
    mouse_connected = False
    joystick_connected = False
    path_1 = ""
    path_2 = ""

    # Boucle principale
    while running:

        event = pygame.event.poll()

        # On éfface l'écran
        screen.fill(BACKGROUND_COLOR)

        # Ecriture sur l'ecran
        # Gestion du clavier
        if keyboard_connected:
            path_1 = "clavier_images/connected.png"
        else:
            path_1 = "clavier_images/deconnected.png"

        clavier = pygame.image.load(path_1).convert_alpha()
        screen.blit(clavier, KEYBOARD_DEST)

        # Gestion de la souris
```

```
        if mouse_connected:
            path_1 = "souris_images/connected.png"
        else:
            path_1 = "souris_images/deconnected.png"
        image = pygame.image.load(path_1).convert_alpha()
        screen.blit(image, MOUSE_DEST)

        # Gestion du joystick
        if joystick_connected:
            # Initialisation du joystick
            # Le joystick à besion d'etre initialiser avec la sdl. On le met dans un try afin de
            # aurais aucun joystick connecté car ci c'est le cas l'initialisation renverait une
            try:
                joystick = pygame.joystick.Joystick(0)
                joystick.init()
                path_1 = "joystick_images/connected_front.png"
                path_2 = "joystick_images/connected_back.png"
            except:
                path_1 = "joystick_images/deconnected_front.png"
                path_2 = "joystick_images/deconnected_back.png"
        else:
            path_1 = "joystick_images/deconnected_front.png"
            path_2 = "joystick_images/deconnected_back.png"
        image_front = pygame.image.load(path_1).convert_alpha()
        image_back = pygame.image.load(path_2).convert_alpha()
        screen.blit(image_front, JOYSTICK_FRONT_DEST)
        screen.blit(image_back, JOYSTICK_BACK_DEST)

        # Recherche et affichage des informations sur les devices
        connected_devices = parse_devices()
        keyboard_connected,mouse_connected, joystick_connected = display_devices_informations(c

        # Gérer les événements pygame
        if event.type == pygame.KEYDOWN:
            on_key_press(event, screen)

        elif event.type == pygame.MOUSEBUTTONDOWN:
            on_click(event, screen)

        # Gestion des événements de joystick
        elif event.type == pygame.JOYAXISMOTION:
            # Mouvement de l'axe du joystick
            on_joystick_motion(event, screen)

        elif event.type == pygame.JOYBUTTONDOWN:
            # Appui sur un bouton du joystick
            on_joystick_pres(event, screen)

        elif event.type == pygame.QUIT:
           running = False

        # on met à jour l'écran
        pygame.display.update()

        clock.tick(FPS)

    pygame.quit()

if __name__ == "__main__":
    main()
```