Robot Vision

Découverte

Dans Thonny, installer `TensorFlow` `v2.12.1` `Opencv-python`

```python

import cv2
import numpy as np
import pygame
from tensorflow.keras.models import load_model
from random import randint

# 1. Charger le modèle et les labels
# Load the model
model = load_model("keras_Model.h5", compile=False)

# Load the labels
raw_class_names = open("labels.txt", "r").readlines()
class_names=[]
for name in raw_class_names:
    strip = name.strip()
    print (f"[{strip}]")
    class_names.append(strip)

# 2. Initialisation de la webcam OpenCV
camera = cv2.VideoCapture(0)
if not camera.isOpened():
    raise RuntimeError("Impossible d'ouvrir la webcam")

# 3. Initialisation de PyGame
pygame.init()
SCREEN_WIDTH, SCREEN_HEIGHT = 640, 480
screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption("Contrôle par IA (seuil de confiance > 90%)")
clock = pygame.time.Clock()

# 4. Définition du personnage
player_size  = 40
player_color = (255, 0, 0)  # rouge
player = pygame.Rect(
    (SCREEN_WIDTH - player_size) // 2,
    (SCREEN_HEIGHT - player_size) // 2,
    player_size,
    player_size
)
speed = 2  # pixels par frame

# Seuil minimal de confiance (90%)
CONFIDENCE_THRESHOLD = 0.90

```

```python
def get_prediction():
        # --- Capture et prétraitement de l'image
    ret, frame = camera.read()
    if not ret:
        return None

    # Resize the raw image into (224-height,224-width) pixels
    img = cv2.resize(frame, (224, 224), interpolation=cv2.INTER_AREA)

    # Make the image a numpy array and reshape it to the models input shape.
    x   = img.astype(np.float32).reshape(1, 224, 224, 3)

    # Normalize the image array
    x   = (x / 127.5) - 1

    # --- Prédiction
    preds             = model.predict(x)
    idx               = np.argmax(preds[0])
    direction         = class_names[idx]
    confidence_score  = preds[0][idx]

    return direction, confidence_score

# 5. Boucle principale
running = True
while running:
    # --- Gestion des événements PyGame
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    direction, confidence_score = get_prediction()

    # Affichage console (optionnel)
    pct = int(confidence_score * 100)

    # --- Déplacement du personnage seulement si la confiance est suffisante
    if confidence_score > CONFIDENCE_THRESHOLD:
        print(f"Direction [{direction}]****");
        if direction == "0 Haut":
            print("HAUT")
            player.y -= speed
        elif direction == "Bas":
            player.y += speed
        elif direction == "Gauche":
            player.x -= speed
        elif direction == "1 Droite":
            print("DROITE")
            player.x += speed
    else:
        # Confiance < 90% : on ne bouge pas (idle)
        pass

    # Empêcher le personnage de sortir de l'écran
    player.x = max(0, min(player.x, SCREEN_WIDTH - player_size))
    player.y = max(0, min(player.y, SCREEN_HEIGHT - player_size))
```

```
102
103     # --- Affichage PyGame
104     screen.fill((0, 0, 0))              # fond noir
105     pygame.draw.rect(screen, player_color, player)
106     pygame.display.flip()
107
108     # limiter à ~30 images par seconde
109     clock.tick(30)
110
111 # 6. Nettoyage
112 camera.release()
113 pygame.quit()
114
```