NUEXUS

Artificial Intelligence

Internship Program

INTERN WEEKLY TASKS
WEEK 3

2024

NUEXUS TECHNOLOGIES

Artifical Intelligence Internship Program

Task: 03

NUEXUS | WHERE TECHNOLOGY MEETS TOMORROW

```
# The code snippet is importing necessary libraries and modules for a Python program that involves
# hand tracking and a game. Here is a breakdown of each import statement:


import random
import cv2
import cvzone
from cvzone.HandTrackingModule import HandDetector
import time



# The code snippet `cap = cv2.VideoCapture(0)` followed by `cap.set(3, 640)` and `cap.set(4, 480)` is
# setting up a video capture object in OpenCV. Here's what each line does:


cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)



# The line `detector = HandDetector(maxHands=1)` is creating an instance of the `HandDetector` class
# from the `cvzone` module with a parameter `maxHands=1`. This parameter specifies that the detector
# should track a maximum of 1 hand in the video feed. The `HandDetector` class is used for detecting
# and tracking hands in the video frames captured by the camera.


detector = HandDetector(maxHands=1)



# The lines `timer = 0`, `stateResult = False`, `startGame = False`, and `scores = [0, 0]` are
# initializing variables for the game logic. Here's a breakdown of each variable:


timer = 0
stateResult = False
startGame = False
scores = [0, 0]  # [AI, Player]




# This `while True` loop in the code snippet is the main loop that runs continuously
# to handle the
# game logic and image processing for the Rock-Paper-Scissors game using hand gestures.


while True:
# The line `imgBG = cv2.imread(r"C:\Users\ZIA UL ISLAM MUGHAL\Desktop\task\Resources\BG.png")` is
# reading an image file named "BG.png" from the specified file path on the local system and storing it
# in the variable `imgBG`. This image will be used as the background image for the game interface or
# display.

    imgBG = cv2.imread(r"C:\Users\ZIA UL ISLAM MUGHAL\Desktop\task\Resources\BG.png")



# The line `success, img = cap.read()` is capturing a frame from the video feed obtained by the video
# capture object `cap` created using OpenCV. The `cap.read()` method reads a frame from the video
# capture object, and the returned values are assigned to `success` and `img`.


    success, img = cap.read()

# The code snippet `imgScaled = cv2.resize(img, (0, 0), None, 0.875, 0.875)` is resizing the captured
# image `img` using OpenCV's `resize` function. The parameters passed to the `resize` function are
# `(0, 0)` which means the output size is calculated based on the scaling factors provided next. The
# scaling factors `0.875, 0.875` indicate that the image will be resized to 87.5% of its original
# width and height.



    imgScaled = cv2.resize(img, (0, 0), None, 0.875, 0.875)
```

```
    imgScaled = imgScaled[:, 80:480]



    # Find Hands
# The line `hands, img = detector.findHands(imgScaled)  # with draw` is using the `HandDetector`
# instance `detector` to detect and track hands in the resized image `imgScaled`. The `findHands`
# method of the `HandDetector` class returns the detected hands and an image with drawn hand landmarks
# and bounding boxes.



    hands, img = detector.findHands(imgScaled)  # with draw

    if startGame:



 # This block of code is responsible for updating the timer display on the game interface. Here's a
 # breakdown of what it does:

        if stateResult is False:
            timer = time.time() - initialTime
            cv2.putText(imgBG, str(int(timer)), (605, 435), cv2.FONT_HERSHEY_PLAIN, 6, (255, 0, 255), 4)



           # When the condition `timer > 3` is met, it means that more than 3 seconds have passed
           # since the timer started. In this case, the following actions are taken:
            if timer > 3:
                stateResult = True
                timer = 0



                # This block of code is determining the player's move based on the hand gestures
                # detected by the HandDetector. Here's a breakdown of what it does:
                if hands:
                    playerMove = None
                    hand = hands[0]
                    fingers = detector.fingersUp(hand)
                    if fingers == [0, 0, 0, 0, 0]:
                        playerMove = 1
                    if fingers == [1, 1, 1, 1, 1]:
                        playerMove = 2
                    if fingers == [0, 1, 1, 0, 0]:
                        playerMove = 3


 # The code snippet you provided is part of a Rock-Paper-Scissors game implementation using hand
 # gestures. Let's break down the specific part you mentioned:



                    randomNumber = random.randint(1, 3)
                    imgAI = cv2.imread(fr"C:\Users\ZIA UL ISLAM MUGHAL\Desktop\task\Resources/{randomNumber}.png", cv2.IMREAD_UNCHANGED)
                    imgBG = cvzone.overlayPNG(imgBG, imgAI, (149, 310))



                   # This block of code is determining the outcome of the game round based on the
                   # player's move (`playerMove`) and the randomly generated move for the AI
                   # (`randomNumber`). Here's a breakdown of how the scoring works for a player win:
                    # Player Wins


                    if (playerMove == 1 and randomNumber == 3) or \
                            (playerMove == 2 and randomNumber == 1) or \
                            (playerMove == 3 and randomNumber == 2):
                        scores[1] += 1


                   # This block of code is determining the outcome of the game round where the AI wins
                   # based on the player's move (`playerMove`) and the randomly generated move for the
                   # AI (`randomNumber`).
                    # AI Wins



                    if (playerMove == 3 and randomNumber == 1) or \
```

```
                        (playerMove == 1 and randomNumber == 2) or \
                        (playerMove == 2 and randomNumber == 3):
                    scores[0] += 1



 # The line `imgBG[234:654, 795:1195] = imgScaled` is performing an image overlay operation in the
 # Python code snippet. Here's a breakdown of what this line is doing:
 imgBG[234:654, 795:1195] = imgScaled



# This part of the code snippet is responsible for updating the game interface with the current
# scores of the AI and the player. Here's a breakdown of what each line is doing:
 if stateResult:
     imgBG = cvzone.overlayPNG(imgBG, imgAI, (149, 310))

 cv2.putText(imgBG, str(scores[0]), (410, 215), cv2.FONT_HERSHEY_PLAIN, 4, (255, 255, 255), 6)
 cv2.putText(imgBG, str(scores[1]), (1112, 215), cv2.FONT_HERSHEY_PLAIN, 4, (255, 255, 255), 6)



 # cv2.imshow("Image", img)
 cv2.imshow("BG", imgBG)


 # cv2.imshow("Scaled", imgScaled)

# The code snippet `key = cv2.waitKey(1)` is capturing the key input from the user with a delay of
# 1 millisecond. It waits for a key event to occur and returns the ASCII value of the key that was
# pressed.



 key = cv2.waitKey(1)
 if key == ord('s'):
     startGame = True
     initialTime = time.time()
     stateResult = False
```