MediaPipe를 이용한 가위 바위 보 게임

소프트웨어학과 빅데이터전공 한서희 20215267

아이디어 주제

MediaPipe을 이용한 가위 바위 보 게임

아이디어 구체화

웹캠을 이용하여 사용자의 손동작을 실시간을 감지하여

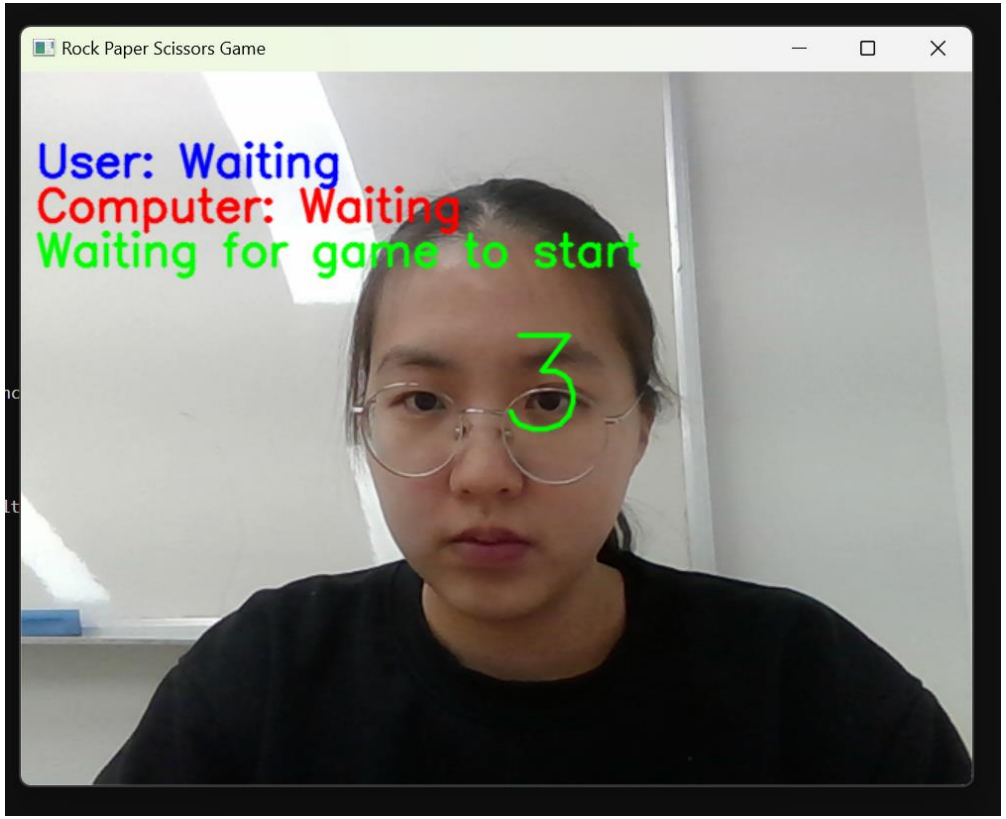
가위, 바위, 보 중 하나로 인식한다.

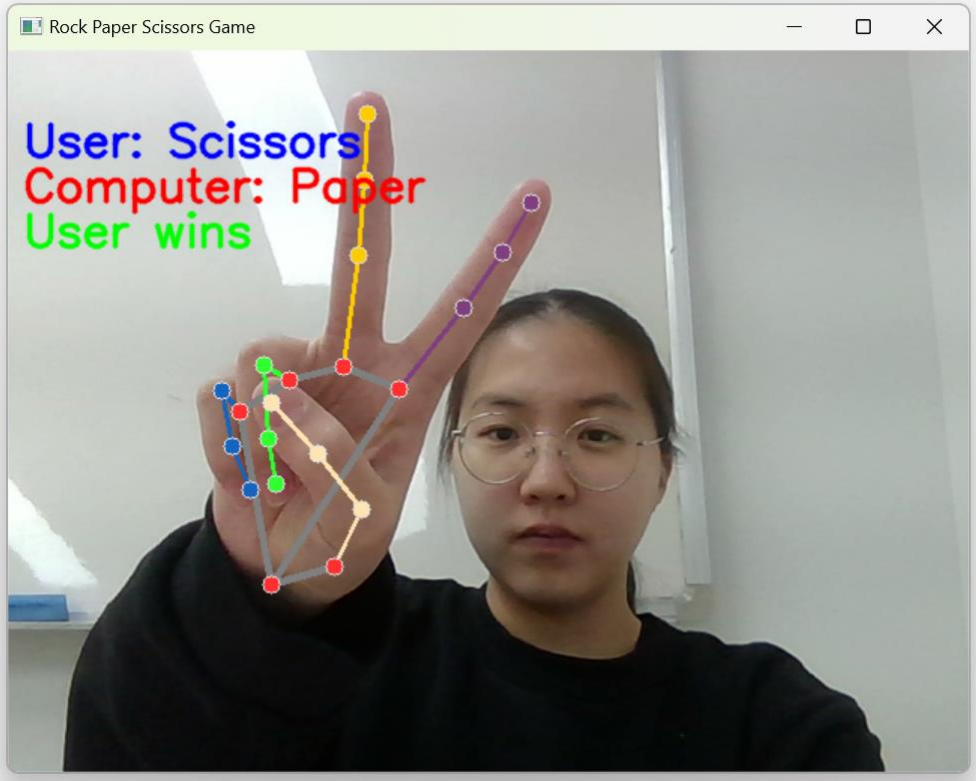
컴퓨터는 가위, 바위, 보 중 하나를 무작위로 선택하고

사용자의 손동작과 컴퓨터의 손동작을 비교하여 승패를 결정하여

승자를 정하는 게임이다

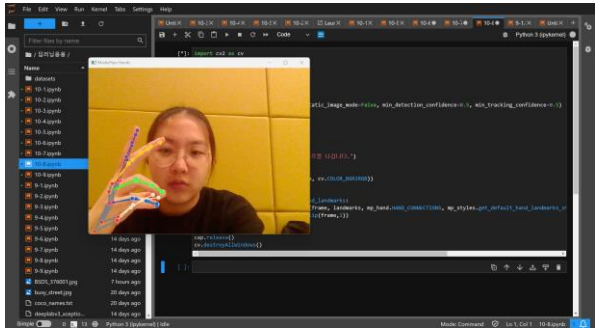
결과 미리보기


Rock Paper Scissors Game
User: Waiting
Computer: Waiting
Waiting for game to start
3


Rock Paper Scissors Game
User: Scissors
Computer: Paper
User wins

해당 주제를 선정한 이유 및 동기

실습 시간에 배웠던 프로그램 중에 10-8 [손 랜드마크 검출하기]가 가장 흥미로웠다



해당 프로그램을 응용하여 손의 랜드마크를 검출하는 것에서 더 나아가

어떤 동작을 하고 있는 지 인식하는 프로그램을 만들어보고 싶었다

따라서 가위, 보, 주먹을 인식하는 프로그램을 만들었고

이를 활용할 수 있는 방법을 생각하다가

가위 바위 게임 프로그램을 만들게 되었다

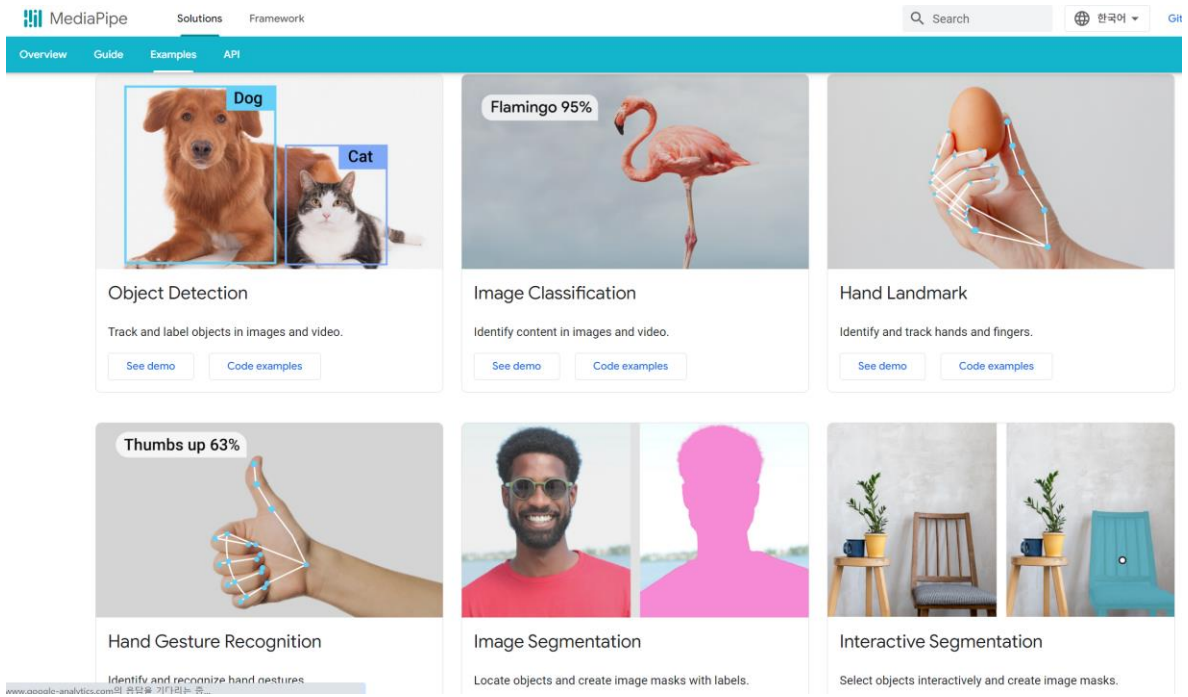
활용 인공지능 모델에 대한 이해 및 설명

MediaPipe

Mediapipe는 구글이 개발하여 제공하는 기계학습 개발 프레임워크로

여러가지 유용한 비디오 처리 솔루션을 제공한다

Mediapipe 홈페이지에 들어가면 여러가지 솔루션들을 볼 수 있다

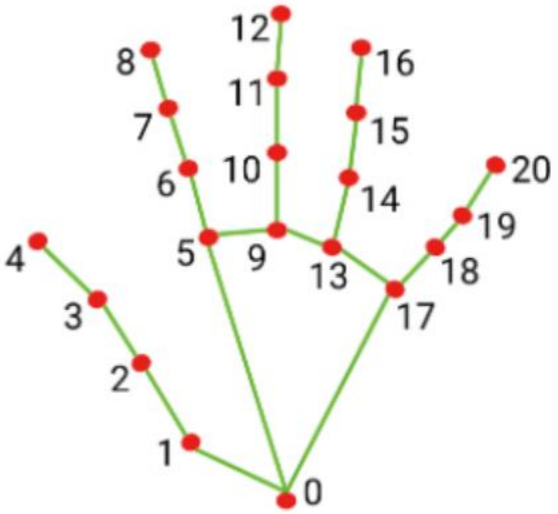


활용 인공지능 모델에 대한 이해 및 설명

MediaPipe

MediaPipe가 제공하는 솔루션을 활용하여 응용 프로그램을 쉽게 개발할 수 있다

해당 프로그램은 BlazeHand를 이용하였다



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

활용 인공지능 모델에 대한 이해 및 설명

MediaPipe

BlazeHand 솔루션은 손을 검출하는 BlazePalm 모듈과

랜드마크를 검출하고 추적하는 모듈로 구성된다.

손 랜드마크는 3차원 좌표로 표현한다.

처음에 손 동작 인식 프로그램을 만들려고 했을 때

어떻게 해야될지 막막하여

홈페이지에 있는 Hand Gesture Recognition 코드를

참고하여 만들어보려고 시도를 했었는데 실패한 후…

배운 프로그램을 응용하여 직접 코드를 짜봐야겠다고 마음먹었다

활용 인공지능 모델에 대한 이해 및 설명

MediaPipe

사실 처음부터 가위 바위 보를 만들어야겠다고 생각한 것은 아니였고

만들 수 있는 것들을 다 만들어봐야겠다고 생각했다

손을 펼친 모양이 제일 쉽지 않을까 해서 보자기부터 만들었고

사실 이것도 보자기가 아니라 처음엔 숫자 5를 인식하는 프로그램이였다

우선 보자기는 손을 활짝 핀 모양이다

프로그램을 구현하기 위해서는 랜드마크를 잘 이해해야됐다

손 랜드마크는 3차원 좌표로 표현한다는 점도 활용하여

각 손가락의 끝이 그 밑에 있는 관절보다 높은 곳에 위치해야된다

활용 인공지능 모델에 대한 이해 및 설명

MediaPipe

손 랜드마크의 위치는 x,y,z로 3차원 좌표로 표현하는데

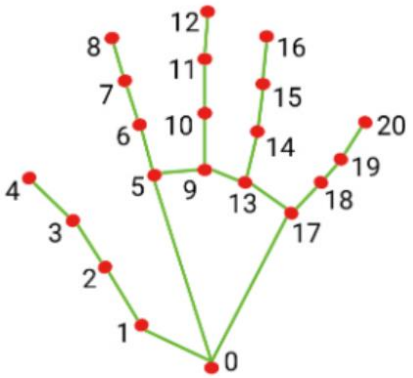
Y좌표는 랜드마크의 수직 위치를 나타낸다

이미지의 상단에서 하단으로 증가하기때문에

Y좌표가 작다는 것은 이미지의 상단에 가깝고

y좌표가 클 수록 이미지의 하단에 가깝다

처음에는 이것을 반대로 생각해서 한참 헤맸다



활용 인공지능 모델에 대한 이해 및 설명

MediaPipe

그래서 보자기 같은 경우에는

각 손가락 끝이 그 밑에 있는 관절보다 위에 있어야되니까

손가락 끝의 y좌표가 손가락 관절의 y좌표보다 작아야된다

따라서

thumb_tip.y < thumb_mcp.y and index_finger_tip.y < index_finger_mcp.y and middle_finger_tip.y

< middle_finger_mcp.y and ring_finger_tip.y < ring_finger_mcp.y and pinky_tip.y < pinky_mcp.y

이런식으로 조건을 붙여주면 된다

활용 인공지능 모델에 대한 이해 및 설명

MediaPipe

그리고 주먹은 사실

엄지를 올리고 있는 손 동작을 인식하는 프로그램을 만들려고 했는데

주먹 인식이 너무 잘됐다

왜 그럴까 생각해보니 주먹을 쥐어보면

엄지의 끝이 항상 다른 손가락의 끝보다 위에 있다는 것을 깨달았다

thumb_tip.y < index_finger_tip.y and thumb_tip.y < middle_finger_tip.y and thumb_tip.y

< ring_finger_tip.y and thumb_tip.y < pinky_tip.y

활용 인공지능 모델에 대한 이해 및 설명

MediaPipe

그리고 가위는 원래 v를 그리고 있는 손동작이였다

두번째 손가락과 가운데 손가락만 활짝 핀 모양이다

그래서 두번째 손가락과 가운데 손가락의 끝이 다른 손가락의 끝보다 위에 있는다

index_finger_tip.y < thumb_tip.y and index_finger_tip.y < ring_finger_tip.y and index_finger_tip.y

< pinky_tip.y and middle_finger_tip.y < thumb_tip.y and middle_finger_tip.y < ring_finger_tip.y and

middle_finger_tip.y < pinky_tip.y

제안 아이디어 방법 설명

게임 방식: 사용자는 가위, 바위, 보 중 하나를 선택한다

선택 후, 사용자는 준비 시간인 5초 동안 손동작을 취해야한다

게임 진행: 준비 시간이 끝나면, 화면에 나타난 사용자의 손동작을 시스템이 인식합니다.

동시에, 가위, 바위, 보 중 하나를 무작위로 선택하여 컴퓨터의 손동작으로 반영합니다.

사용자의 손동작과 컴퓨터의 손동작을 비교하여 승자를 결정합니다.

게임 재시작 조건: 플레이어와 컴퓨터의 손동작이 비겼을 경우

웹캠이 사용자의 손동작을 가위, 바위, 보 중 어느 것으로도 인식하지 못했을 경우

웹캠이 사용자의 손을 전혀 인지하지 못했을 경우 (손이 화면에 나타나지 않을 때)

0. 프로그램 10-8

1. 보자기 인식 프로그램

2. 바위 인식 프로그램

3. 가위 인식 프로그램

4. 가위 바위 보 인식 프로그램

5. 사용자의 손동작 프로그램

6. 컴퓨터의 손동작 추가

7. 승패 결정 기능 추가

8. 시간 제한 추가 / r을 누르면 게임을 다시 시작하는 기능 추가

9. 손 감지를 아예 못 할 경우 게임을 다시 시작하는 기능 추가

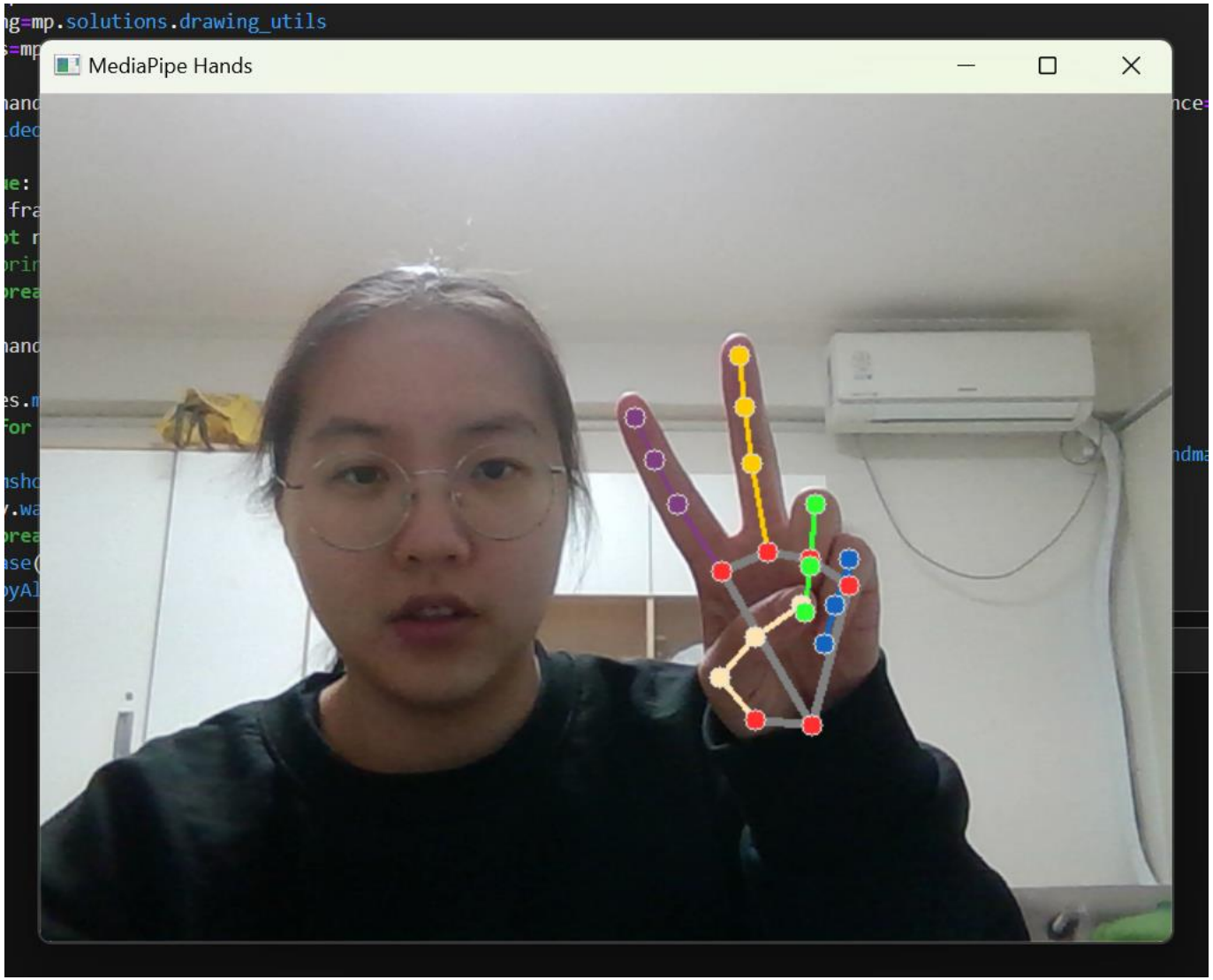
구현

0. 프로그램 10-8

```python
import cv2 as cv
import mediapipe as mp

mp_hand=mp.solutions.hands
mp_drawing=mp.solutions.drawing_utils
mp_styles=mp.solutions.drawing_styles

hand=mp_hand.Hands(max_num_hands=2, static_image_mode=False, min_detection_confidence=0.5, min_tracking_confidence=0.5)
cap=cv.VideoCapture(0,cv.CAP_DSHOW)

while True:
    ret, frame=cap.read()
    if not ret:
        print("프레임 획득에 실패하여 루프를 나갑니다.")
        break

    res=hand.process(cv.cvtColor(frame, cv.COLOR_BGR2RGB))

    if res.multi_hand_landmarks:
        for landmarks in res.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, landmarks, mp_hand.HAND_CONNECTIONS, mp_styles.get_default_hand_landmarks_style(), mp_styles.get_default_hand_connections_style())
    cv.imshow("MediaPipe Hands", cv.flip(frame,1))
    if cv.waitKey(5)==ord('q'):
        break
cap.release()
cv.destroyAllWindows()
```

중간 결과

0. 프로그램 10-8



구현

1. 보자기 인식 프로그램

```
[1]: #보자기
import cv2 as cv
import mediapipe as mp

#추가한 코드
def paper(hand_landmarks):
    landmarks = hand_landmarks.landmark
    thumb_tip = landmarks[mp_hand.HandLandmark.THUMB_TIP]
    index_finger_tip = landmarks[mp_hand.HandLandmark.INDEX_FINGER_TIP]
    middle_finger_tip = landmarks[mp_hand.HandLandmark.MIDDLE_FINGER_TIP]
    ring_finger_tip = landmarks[mp_hand.HandLandmark.RING_FINGER_TIP]
    pinky_tip = landmarks[mp_hand.HandLandmark.PINKY_TIP]

    thumb_mcp = landmarks[mp_hand.HandLandmark.THUMB_MCP]
    index_finger_mcp = landmarks[mp_hand.HandLandmark.INDEX_FINGER_MCP]
    middle_finger_mcp = landmarks[mp_hand.HandLandmark.MIDDLE_FINGER_MCP]
    ring_finger_mcp = landmarks[mp_hand.HandLandmark.RING_FINGER_MCP]
    pinky_mcp = landmarks[mp_hand.HandLandmark.PINKY_MCP]

    return thumb_tip.y < thumb_mcp.y and index_finger_tip.y < index_finger_mcp.y and middle_finger_tip.y < middle_finger_mcp.y and

mp_hand = mp.solutions.hands
mp_styles = mp.solutions.drawing_styles

hand = mp_hand.Hands(max_num_hands=2, static_image_mode=False, min_detection_confidence=0.5, min_tracking_confidence=0.5)
cap = cv.VideoCapture(0, cv.CAP_DSHOW)

while True:
    ret, frame = cap.read()
    if not ret:
        print("Failed to capture frame. Exiting.")
        break

    frame_rgb = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
    res = hand.process(frame_rgb)

    if res.multi_hand_landmarks:
        for landmarks in res.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, landmarks, mp_hand.HAND_CONNECTIONS,
                                      mp_styles.get_default_hand_landmarks_style(),
                                      mp_styles.get_default_hand_connections_style())

            #추가한 코드
            if paper(landmarks):
                cv.putText(frame, 'Paper', (10, 50), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv.LINE_AA)

    cv.imshow("MediaPipe Hands", frame)

    if cv.waitKey(5) == ord('q'):
        break

cap.release()
```

기존 프로그램 10-8 코드에서
이 부분만 추가를 하여 보자기 인식 프로그램을
만들었다

구현

1. 보자기 인식 프로그램

```
[1]: #보자기
import cv2 as cv
import mediapipe as mp

#추가한 코드
def paper(hand_landmarks):
    landmarks = hand_landmarks.landmark
    thumb_tip = landmarks[mp_hand.HandLandmark.THUMB_TIP]
    index_finger_tip = landmarks[mp_hand.HandLandmark.INDEX_FINGER_TIP]
    middle_finger_tip = landmarks[mp_hand.HandLandmark.MIDDLE_FINGER_TIP]
    ring_finger_tip = landmarks[mp_hand.HandLandmark.RING_FINGER_TIP]
    pinky_tip = landmarks[mp_hand.HandLandmark.PINKY_TIP]

    thumb_mcp = landmarks[mp_hand.HandLandmark.THUMB_MCP]
    index_finger_mcp = landmarks[mp_hand.HandLandmark.INDEX_FINGER_MCP]
    middle_finger_mcp = landmarks[mp_hand.HandLandmark.MIDDLE_FINGER_MCP]
    ring_finger_mcp = landmarks[mp_hand.HandLandmark.RING_FINGER_MCP]
    pinky_mcp = landmarks[mp_hand.HandLandmark.PINKY_MCP]

    return thumb_tip.y < thumb_mcp.y and index_finger_tip.y < index_finger_mcp.y and middle_finger_tip.y < middle_finger_mcp.y and

mp_hand = mp.solutions.hands
mp_styles = mp.solutions.drawing_styles

hand = mp_hand.Hands(max_num_hands=2, static_image_mode=False, min_detection_confidence=0.5, min_tracking_confidence=0.5)
cap = cv.VideoCapture(0, cv.CAP_DSHOW)

while True:
    ret, frame = cap.read()
    if not ret:
        print("Failed to capture frame. Exiting.")
        break

    frame_rgb = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
    res = hand.process(frame_rgb)

    if res.multi_hand_landmarks:
        for landmarks in res.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, landmarks, mp_hand.HAND_CONNECTIONS,
                                      mp_styles.get_default_hand_landmarks_style(),
                                      mp_styles.get_default_hand_connections_style())

            #추가한 코드
            if paper(landmarks):
                cv.putText(frame, 'Paper', (10, 50), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv.LINE_AA)

    cv.imshow("MediaPipe Hands", frame)

    if cv.waitKey(5) == ord('q'):
        break

cap.release()
```

보자기: 손을 활짝 핀 모양

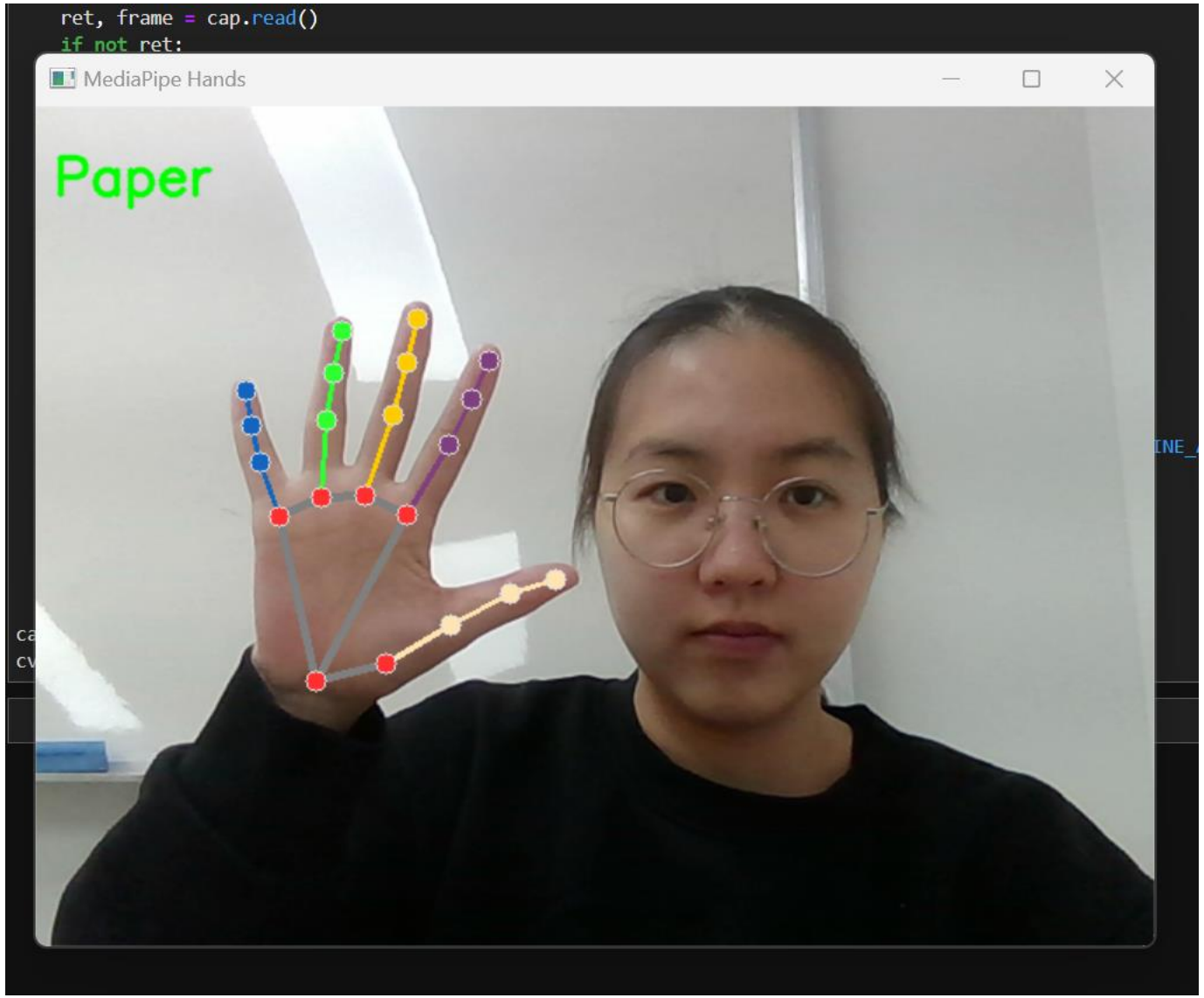
보자기 함수 return 값
각 손가락의 끝이 그 밑에 있는 관절보다 위에 있어야된다. 따라서 각 손가락 끝의 y좌표가 해당 손가락 관절의 y좌표보다 작아야된다

따라서 손의 랜드마크 (tip과 mcp만 했다)를 불러와서 각 변수에 저장을 해주고
보자기에 알맞는 조건을 return해준다

화면에 글자를 표시하고 싶었는데
프로그램 10-8은 거울모드로 되어있어서
글자 또한 뒤집어졌다. 그래서 flip을 지워버렸다

중간 결과

1. 보자기 인식 프로그램



구현

2. 바위 인식 프로그램

```
[2]: #바위
import cv2 as cv
import mediapipe as mp

#변경한 코드
def rock(hand_landmarks):
    landmarks = hand_landmarks.landmark
    thumb_tip = landmarks[mp_hand.HandLandmark.THUMB_TIP]
    index_finger_tip = landmarks[mp_hand.HandLandmark.INDEX_FINGER_TIP]
    middle_finger_tip = landmarks[mp_hand.HandLandmark.MIDDLE_FINGER_TIP]
    ring_finger_tip = landmarks[mp_hand.HandLandmark.RING_FINGER_TIP]
    pinky_tip = landmarks[mp_hand.HandLandmark.PINKY_TIP]

    thumb_mcp = landmarks[mp_hand.HandLandmark.THUMB_MCP]
    index_finger_mcp = landmarks[mp_hand.HandLandmark.INDEX_FINGER_MCP]
    middle_finger_mcp = landmarks[mp_hand.HandLandmark.MIDDLE_FINGER_MCP]
    ring_finger_mcp = landmarks[mp_hand.HandLandmark.RING_FINGER_MCP]
    pinky_mcp = landmarks[mp_hand.HandLandmark.PINKY_MCP]

    return thumb_tip.y < index_finger_tip.y and thumb_tip.y < middle_finger_tip.y and thumb_tip.y < ring_finger_tip.y and thumb_tip.y < pinky_tip.y

mp_hand = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_styles = mp.solutions.drawing_styles

hand = mp_hand.Hands(max_num_hands=2, static_image_mode=False, min_detection_confidence=0.5, min_tracking_confidence=0.5)
cap = cv.VideoCapture(0, cv.CAP_DSHOW)

while True:
    ret, frame = cap.read()
    if not ret:
        print("Failed to capture frame. Exiting.")
        break

    frame_rgb = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
    res = hand.process(frame_rgb)

    if res.multi_hand_landmarks:
        for landmarks in res.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, landmarks, mp_hand.HAND_CONNECTIONS,
                                      mp_styles.get_default_hand_landmarks_style(),
                                      mp_styles.get_default_hand_connections_style())

            #변경한 코드
            if rock(landmarks):
                cv.putText(frame, 'Rock', (10, 50), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv.LINE_AA)

    cv.imshow("MediaPipe Hands", frame)

    if cv.waitKey(5) == ord('q'):
        break

cap.release()
```

바위: 손을 오므린 모양

바위 함수 return 값

엄지의 끝이 항상

다른 손가락의 끝보다 위에 있다

변수 저장 부분

보자기와 동일하게 했다

바위는 finger_tip 랜드마크만 있으면

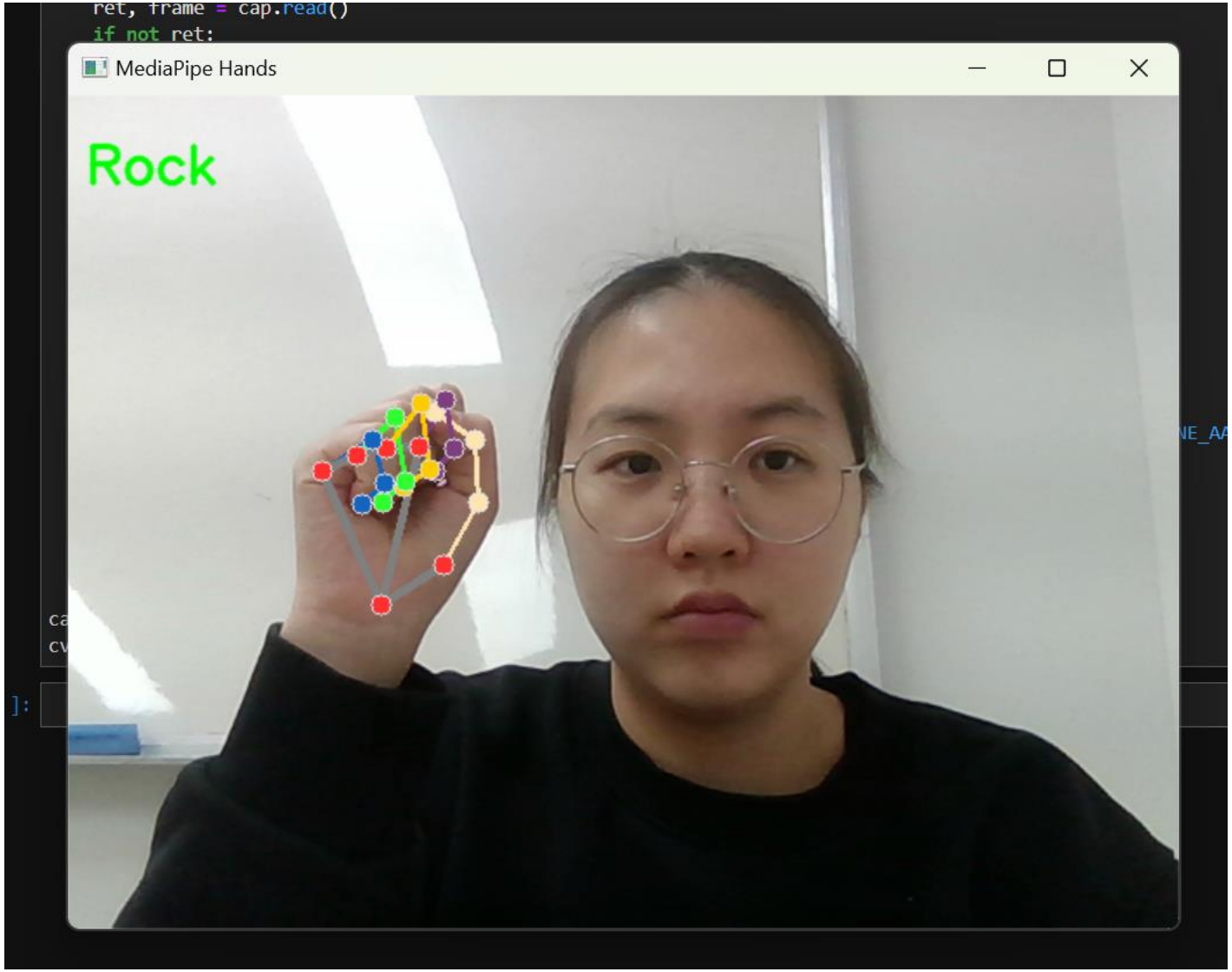
되긴 하지만 어차피 나중에

가위 바위 보 프로그램을 하나로 합칠 것이니

크게 신경쓰지 않아도 될 것 같다

중간 결과

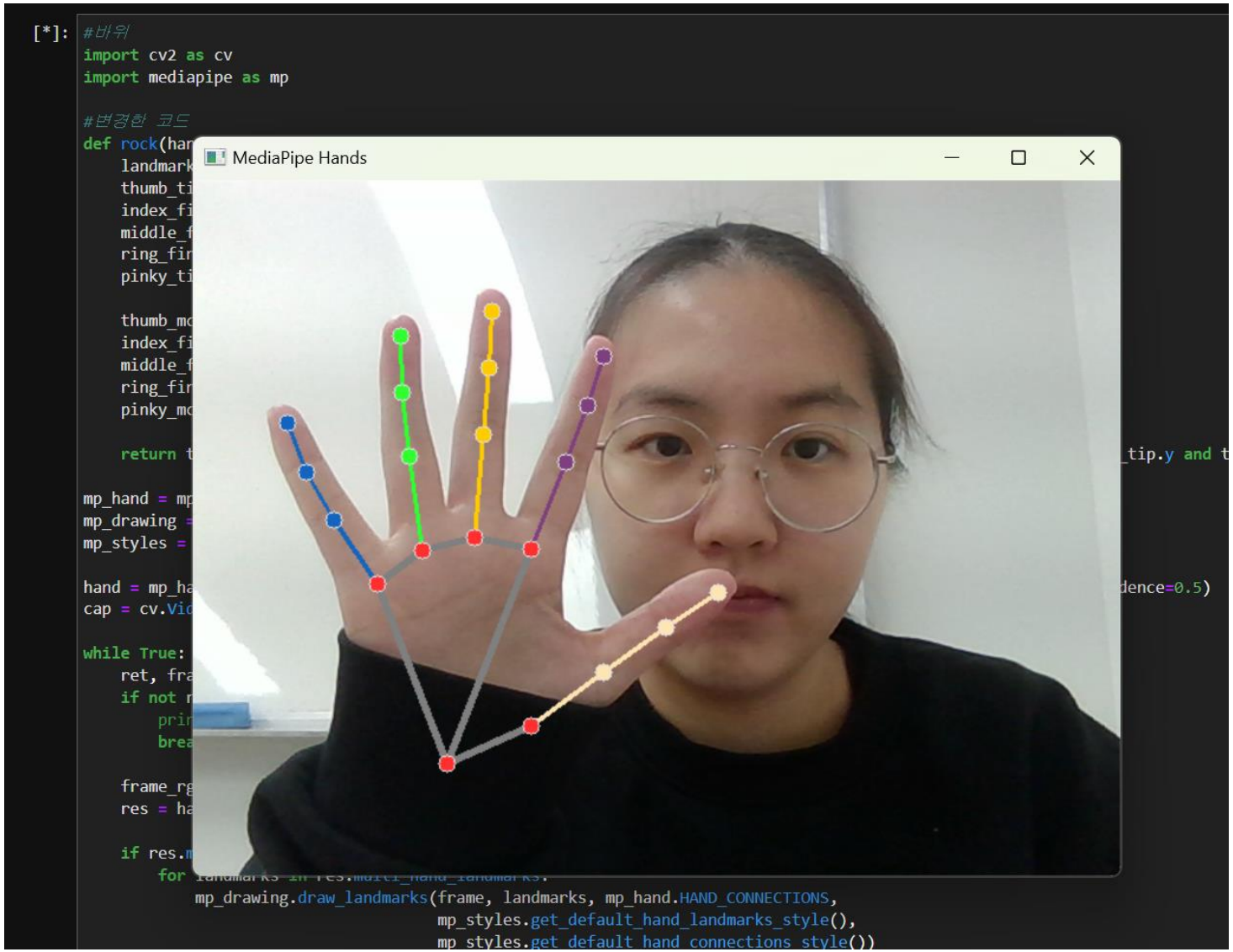
2. 바위 인식 프로그램



중간 결과

2. 바위 인식 프로그램

보자기일때는
Rock 문구 안뜸



구현

3. 가위 인식 프로그램

```
[2]: #가위
import cv2 as cv
import mediapipe as mp

#변경한 코드
def scissors(hand_landmarks):
    landmarks = hand_landmarks.landmark
    thumb_tip = landmarks[mp_hand.HandLandmark.THUMB_TIP]
    index_finger_tip = landmarks[mp_hand.HandLandmark.INDEX_FINGER_TIP]
    middle_finger_tip = landmarks[mp_hand.HandLandmark.MIDDLE_FINGER_TIP]
    ring_finger_tip = landmarks[mp_hand.HandLandmark.RING_FINGER_TIP]
    pinky_tip = landmarks[mp_hand.HandLandmark.PINKY_TIP]

    thumb_mcp = landmarks[mp_hand.HandLandmark.THUMB_MCP]
    index_finger_mcp = landmarks[mp_hand.HandLandmark.INDEX_FINGER_MCP]
    middle_finger_mcp = landmarks[mp_hand.HandLandmark.MIDDLE_FINGER_MCP]
    ring_finger_mcp = landmarks[mp_hand.HandLandmark.RING_FINGER_MCP]
    pinky_mcp = landmarks[mp_hand.HandLandmark.PINKY_MCP]

    return index_finger_tip.y < thumb_tip.y and index_finger_tip.y < ring_finger_tip.y and index_finger_tip.y < pinky_tip.y

mp_hand = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_styles = mp.solutions.drawing_styles

hand = mp_hand.Hands(max_num_hands=2, static_image_mode=False, min_detection_confidence=0.5, min_tracking_confidence=0.5)
cap = cv.VideoCapture(0, cv.CAP_DSHOW)

while True:
    ret, frame = cap.read()
    if not ret:
        print("Failed to capture frame. Exiting.")
        break

    frame_rgb = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
    res = hand.process(frame_rgb)

    if res.multi_hand_landmarks:
        for landmarks in res.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, landmarks, mp_hand.HAND_CONNECTIONS,
                                      mp_styles.get_default_hand_landmarks_style(),
                                      mp_styles.get_default_hand_connections_style())

            #변경한 코드
            if sissors(landmarks):
                cv.putText(frame, 'Scissors', (10, 50), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv.LINE_AA)

    cv.imshow("MediaPipe Hands", frame)

    if cv.waitKey(5) == ord('q'):
        break

cap.release()
```

가위: 두번째 손가락과 가운데 손가락만 핀 모양

가위 함수 return 값

두번째 손가락과 가운데 손가락의 끝이 다른

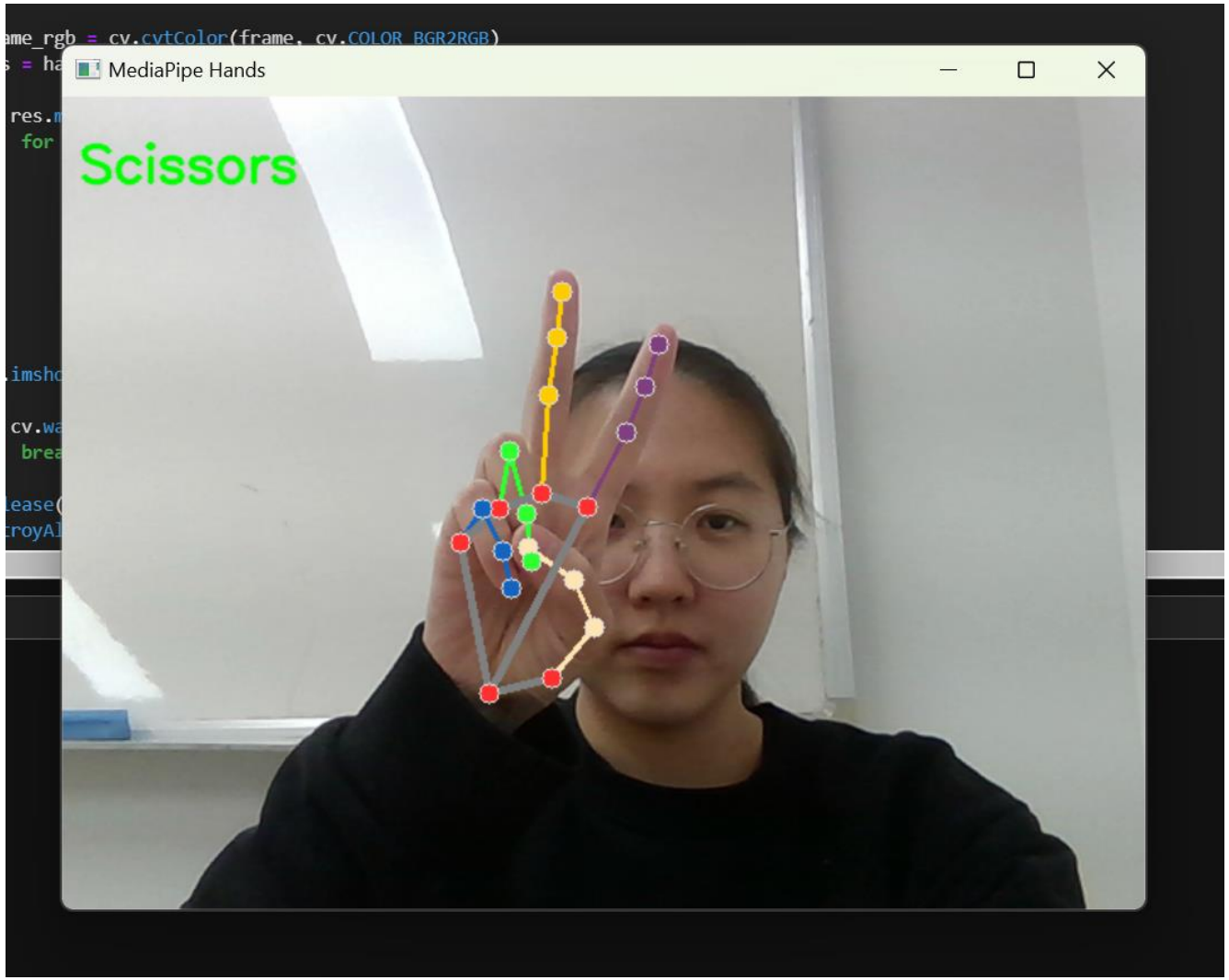
손가락의 끝보다 위에 있다

변수 저장 부분

동일

중간 결과

3. 가위 인식 프로그램



구현

4. 가위 바위 보 인식 프로그램

```python
#가위
def scissors(hand_landmarks):
    landmarks = hand_landmarks.landmark
    thumb_tip = landmarks[mp_hand.HandLandmark.THUMB_TIP]
    index_finger_tip = landmarks[mp_hand.HandLandmark.INDEX_FINGER_TIP]
    middle_finger_tip = landmarks[mp_hand.HandLandmark.MIDDLE_FINGER_TIP]
    ring_finger_tip = landmarks[mp_hand.HandLandmark.RING_FINGER_TIP]
    pinky_tip = landmarks[mp_hand.HandLandmark.PINKY_TIP]

    thumb_mcp = landmarks[mp_hand.HandLandmark.THUMB_MCP]
    index_finger_mcp = landmarks[mp_hand.HandLandmark.INDEX_FINGER_MCP]
    middle_finger_mcp = landmarks[mp_hand.HandLandmark.MIDDLE_FINGER_MCP]
    ring_finger_mcp = landmarks[mp_hand.HandLandmark.RING_FINGER_MCP]
    pinky_mcp = landmarks[mp_hand.HandLandmark.PINKY_MCP]

    return index_finger_tip.y < thumb_tip.y and index_finger_tip.y < ring_finger_tip.y and index_finger_tip.y < pinky_tip.y and middle_finger_tip.y < t


mp_hand = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_styles = mp.solutions.drawing_styles

hand = mp_hand.Hands(max_num_hands=2, static_image_mode=False, min_detection_confidence=0.5, min_tracking_confidence=0.5)
cap = cv.VideoCapture(0, cv.CAP_DSHOW)

while True:
    ret, frame = cap.read()
    if not ret:
        print("Failed to capture frame. Exiting.")
        break

    frame_rgb = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
    res = hand.process(frame_rgb)

    if res.multi_hand_landmarks:
        for landmarks in res.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, landmarks, mp_hand.HAND_CONNECTIONS,
                                      mp_styles.get_default_hand_landmarks_style(),
                                      mp_styles.get_default_hand_connections_style())

            if paper(landmarks):
                cv.putText(frame, 'Paper', (10, 50), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv.LINE_AA)

            if rock(landmarks):
                cv.putText(frame, 'Rock', (10, 100), cv.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv.LINE_AA)

            if sissor(landmarks):
                cv.putText(frame, 'Scissors', (10, 150), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv.LINE_AA)


    cv.imshow("MediaPipe Hands", frame)
    if cv.waitKey(5) == ord('q'):
        break
```

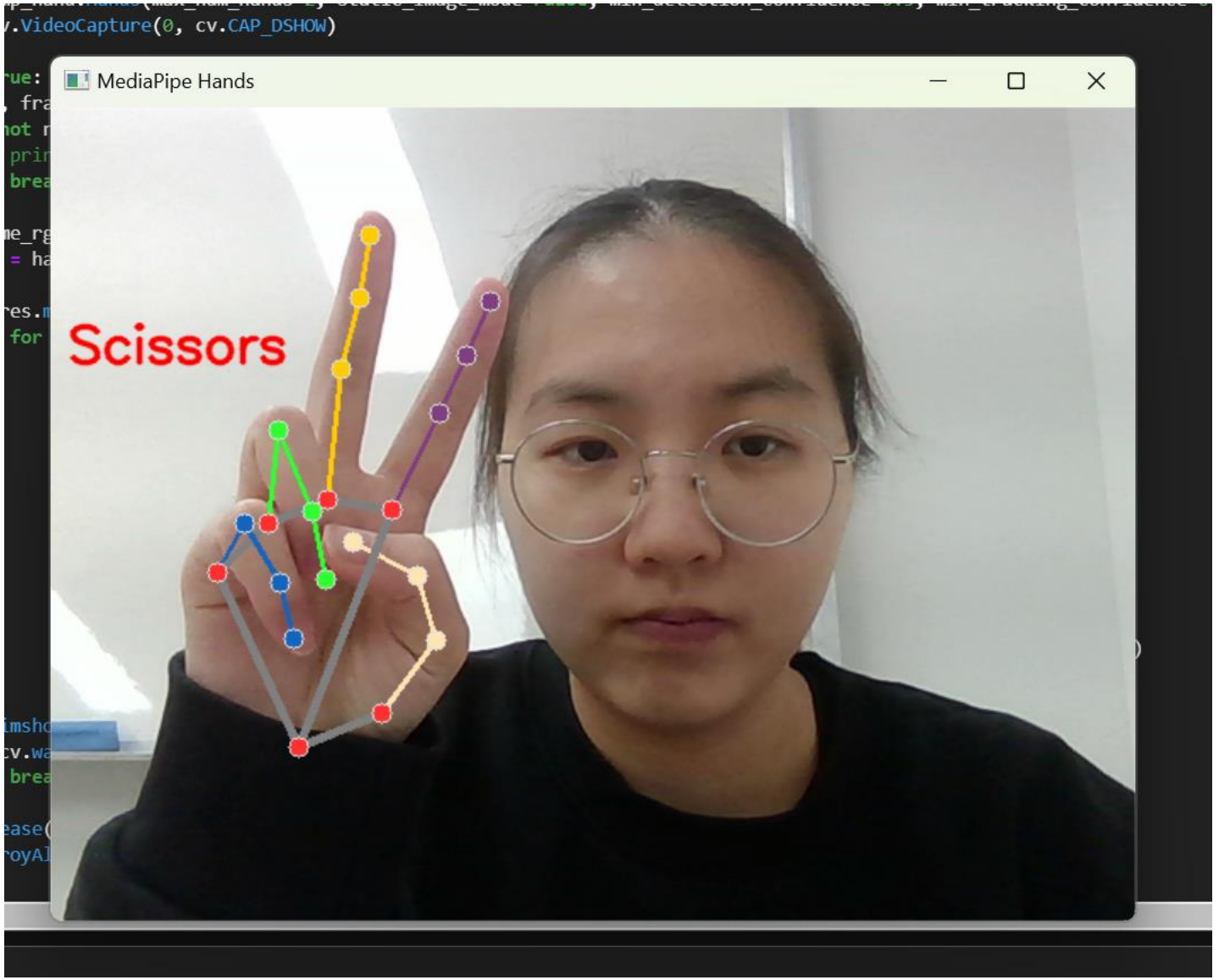
단순하게

가위 바위 보 함수를 다 적어주고

If 문을 세 개 추가해줬다

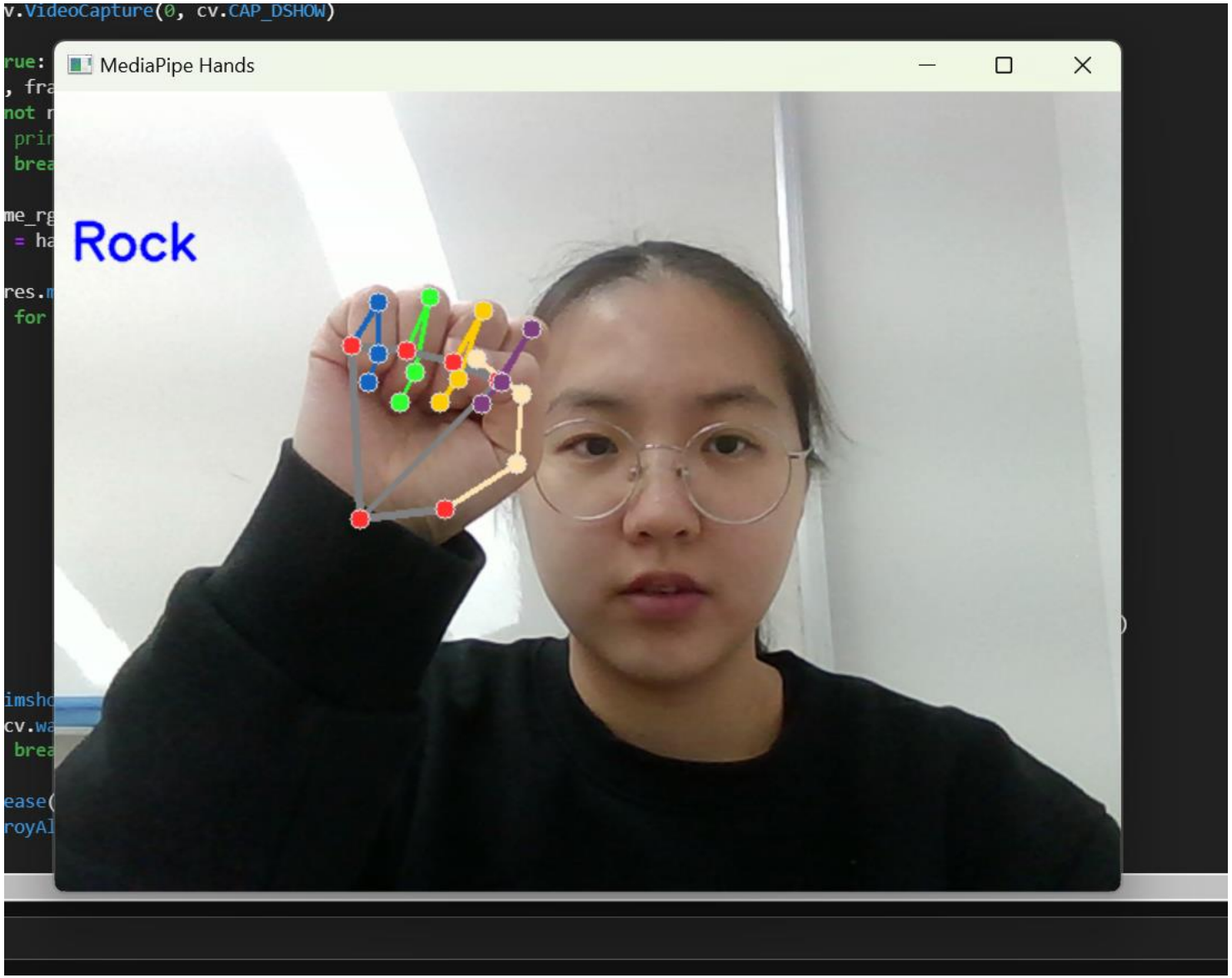
중간 결과

4. 가위 바위 보 인식 프로그램



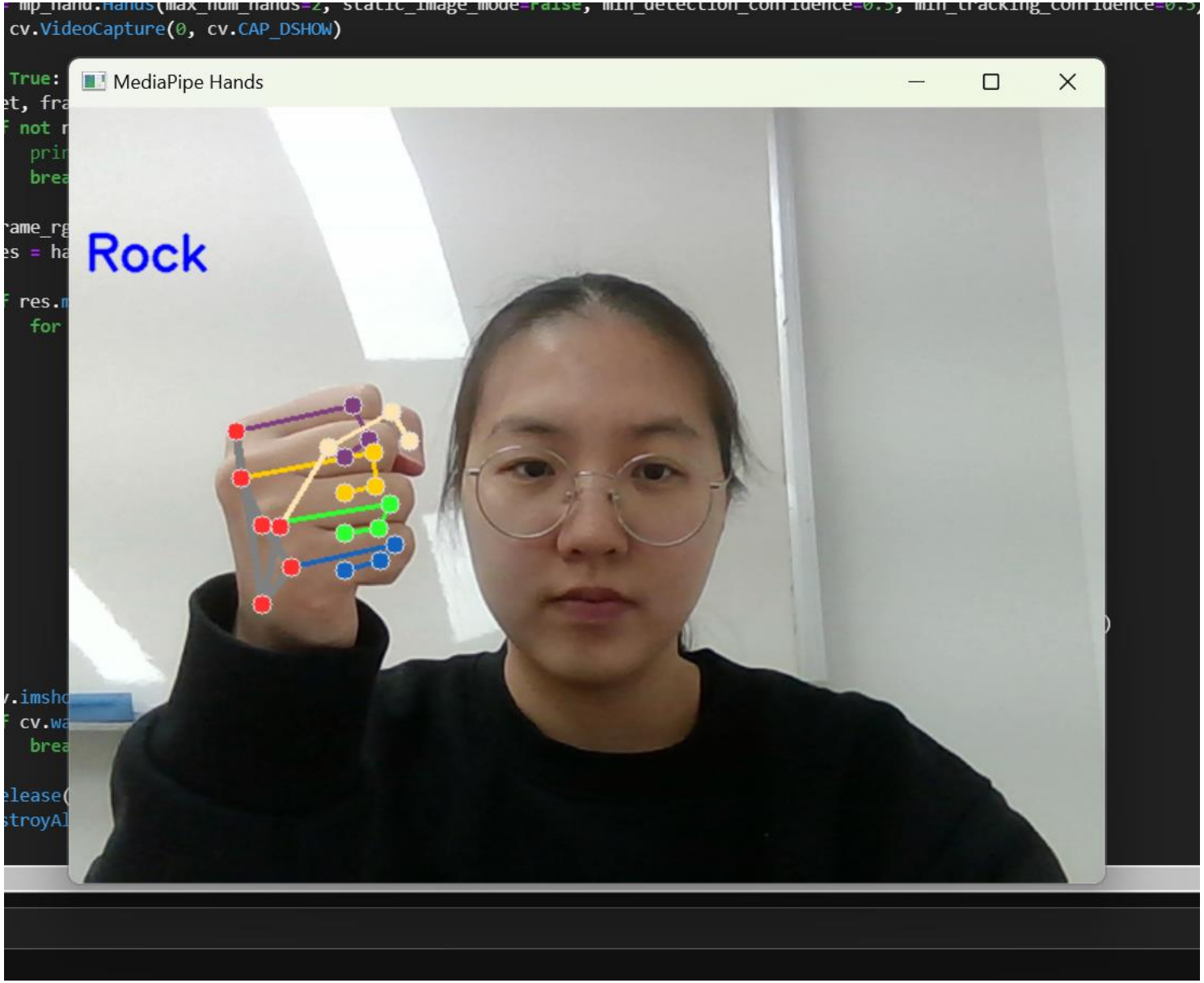
중간 결과

4. 가위 바위 보 인식 프로그램



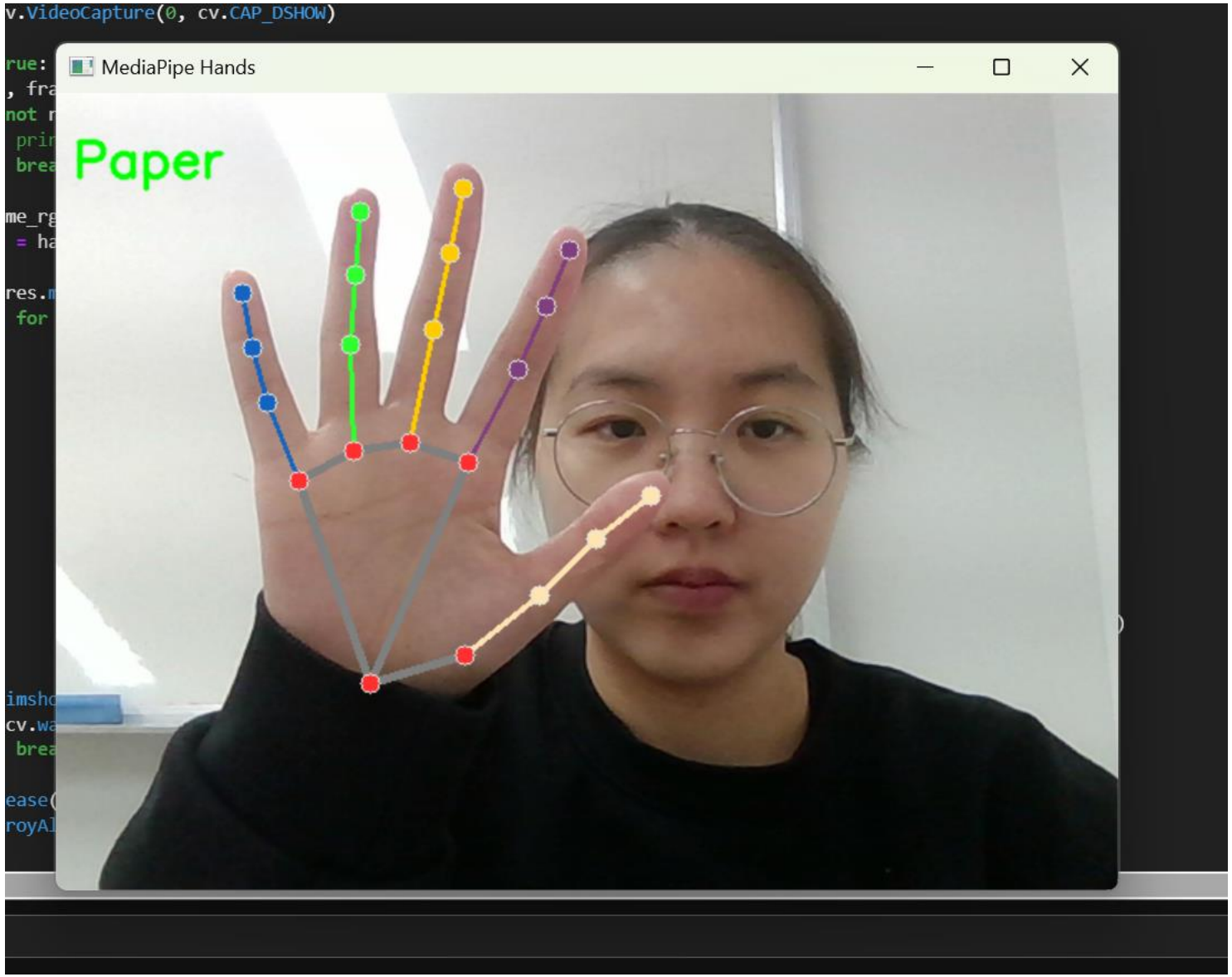
중간 결과

4. 가위 바위 보 인식 프로그램



중간 결과

4. 가위 바위 보 인식 프로그램



구현

5. 사용자 손 동작 인식 프로그램

```
[2]: import cv2 as cv
     import mediapipe as mp

     #변경한 부분 (함수 하나로 합치기)
     #User
     def user_choice(hand_landmarks):
         landmarks = hand_landmarks.landmark
         thumb_tip = landmarks[mp_hand.HandLandmark.THUMB_TIP]
         index_finger_tip = landmarks[mp_hand.HandLandmark.INDEX_FINGER_TIP]
         middle_finger_tip = landmarks[mp_hand.HandLandmark.MIDDLE_FINGER_TIP]
         ring_finger_tip = landmarks[mp_hand.HandLandmark.RING_FINGER_TIP]
         pinky_tip = landmarks[mp_hand.HandLandmark.PINKY_TIP]

         thumb_mcp = landmarks[mp_hand.HandLandmark.THUMB_MCP]
         index_finger_mcp = landmarks[mp_hand.HandLandmark.INDEX_FINGER_MCP]
         middle_finger_mcp = landmarks[mp_hand.HandLandmark.MIDDLE_FINGER_MCP]
         ring_finger_mcp = landmarks[mp_hand.HandLandmark.RING_FINGER_MCP]
         pinky_mcp = landmarks[mp_hand.HandLandmark.PINKY_MCP]

         #보
         if thumb_tip.y < thumb_mcp.y and index_finger_tip.y < index_finger_mcp.y and middle_finger_tip.y < middle_finger_mcp.y and ring_finger_tip.y < ring
             return "Paper"
         #바위
         elif thumb_tip.y < index_finger_tip.y and thumb_tip.y < middle_finger_tip.y and thumb_tip.y < ring_finger_tip.y and thumb_tip.y < pinky_tip.y:
             return "Rock"
         #가위
         elif index_finger_tip.y < thumb_tip.y and index_finger_tip.y < ring_finger_tip.y and index_finger_tip.y < pinky_tip.y and middle_finger_tip.y < thu
             return "Scissors"

         else:
             return "null"


     mp_drawing = mp.solutions.drawing_utils
     mp_styles = mp.solutions.drawing_styles

     hand = mp_hand.Hands(max_num_hands=2, static_image_mode=False, min_detection_confidence=0.5, min_tracking_confidence=0.5)
     cap = cv.VideoCapture(0, cv.CAP_DSHOW)

     while True:
```

가위 바위 보 함수를

하나의 함수로 합쳤다

구현

5. 사용자 손 동작 인식 프로그램

```
hand = mp_hand.Hands(max_num_hands=2, static_image_mode=False, min_detection_confidence=0.5, min_tracking_confidence=0.5)
cap = cv.VideoCapture(0, cv.CAP_DSHOW)

while True:
    ret, frame = cap.read()
    if not ret:
        print("Failed to capture frame. Exiting.")
        break

    frame_rgb = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
    res = hand.process(frame_rgb)

    if res.multi_hand_landmarks:
        for landmarks in res.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, landmarks, mp_hand.HAND_CONNECTIONS,
                                      mp_styles.get_default_hand_landmarks_style(),
                                      mp_styles.get_default_hand_connections_style())

            #변경한 부분
            user_hand_shape = user_choice(landmarks)
            cv.putText(frame, f"User: {user_hand_shape}", (10, 50), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv.LINE_AA)


    cv.imshow("MediaPipe Hands", frame)
    if cv.waitKey(5) == ord('q'):
        break

cap.release()
cv.destroyAllWindows()
```

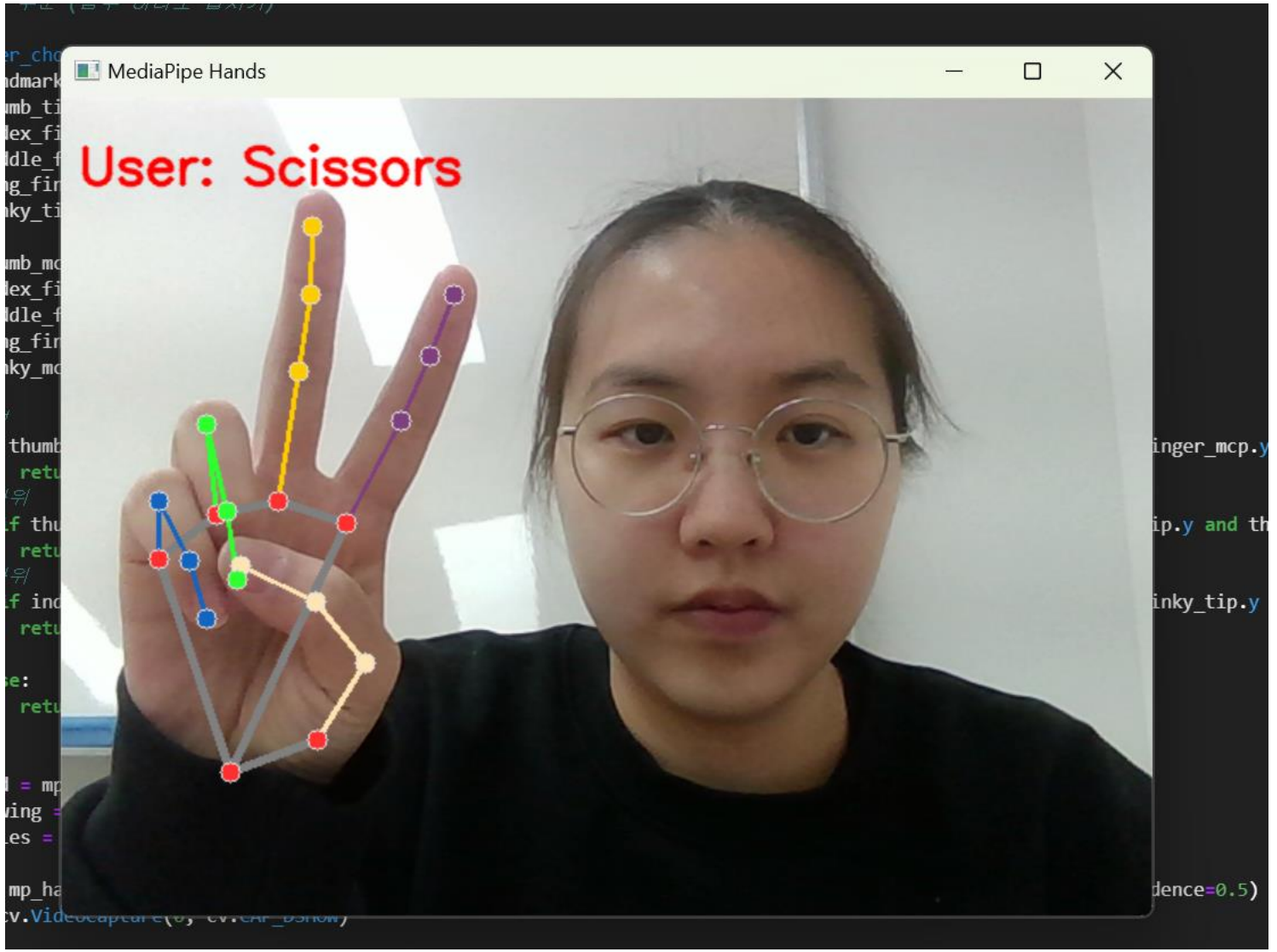
User_hand_shape에

User_choice 함수의 리턴값을

저장하여 사용한다

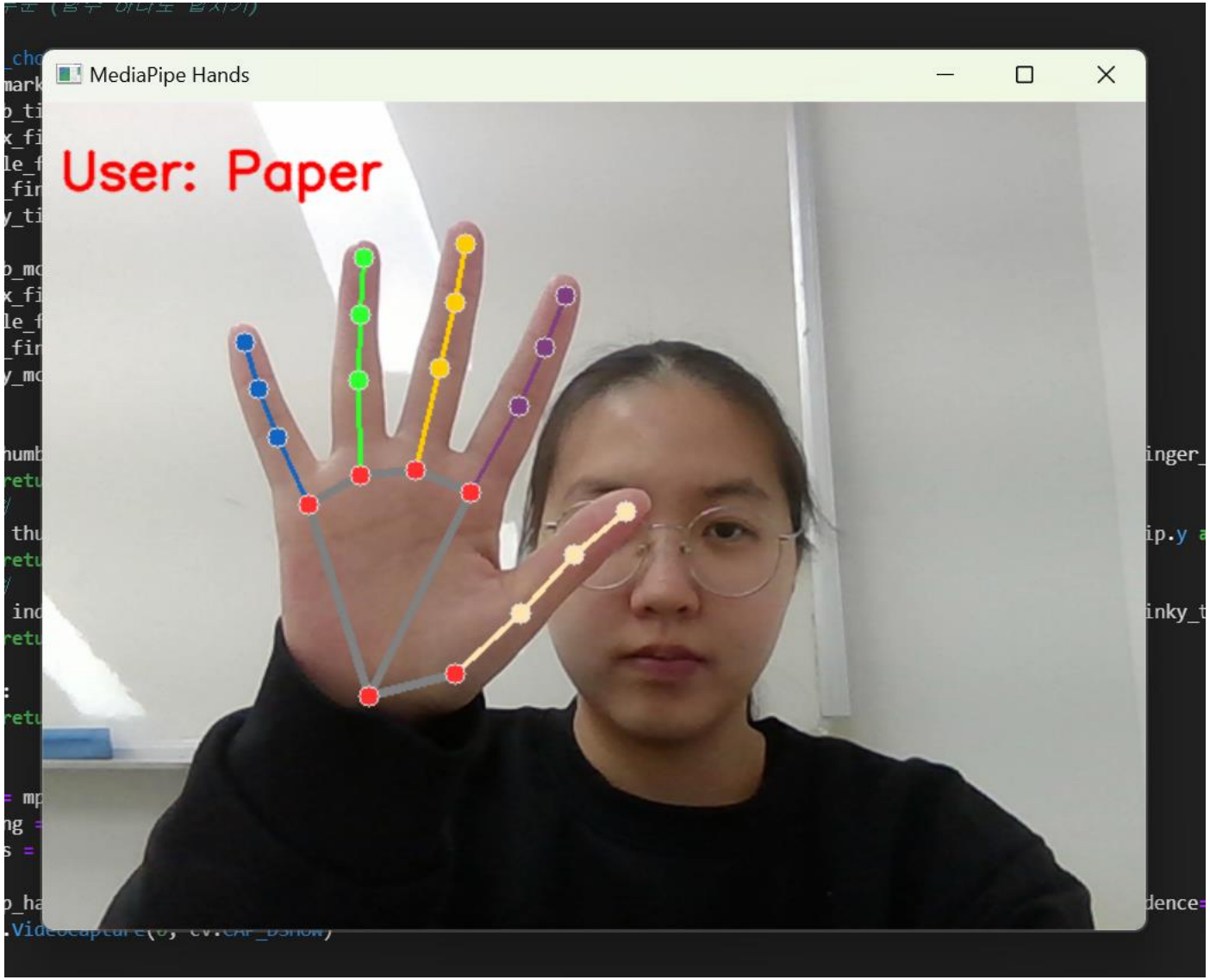
중간 결과

5. 사용자 손 동작 인식 프로그램



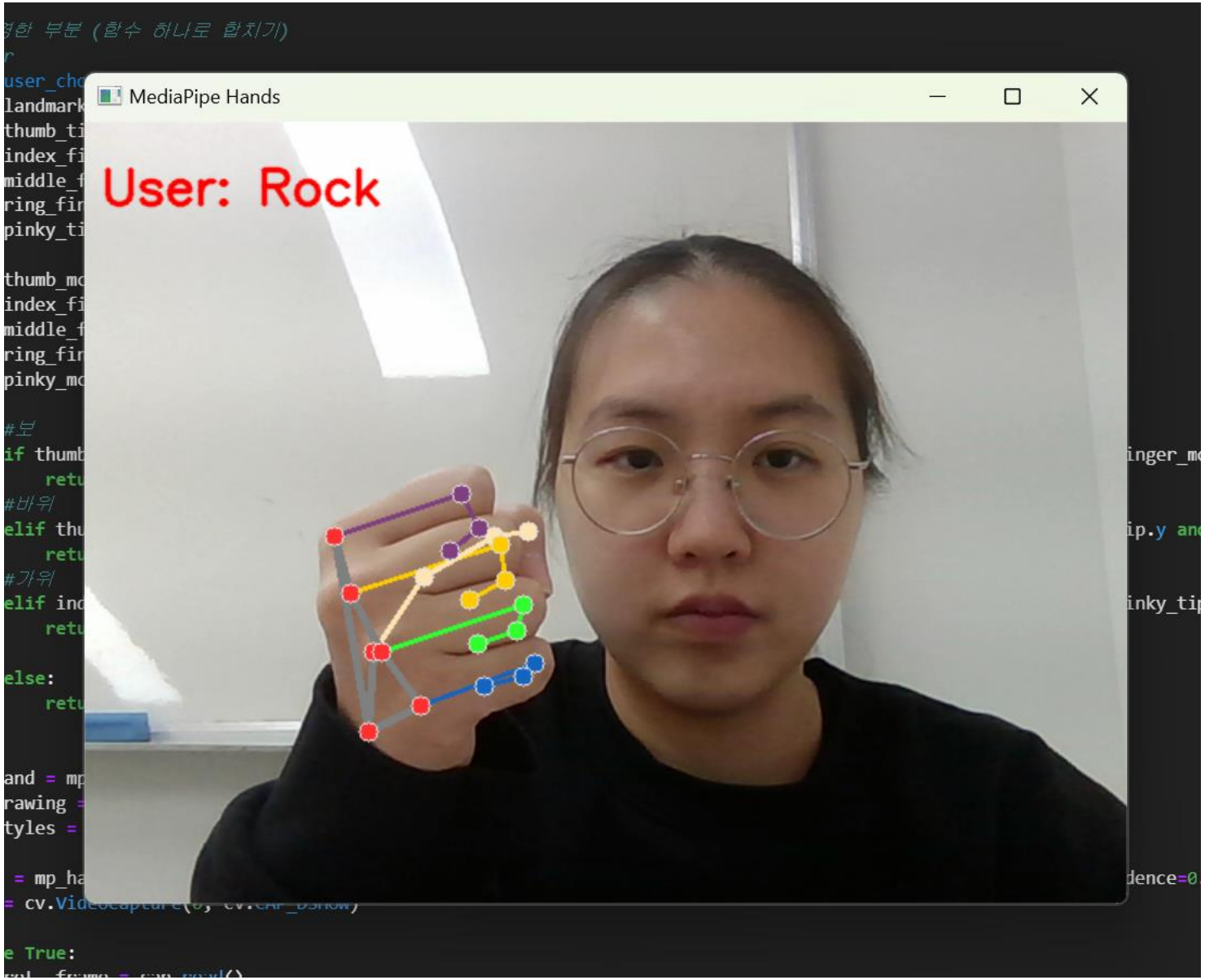
중간 결과

5. 사용자 손 동작 인식 프로그램



중간 결과

5. 사용자 손 동작 인식 프로그램



구현

6. 컴퓨터 손 동작 추가

```python
#Computer
def computer_choice():
    return random.choice(["Rock", "Paper", "Scissors"])

mp_hand = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_styles = mp.solutions.drawing_styles

hand = mp_hand.Hands(max_num_hands=2, static_image_mode=False, min_detection_confidence=0.5, min_tracking_confidence=0.5)
cap = cv.VideoCapture(0, cv.CAP_DSHOW)

while True:
    ret, frame = cap.read()
    if not ret:
        print("Failed to capture frame. Exiting.")
        break

    frame_rgb = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
    res = hand.process(frame_rgb)

    if res.multi_hand_landmarks:
        for landmarks in res.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, landmarks, mp_hand.HAND_CONNECTIONS,
                                      mp_styles.get_default_hand_landmarks_style(),
                                      mp_styles.get_default_hand_connections_style())


            user_hand_shape = user_choice(landmarks)
            cv.putText(frame, f"User: {user_hand_shape}", (10, 50), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv.LINE_AA)

            #추가한 부분
            computer_hand_shape = computer_choice()
            cv.putText(frame, f"Computer: {computer_hand_shape}", (10, 100), cv.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv.LINE_AA)
```

가위 바위 보 중에

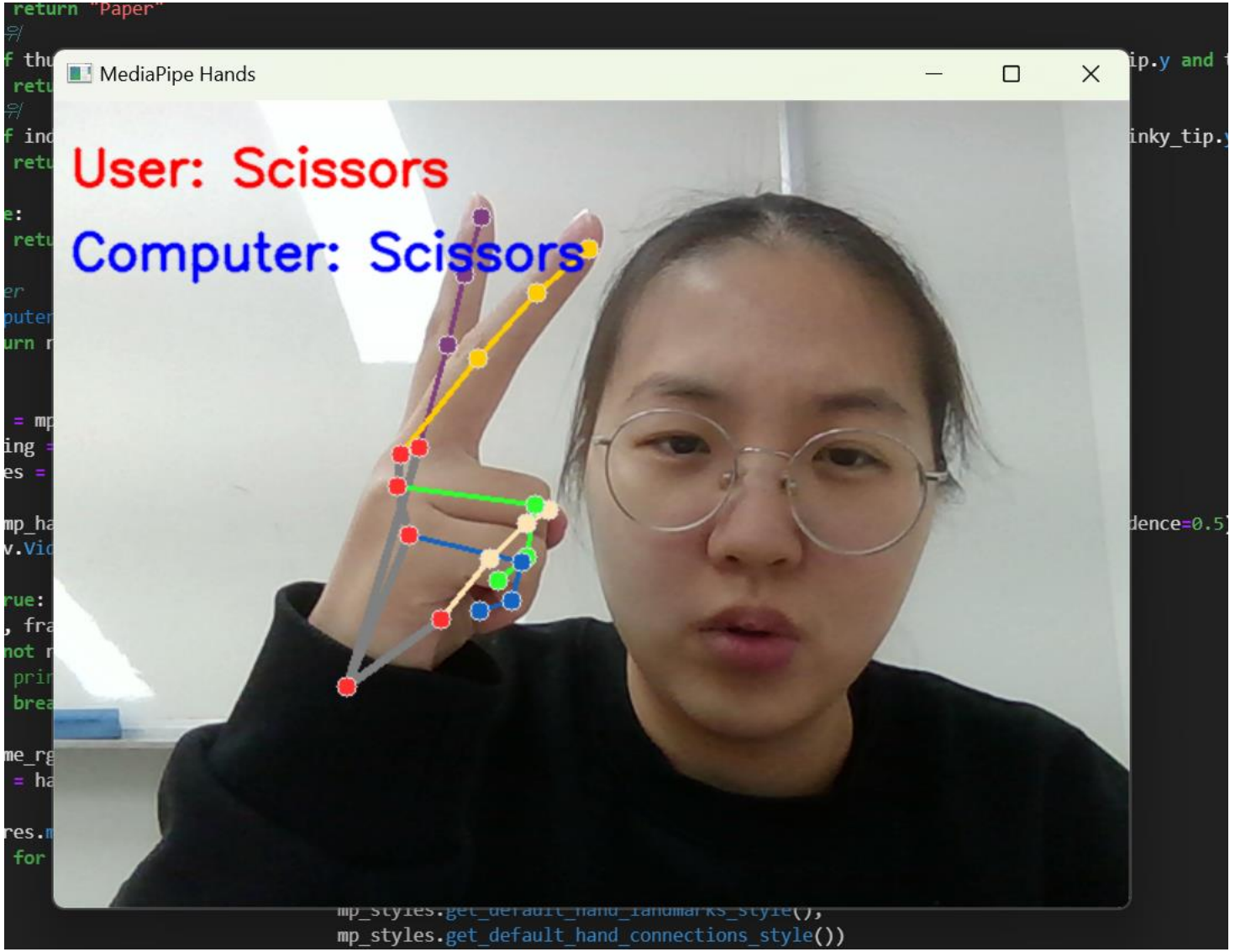
무작위로 하나 선택하는 함수

그리고 computer_hand_shape에

Return 값 저장해준다

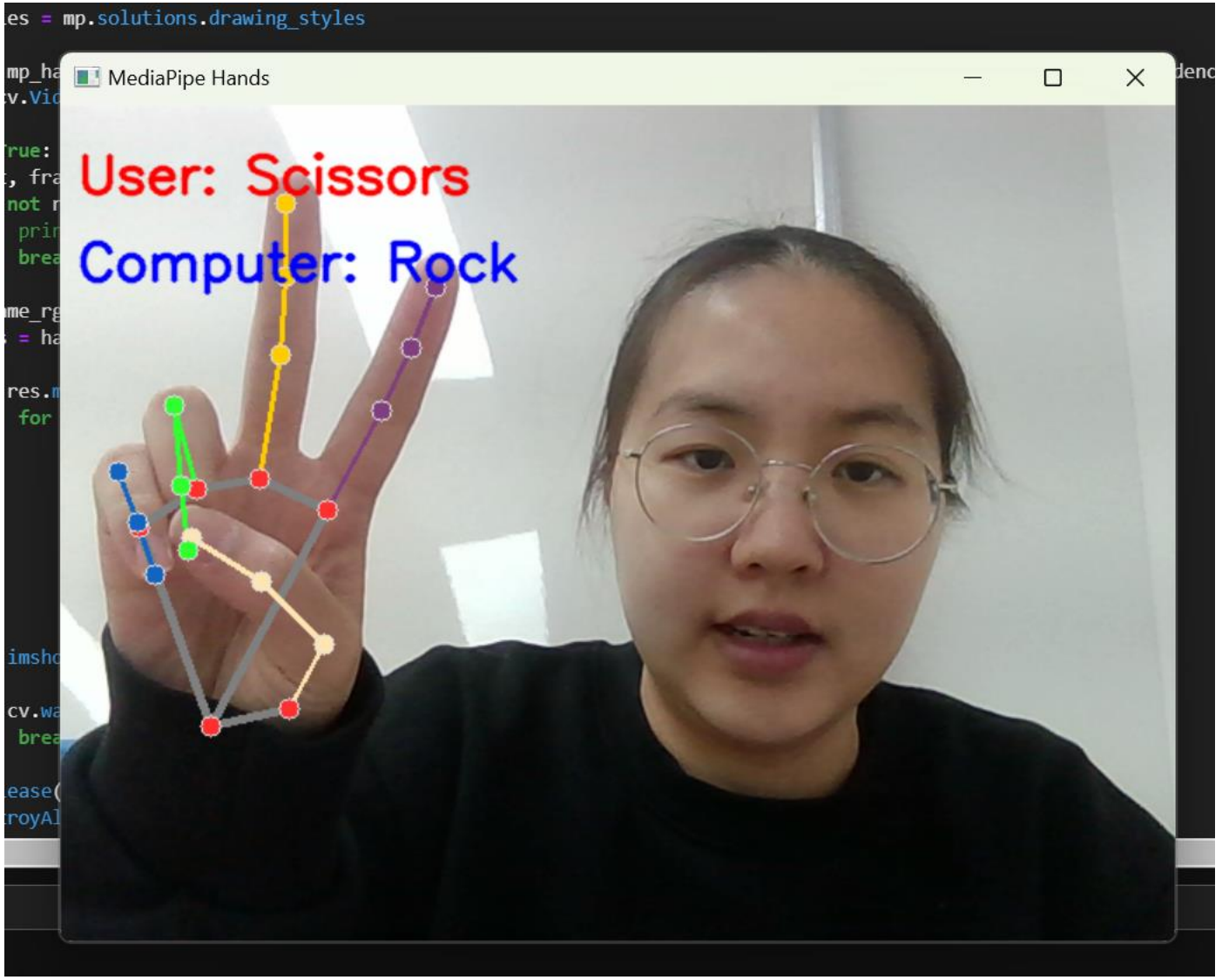
중간 결과

6. 컴퓨터 손 동작 추가



중간 결과

6. 컴퓨터 손 동작 추가



구현

7. 승패 결정 추가

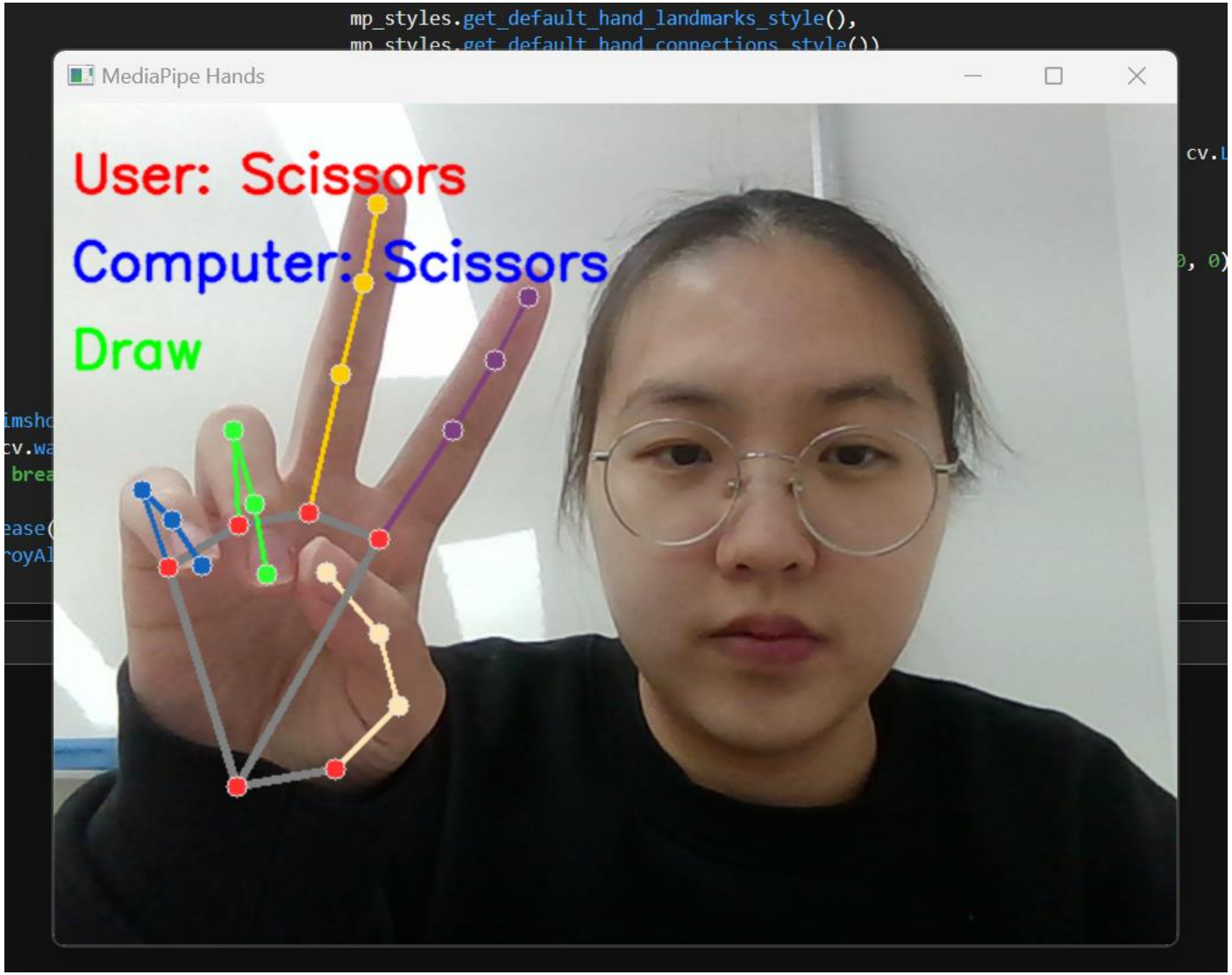
```python
#추가한 부분
#winner
def winner(user_choice, computer_choice):
    if user_choice=="null":
        return "Do it again"

    elif user_choice == computer_choice:
        return "Draw"

    elif (user_choice == "Rock" and computer_choice == "Scissors") or \
         (user_choice == "Scissors" and computer_choice == "Paper") or \
         (user_choice == "Paper" and computer_choice == "Rock"):
        return "User wins"
    else:
        return "Computer wins"
```

```python
mp_hand = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_styles = mp.solutions.drawing_styles

hand = mp_hand.Hands(max_num_hands=2, static_image_mode=False, min_detection_confidence=0.5, min_tracking_confidence=0.5)
cap = cv.VideoCapture(0, cv.CAP_DSHOW)

while True:
    ret, frame = cap.read()
    if not ret:
        print("Failed to capture frame. Exiting.")
        break

    frame_rgb = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
    res = hand.process(frame_rgb)

    if res.multi_hand_landmarks:
        for landmarks in res.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, landmarks, mp_hand.HAND_CONNECTIONS,
                                      mp_styles.get_default_hand_landmarks_style(),
                                      mp_styles.get_default_hand_connections_style())


            user_hand_shape = user_choice(landmarks)
            cv.putText(frame, f"User: {user_hand_shape}", (10, 50), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv.LINE_AA)

            computer_hand_shape = computer_choice()
            cv.putText(frame, f"Computer: {computer_hand_shape}", (10, 100), cv.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv.LINE_AA)

            result =winner(user_hand_shape, computer_hand_shape)
            cv.putText(frame, result, (10, 150), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv.LINE_AA)
```

중간 결과

7. 승패 결정 추가

이 화면은
캡쳐하였기때문에
이렇게 나오는 것이지
영상으로는 매순간
계속 컴퓨터의
손동작과 승패 결과가
바뀌게 된다



구현

8. 시간 추가 / r을 누르면 게임을 다시 시작하는 기능 추가

승패가 실시간으로 결정되어

매순간 계속 게임이 진행되어 승패를 확인하기도 어렵고

게임이 정상적으로 진행되지 않았다

이 문제점을 어떻게 해결하면 좋을 까 고민을 하다가

현실에서 '가위 바위 보'를 외쳐서 준비할 수 있는 시간을 주고

'보'를 외칠 때의 손 동작을 가지고 승패를 결정한다는 것을 떠올렸다

그래서 게임에 5초의 준비시간을 추가하였고

5초 뒤의 화면을 가지고 손동작을 분석하여 승패를 결정하는 프로그램으로 수정하였다.

또한 게임을 더욱 편리하게 만들기 위하여

키보드에서 R을 누르면 게임을 다시할 수 있도록 하였다

구현

8. 시간 추가 / r을 누르면 게임을 다시 시작하는 기능 추가

```python
#게임 시작
def reset_game():
    global game_start_time, game_started, user_hand_shape, computer_hand_shape, result
    game_start_time = time.time() + 5  #5초로 설정
    game_started = False
    user_hand_shape = "Waiting"
    computer_hand_shape = "Waiting"
    result = "Waiting for game to start"

reset_game()
```

```python
while True:
    ret, frame = cap.read()
    if not ret:
        print("Failed to capture frame. Exiting.")
        break

    frame_rgb = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
    res = hand.process(frame_rgb)
```

```python
    # 시간 카운트다운 표시
    time_left = int(game_start_time - time.time())
    if time_left > 0:
        cv.putText(frame, str(time_left), (frame.shape[1] // 2, frame.shape[0] // 2), cv.FONT_HERSHEY_SIMPLEX, 3, (0, 255, 0), 2, cv.LINE_AA)
    else:
        if not game_started:
            game_started = True

            #손을 감지했을 경우
            if res.multi_hand_landmarks:
                for landmarks in res.multi_hand_landmarks:

                    user_hand_shape = user_choice(landmarks)

                    #제대로 감지 못하면 다시 시작
                    if user_hand_shape == "null":
                        reset_game()
                        break

                    computer_hand_shape = computer_choice()
                    result = winner(user_hand_shape, computer_hand_shape)

                    #비기면 다시 시작
                    if result == "Draw":
                        reset_game()
                        break
```

```python
    if res.multi_hand_landmarks:
        for landmarks in res.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, landmarks, mp_hand.HAND_CONNECTIONS,
                                      mp_styles.get_default_hand_landmarks_style(),
                                      mp_styles.get_default_hand_connections_style())
```

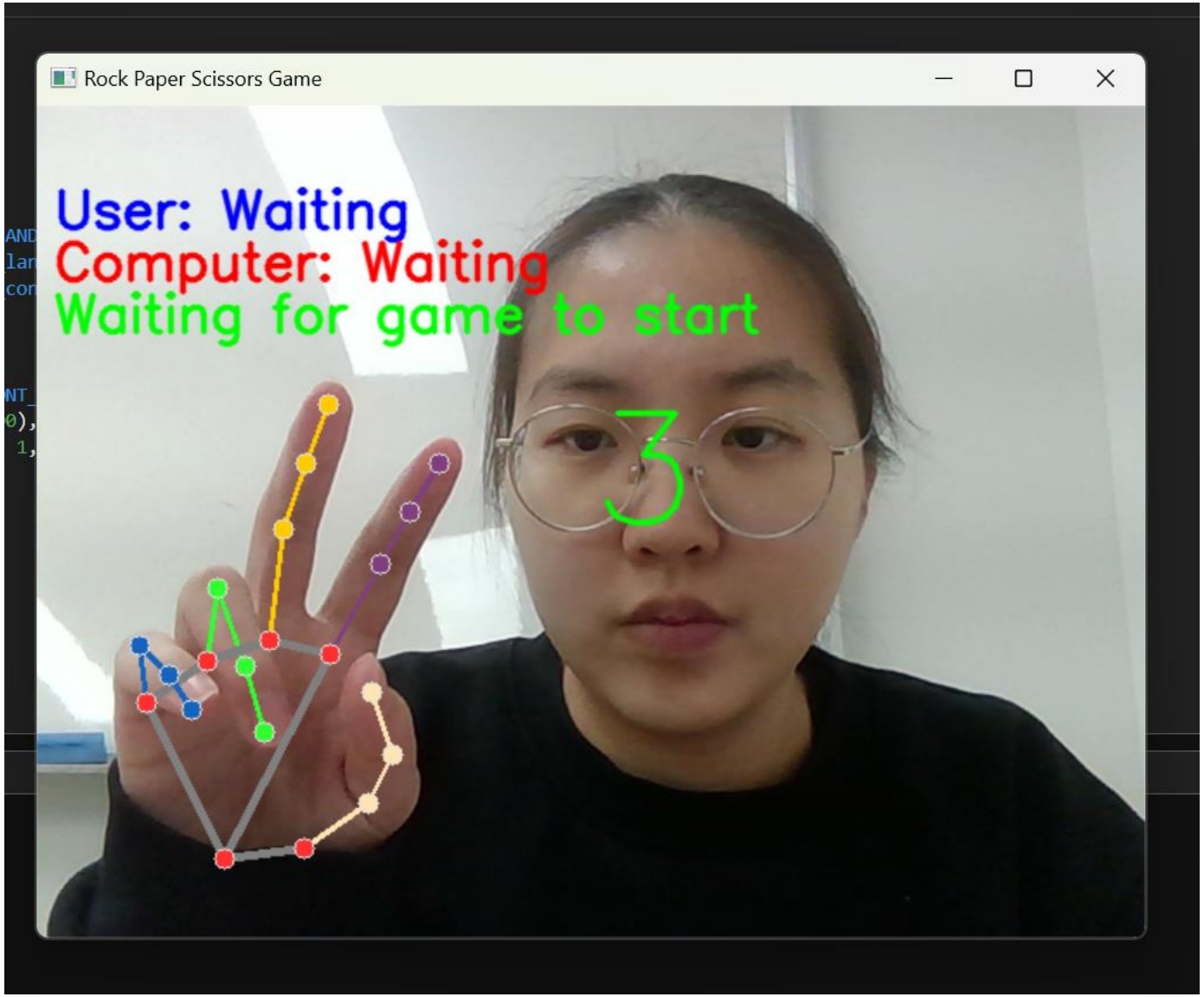
제대로 감지를 못하거나
비겼을 경우 다시 시작할 수 있게
해주기 위해 reset_game 함수를
만들었다

또한 시간을 5,4,3,2,1 이런 식으로
화면에 표시할 수 있도록 하였다

+) 컴퓨터 손동작은 제대로 썼는데
사용자 손동작 scissors 스펠링
잘못써서 승패 결정이 제대로
안됐었다.....

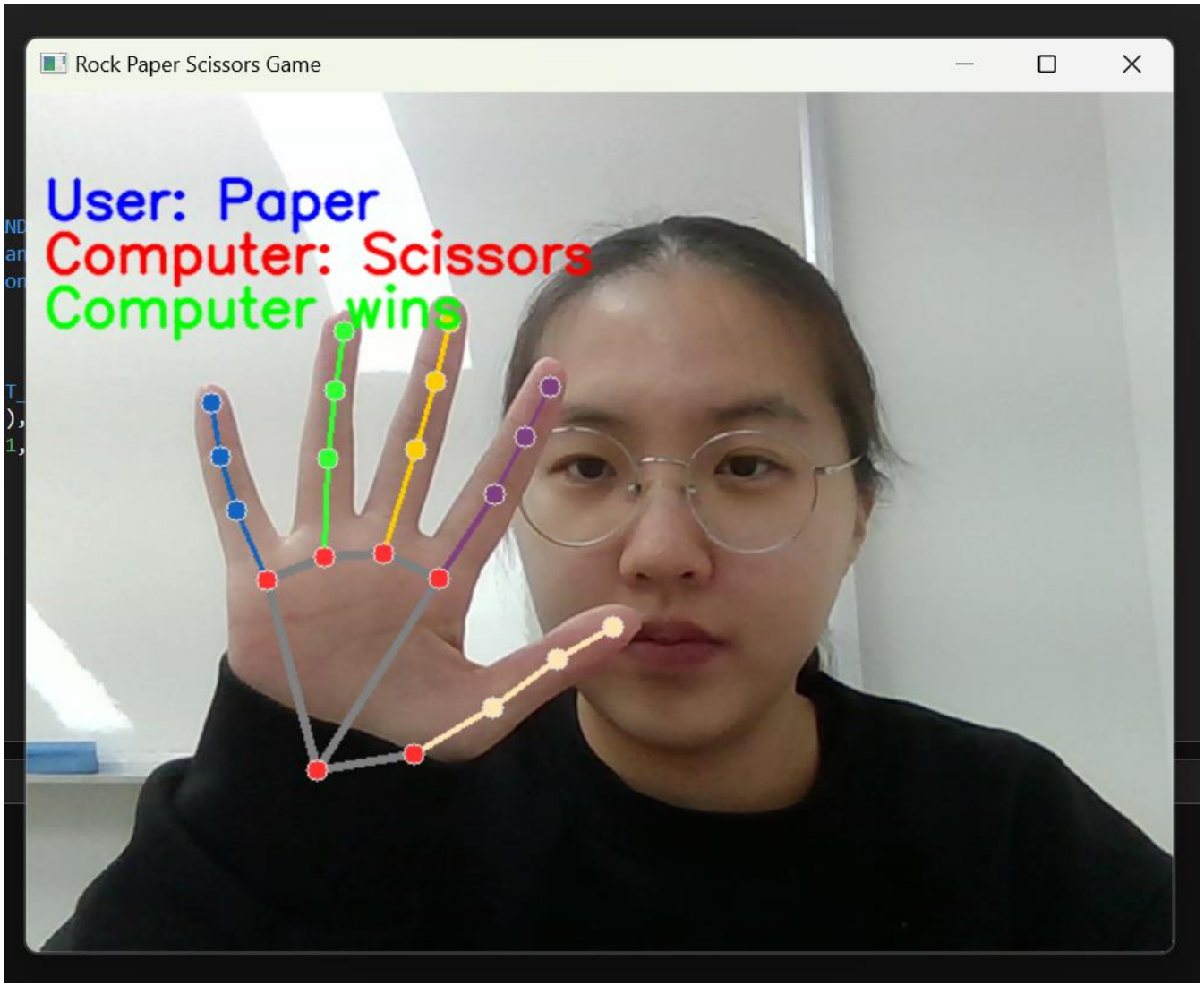
중간 결과

8. 시간 추가 / r을 누르면 게임을 다시 시작하는 기능 추가



중간 결과

8. 시간 추가 / r을 누르면 게임을 다시 시작하는 기능 추가



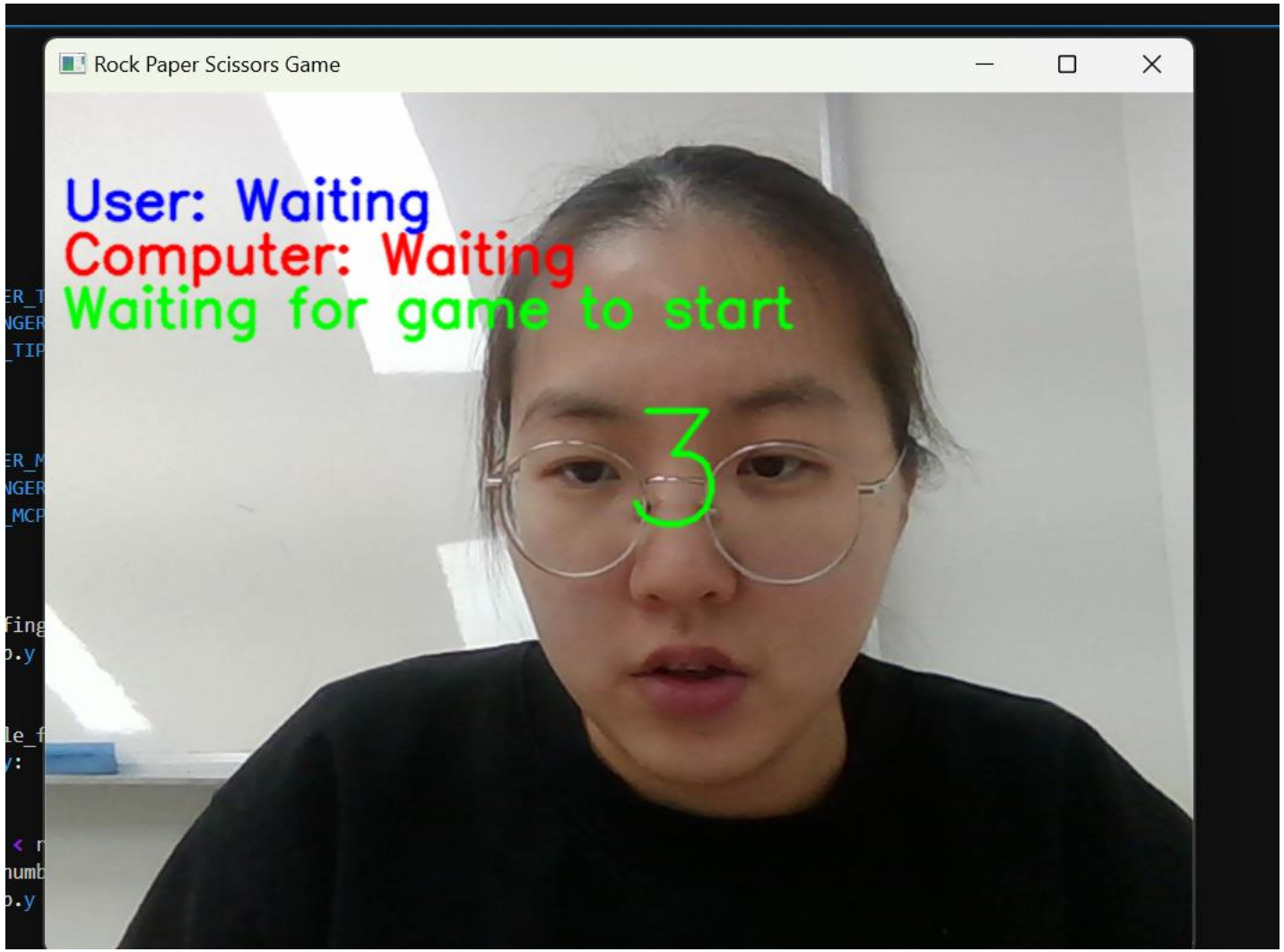
구현

9. 손 감지를 아예 못 할 경우 게임을 다시 시작하는 기능 추가

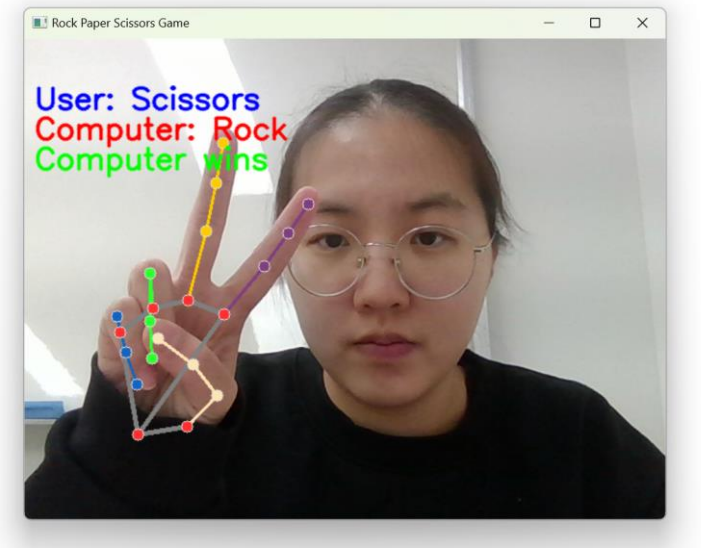
```
if not res.multi_hand_landmarks and time.time() > game_start_time:
    reset_game()
    continue

#추가된 부분
# 시간 카운트다운 표시
time_left = int(game_start_time - time.time())
if time_left > 0:
    cv.putText(frame, str(time_left), (frame.shape[1] // 2, frame.shape[0] // 2), cv.FONT_HERSHEY_SIMPLEX, 3, (0, 255, 0), 2, cv.LINE_AA)
else:
    if not game_started:
        game_started = True

        #손을 감지했을 경우
        if res.multi_hand_landmarks:
            for landmarks in res.multi_hand_landmarks:

                user_hand_shape = user_choice(landmarks)

                #제대로 감지 못하면 다시 시작
                if user_hand_shape == "null":
                    reset_game()
                    break

                computer_hand_shape = computer_choice()
                result = winner(user_hand_shape, computer_hand_shape)

                #비기면 다시 시작
                if result == "Draw":
                    reset_game()
                    break

if res.multi_hand_landmarks:
    for landmarks in res.multi_hand_landmarks:
        mp_drawing.draw_landmarks(frame, landmarks, mp_hand.HAND_CONNECTIONS,
                                  mp_styles.get_default_hand_landmarks_style(),
                                  mp_styles.get_default_hand_connections_style())
```

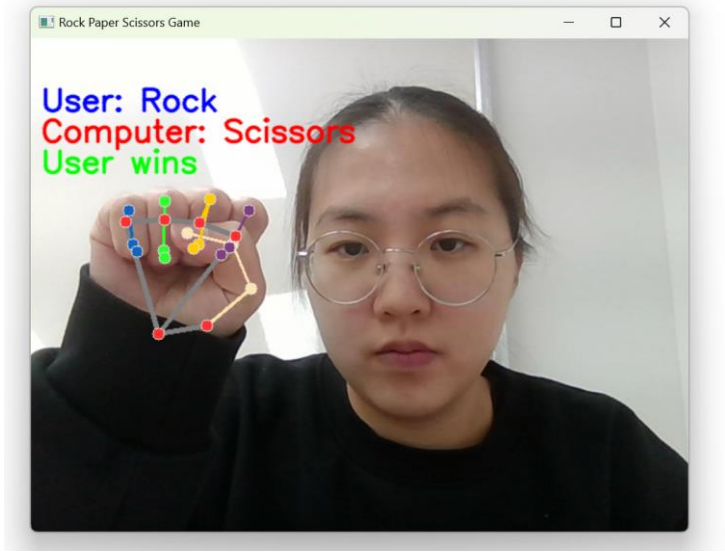
현실에서 '안내면 진거 가위 바위 보 ' 라고
하는게 생각이 나서
안낼 경우를 추가하고 싶었다
처음에는 안냈을 경우 진걸로 하려다가
5초후에 아무것도 안냈을 경우에도 다시
게임을 시작하지만
결과가 나온 다음
R버튼을 누르지 않고 손만 내려도
게임을 다시 시작할 수 있게 하면
게임을 할때 더 편리하다고 생각해서
reset_game()을 하고 continue하게 함수를
만들어줬다

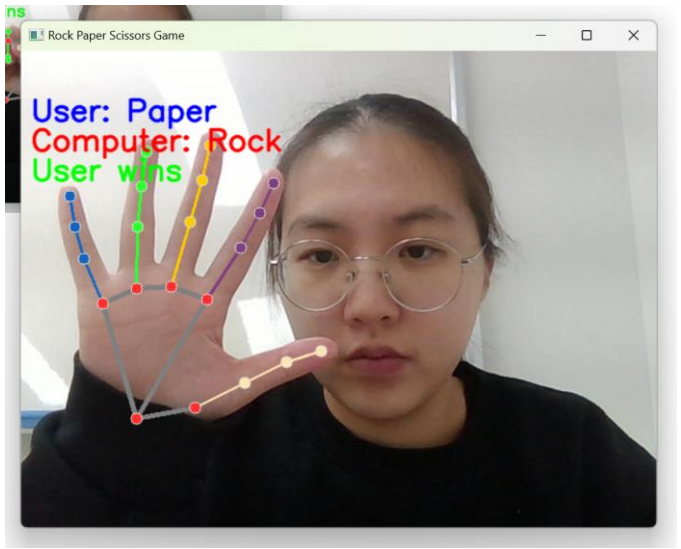
중간 결과


Rock Paper Scissors Game
User: Waiting
Computer: Waiting
Waiting for game to start
3

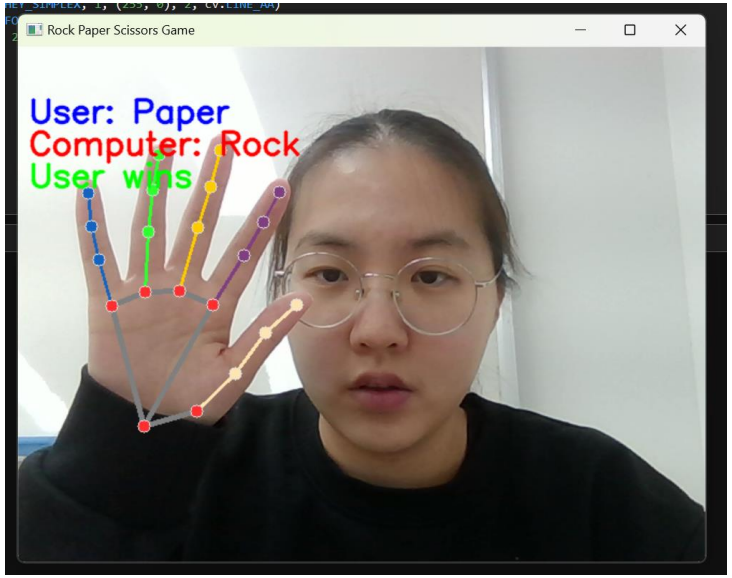
최종 결과

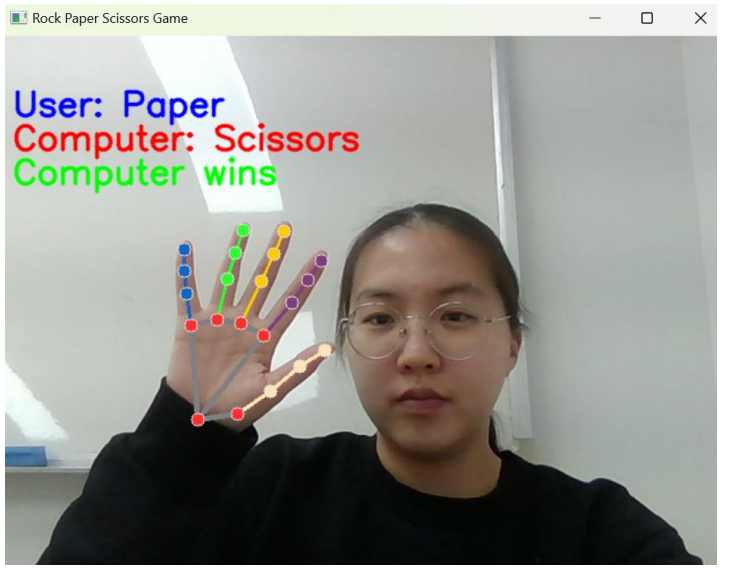

Rock Paper Scissors Game
User: Scissors
Computer: Rock
Computer wins


Rock Paper Scissors Game
User: Rock
Computer: Scissors
User wins


Rock Paper Scissors Game
User: Paper
Computer: Rock
User wins


Rock Paper Scissors Game
User: Paper
Computer: Rock
User wins


Rock Paper Scissors Game
User: Paper
Computer: Rock
User wins


Rock Paper Scissors Game
User: Paper
Computer: Scissors
Computer wins

논의

바위와 가위 같은 경우

손가락 끝 랜드마크만 이용하여 조건을 설정하였는데

모든 손 랜드마트를 활용하여 조건을 설정하면 정확도를 더 높일 수 있을 것 같다

또한 게임을 진행한 뒤

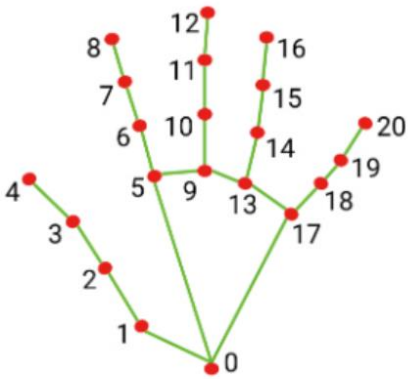
웹캠을 끄고

게임을 다시 진행하겠냐는 문구만을 화면에 나타내고

Y를 누르면 게임을 다시 진행하고

N을 누르면 게임을 종료하는 식으로 프로그램을 개선한다면

더욱 게임 같은 게임이 될 것 같다



또한 이 프로그램을 활용하여

가위 바위 보의 응용 버전 게임인 '묵 찌 빠 ' 도 만들어 볼 수 있을 것 같다

추후 발전 방향

모든 손 랜드마크를 활용하여 더 미세한 손 동작 인식 프로그램을 만들어서

가위 바위 보 게임에서 더 나아가 수화 인식 프로그램을 만들어볼 수도 있겠다

또한 자세 추정 프로그램이랑 함께 활용하여

‘디비디비딥’ 게임을 만들어보는 것도 재미있을 것 같다

결론

수업시간에 배웠던 [손 랜드마크 검출하기] 프로그램을 활용하여

가위 바위 보 게임을 만들어보았다

제대로 동작하는 게임을 만들었을 뿐만 아니라

수업시간의 코드를 조금씩 고쳐나가며 게임을 완성하였고

딥러닝 응용 수업을 들었던 학생이라면

바로 이해할 수 있는 쉬운 코드로 구현을 했다는 점에서 의미가 있는 것 같다

한 번에 가위 바위 보 게임을 구현한 것이 아니라

작은 프로그램들을 모아서 하나의 재미있는 게임을 만들어냈고

중간 중간 잘 실행이 안되는 부분을 고쳐가면서

현실 세계의 가위 바위 보 게임과 비슷하게 완성시켰다