HAND 2048

HAND 2048

AI LAB PROJECT

Cinelli Francesca e Pennini Chiara

Student IDs:

2046606 & 2053997

**1. INTRODUCTION**:

With our project, we aimed to combine real-time hand recognition and the popular game 2048, allowing players to use hand gestures captured in real-time by the camera as input. We also implemented a set of commands to interact not only with the game but also with the GUI, offering the possibility to play entirely without the use of a keyboard and mouse.

2048 is a single-player game where the player must combine tiles with the same number to achieve higher numbers, trying to reach 2048 before running out of space on the grid. The tiles can move in four directions on a 4x4 grid, sliding until they encounter another tile or the grid's edge.

Swipe Right



Example of an action in the game

There are many projects that employ hand recognition, for instance, in the field of virtual reality, sign language interpretation, or human-computer interaction. Nonetheless, we sought a new, original way to exploit this widely known and used model.

**2. Method**:

The project consists of three main parts:

1. Hand recognition using MediaPipe and OpenCV
2. Interpretation and conversion of gestures into commands
3. Sending commands to the game

2.1 **Hand Recognition**

n this part of the project, we use the OpenCV and MediaPipe libraries to recognize hands from a video captured in real-time. MediaPipe is an important and powerful library developed by Google, offering real-time tracking solutions for hands and fingers.

Indicating that the model is being used for real-time tracking and not for static images, we started by initializing the MediaPipe hands module, which requires specifying some parameters:

with mp\_hands.Hands(

        static\_image\_mode=False,

        max\_num\_hands=1,

        min\_detection\_confidence=0.5,

        min\_tracking\_confidence=0.5) as hands:

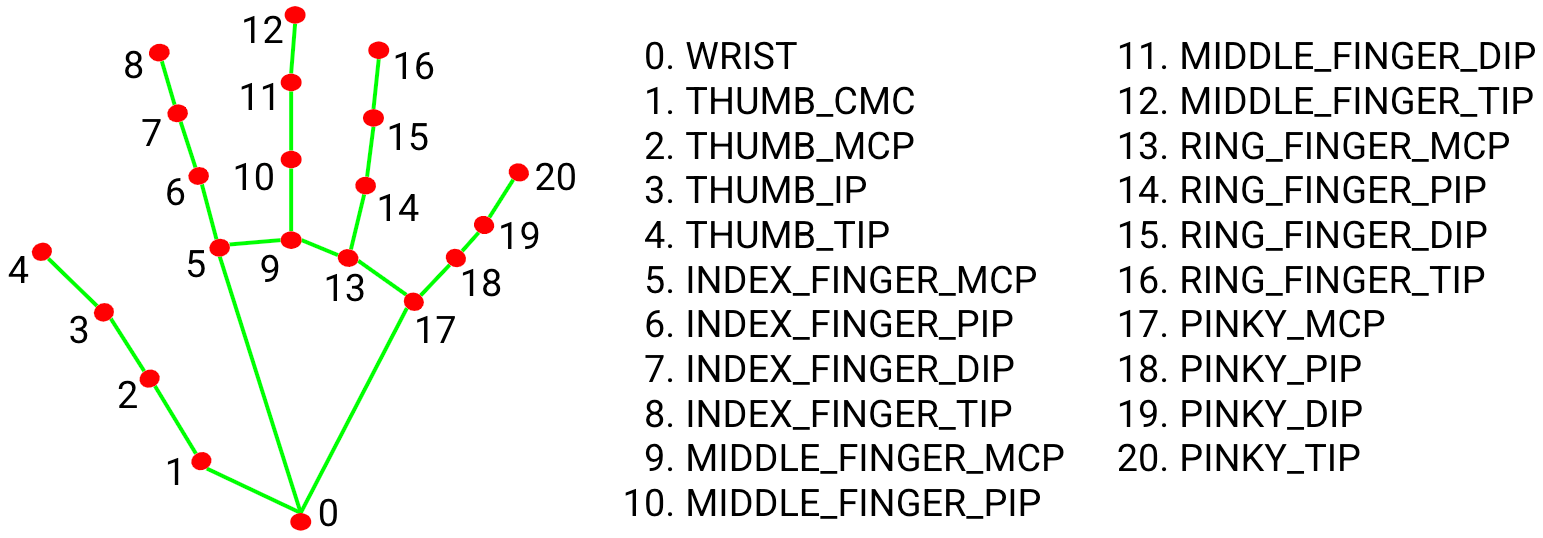
Il massimo numero di mani rilevate contemporaneamente è 1

Soglia di confidenza minima per rilevare e tracciare una mano

After capturing the webcam footage via OpenCV, we processed each frame for hand tracking using the **process()** method, which returns information on the detected hands, including the landmarks. The landmark list contains the 21 reference points of the hand, and each point is an (x, y, z) object. The obtained coordinates for each reference point are normalized, meaning they range from 0 to 1, where (0, 0) is the top-left corner of the image and (1, 1) is the bottom-right corner. Since we use a mirrored image to achieve a mirror effect, these coordinates are also mirrored.

Once we have the hand points, we can use them to recognize specific gestures.

*-List of the 21 hand landmarks in yellow, those used in the project for gesture recognition-*



2.2 **Interpretation and Conversion of Gestures into Commands**

The most challenging part of this project was actually finding unique gestures that could be interpreted by the program as commands and not just simple hand movements. We started by defining the basic movements necessary for the game to understand in which direction to send the tiles. This is done by continuously reading the position of the index finger and, based on the direction the finger moves, converting it into the Left, Right, Up, or Down command, which is then sent to the game.

To achieve this, we compared the current position of the index fingertip with its position a moment earlier, and by doing so, while always considering a certain margin of error, we were able to interpret in which direction the hand was moved.

Since our initial idea was to interact not only with the game but also with the GUI, enabling mouse movement, clicking, and moving windows and objects on the screen, we had to identify unique gestures that would correspond to the various actions mentioned above.

Before implementing this, we decided to add a game pause mode where the hand could be moved freely without movements being interpreted as game commands. We thus associated a gesture with the transition from game state to pause state.

Moreover, since our project aimed not only to interact with the game without the keyboard but also with the entire GUI, we needed a way to distinguish when the index finger should be interpreted as a mouse pointer and when as a reference point for moving tiles on the grid. We thus identified a second gesture indicating the transition from pause mode to mouse mode, where it would be possible to use the mouse to click and move objects on the screen.

Then, in this last mode, we had to define additional gestures for clicking and moving objects.

*-Flow chart of program states-*

thumb+pinky

PLAY GAME

PAUSE GAME

MOUSE MODE

MOVE OBJECTS

CLICK BUTTONS

thumb+pinky

thumb+middle

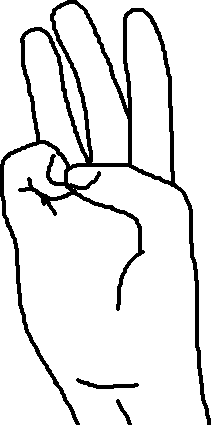
thumb+ring

thumb+ring

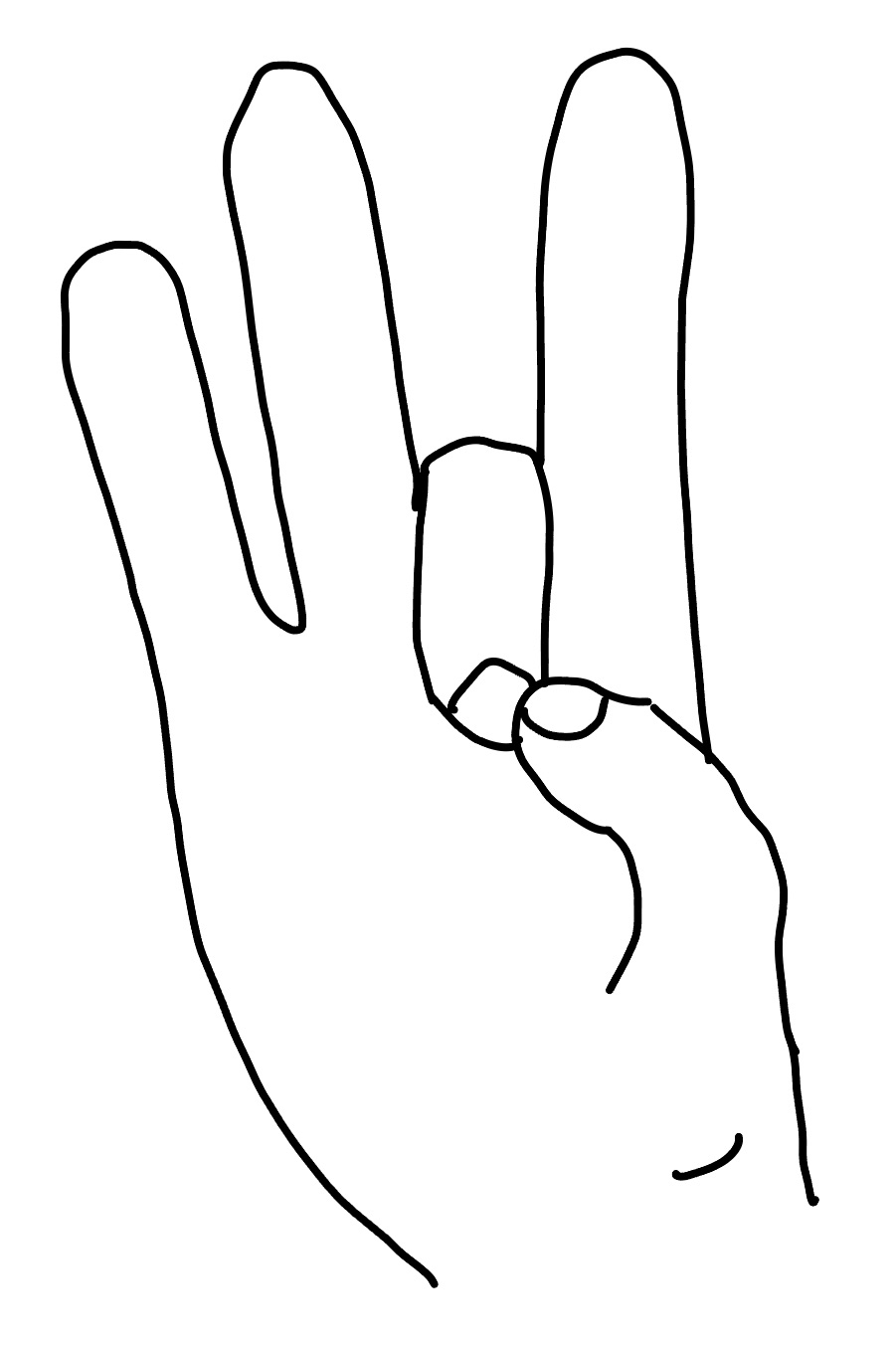
(hold)

*If the program switches from mouse mode back to play mode, it will automatically activate the mouse when it returns to pause mode*

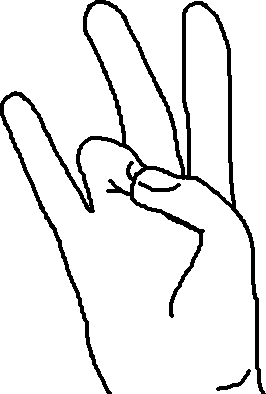
Below we list all the gestures and corresponding actions:



* Thumb tip touches pinky tip: the program enters **PAUSE** mode;
* Thumb tip touches middle finger tip: the program enters **MOUSE** mode, and the index finger, now functioning as a pointer, is marked with a colored circle;



* Thumb tip touches ring finger tip: the program performs a **CLICK**;
* Thumb tip touches ring finger tip for an extended time: the program performs a long click, allowing objects to be moved on the screen.



At the end of the game, when the desired number (which could be 2048, 1024, 512, 256) is reached, or no more moves are available, the game requires keyboard input (Y or N based on whether the player wants to continue playing a new game or exit). Therefore, we identified two more gestures listed below:

* Thumb tip touches pinky tip: sends the **NO** command to indicate the end of the game;
* Thumb tip touches index tip: sends the **YES** command to indicate the desire to continue playing.

Although the first of these gestures is already associated with the pause, there is no problem because, in the final state, the game only interprets the specific movements listed above, ignoring the previous ones.

2.3 S**ending Commands to the Game**:

The 2048 game implemented in Python requires the player to specify the direction to move the tiles by pressing the WASD keys.

To convert the finger movement into actual game movements, we used the keyboard library, which simulates key presses using the press\_and\_release method.

Regarding mouse movement, once the program detected the gesture associated with the transition from pause mode to mouse mode, we used the pyautogui library and the moveTo method, which moves the mouse pointer to the (x, y) coordinates given by the index fingertip's position at the time of the function call.

To simulate mouse clicks, we used the same library but utilized the click method. For moving objects, we used a combination of the mouseDown and mouseUp methods.

**3. RESULTS:**

3.1 **Project Output**

The visual output shown to the player when the program is started is:

* Video Capture: real-time video captured from the device's webcam. Text specifying the recognized action will appear overlaid on the video. Additionally, the end-game rules will be displayed at the bottom of the screen to keep them handy at the time of game over or victory.
* 2048 Game: the main menu of the game will appear, where players can choose game options and consult the rules.

**4. CONCLUSION**:

In conclusion, hand recognition is undoubtedly used for much more significant purposes than a simple computer game. However, we can also confidently say that this small program could easily be used to implement a virtual arcade where, in addition to having the 2048 game, there are many other games that typically use the mouse and WASD keyboard keys, which was our original idea that we couldn't realize due to a lack of time and Python-implemented games. Moreover, this program could be used to operate a computer remotely since the functions we usually perform on the computer involve using the mouse and keyboard, which we have completely converted to hand use. These and many other are the future prospects of this project.

5.REFERENCES: