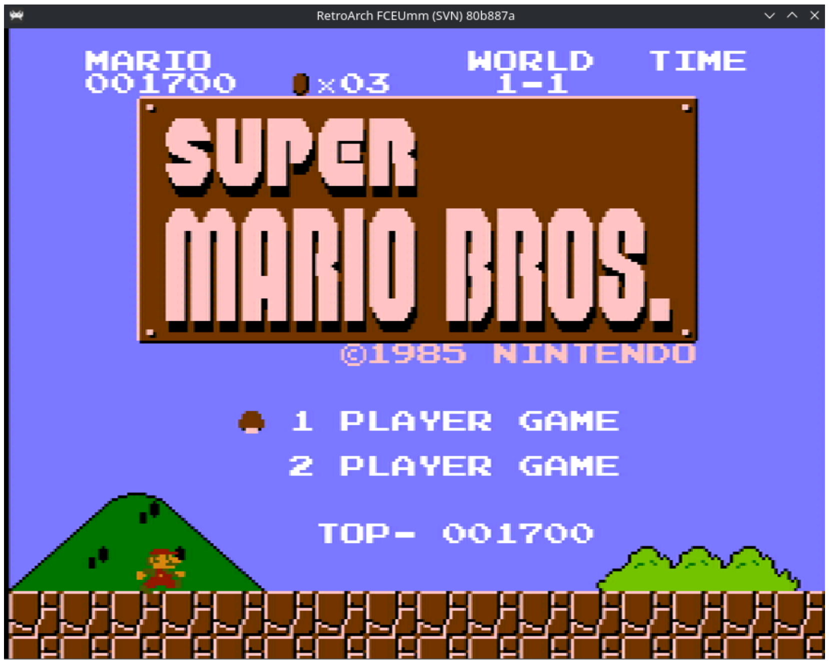
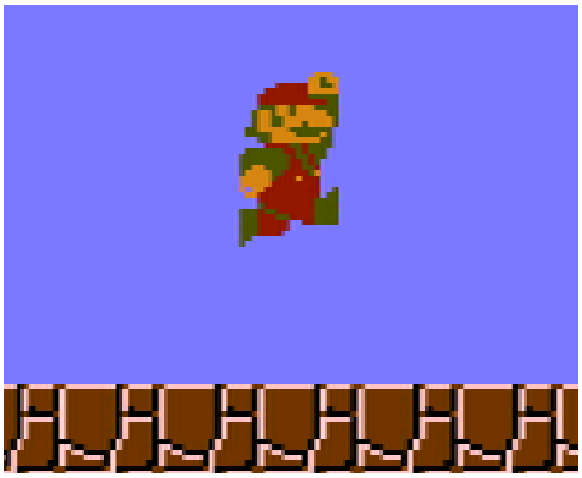
# Meeting 2 - 06.10.2025

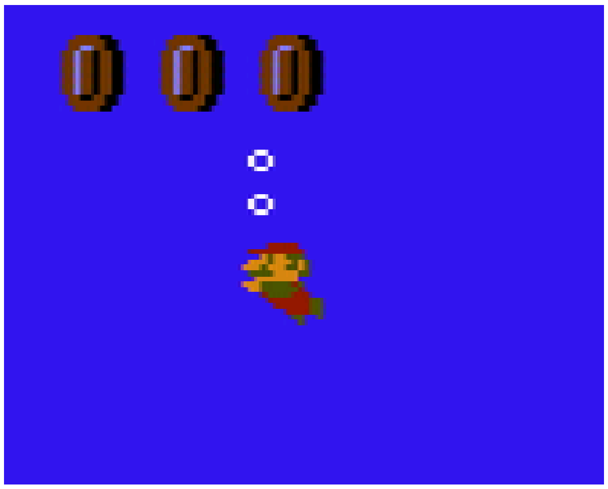
## Mario-Game Controls

**Menu Controls**
- START-Button to choose an option
- SELECT-Button to switch between options

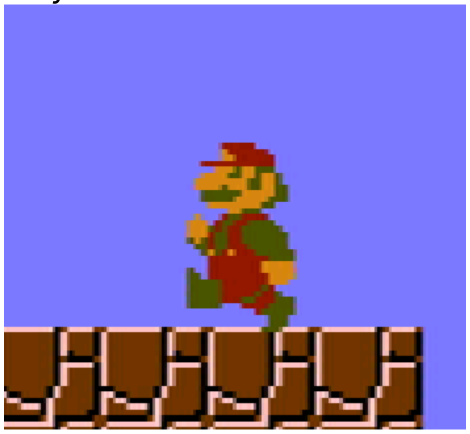


**Jump / Swim**
- At first the Jump and Swim action seem kind of simmilar but there are differences
- Press the A-Button to jump or swim (while in the water)
- Able to control the jump height depending on how long the jump button is held
- Pressing LEFT or RIGHT during a jump / swim gives you more air control
- A swim action has a set distance, not based on how long you hold the button, so maybe swim should be its own motion input

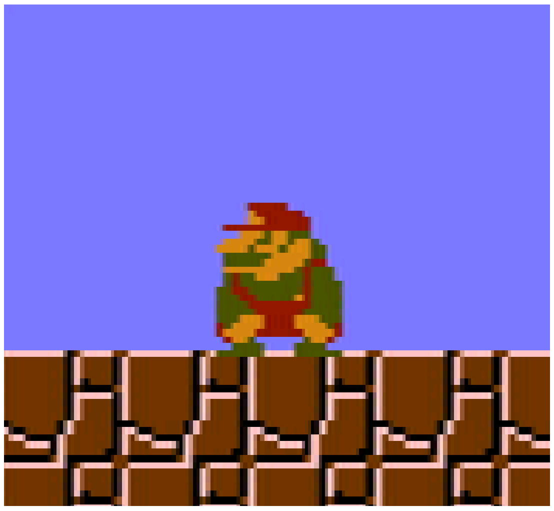




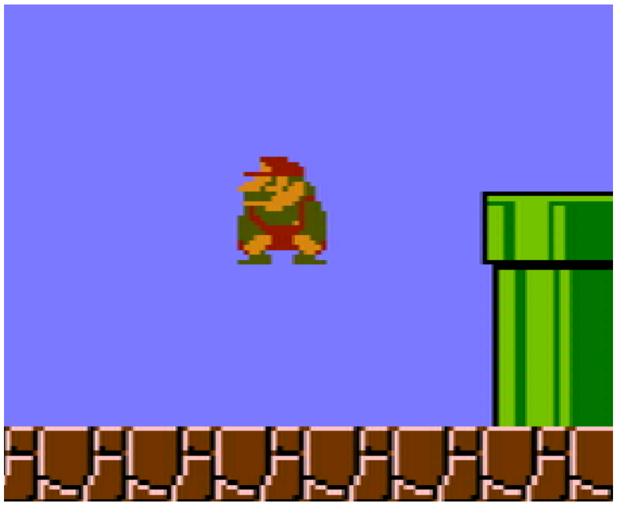
**Walking**
- Mario is able to walk by pressing LEFT and RIGHT
- If you hold the B-Button Mario runs, making him faster



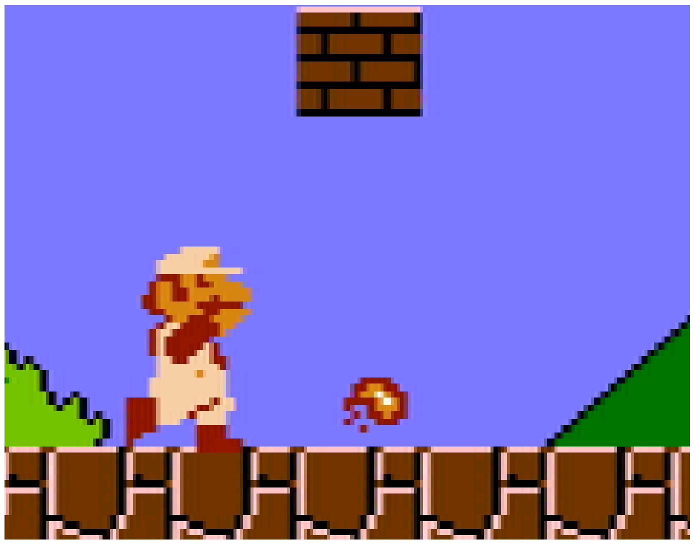
**Crouch**
- Mario can crouch when he is Big Mario (after eating a Mushroom) by pressing DOWN
- mostly used to slide on the ground after running
- Small and Big Mario can enter pipes by crouching



**Crouch Jump**
- holding DOWN and pressing the A-Button at the same time
- needed to get out of situations where Mario is trapped in a gap of blocks



**Throw Fireball**
- When Mario picks up a Fire Flower he turns into Fire Mario
- When you press the B-Button a single time, he throws a fireball
- You can press the B-Button rapidly to quickly throw fireballs consecutively



## How are we going to send inputs to the game?

We have two options:
1. Edit the source code to support our Motion Controls natively
2. Write our Python Program as an independent piece of software, that delivers keystrokes externally to the game (no need to know the code of the game)

## What version of the game should we use?

**Original Super Mario Bros.**

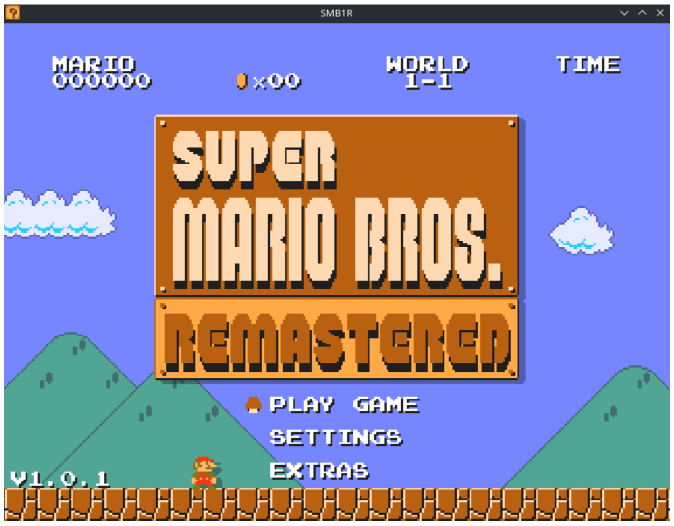
Pros:
- its the original game
- we don't have to think about editing the games code
- predictable game logic

Cons:
- requires a little setup (getting the rom and emulator, even though its quite simple)

We would only would have to write a Python Program that gives keystrokes to the game.

How to install?
Go to Steam: Install RetroArch (comes bundled with multiple emulators)

**Unofficial Super Mario Bros. Remastered**
https://github.com/JHDev2006/Super-Mario-Bros.-Remastered-Public

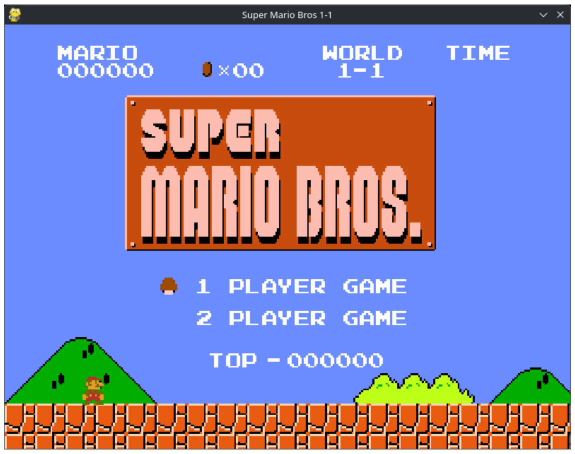


Pros:
- updated visuals
- native Windows / Linux binary
- is being actively developed (just released recently), which means communication with the developers is possible

Cons:
- very big project written in GDScript using the Godot engine (python like)
- also needs a rom in order to start (due to legal reasons)

If we would want to call the movement functions with our own Application, we would need to edit the source code of the Game. An approach would be to use Python sockets to establish a Server/Client relation between our program and the game.

**“Mario-Level-1”**
https://github.com/justinmeister/Mario-Level-1

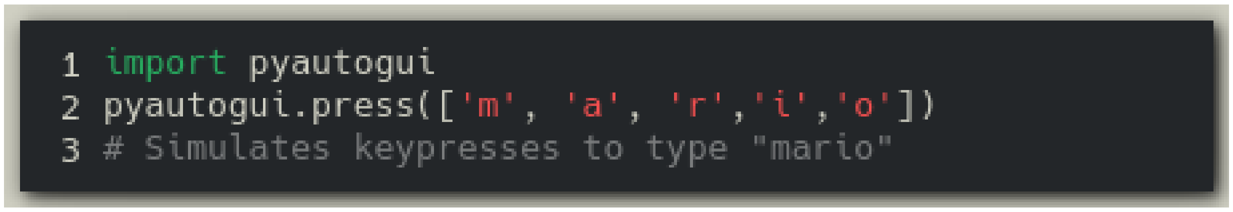


Pros:
- readable python code

Cons:
- the repository is 12 years old and uses Python 2 (we might have to migrate it to Python3 if we want to extend it)
- only the first level
- momentum is missing from the physics
- concern about legal issues since the game comes with Mario assets pre packaged (Copyright Infringement)

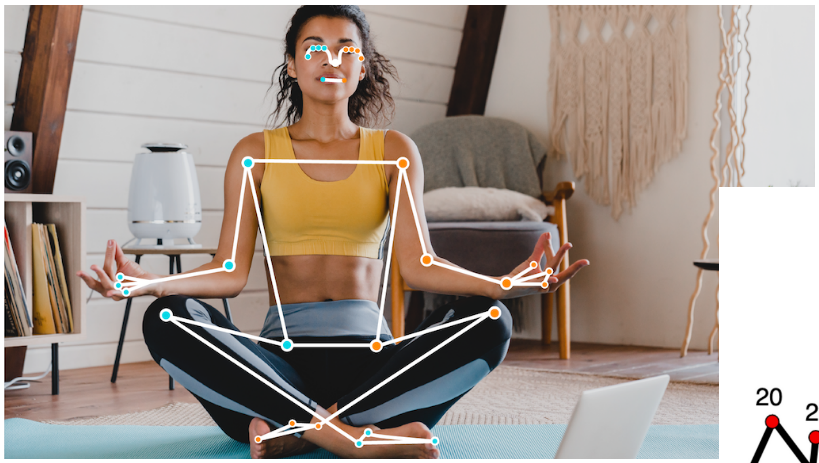
**So how are we going to simulate Keypresses with Python?**
There is a Python library called PyAutoGUI, for that exact use case.
https://pyautogui.readthedocs.io/en/latest/

https://pypi.org/project/PyAutoGUI/

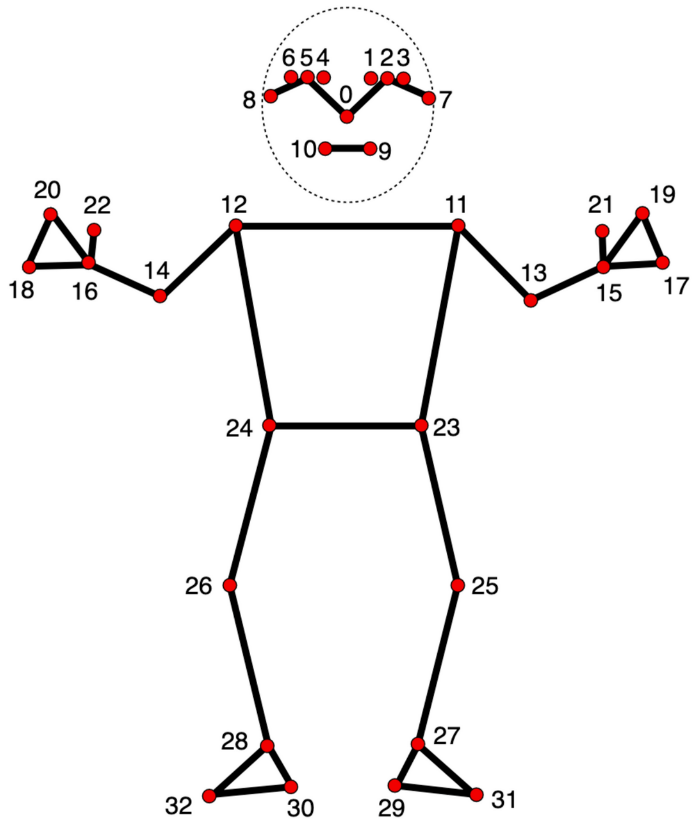


```python
import pyautogui
pyautogui.press(['m', 'a', 'r','i','o'])
# Simulates keypresses to type "mario"
```

**How do we capture our webcam and use its live feed to interpret motions?**
We will be using OpenCV combined with googles mediapipe
https://pypi.org/project/opencv-python/
https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker

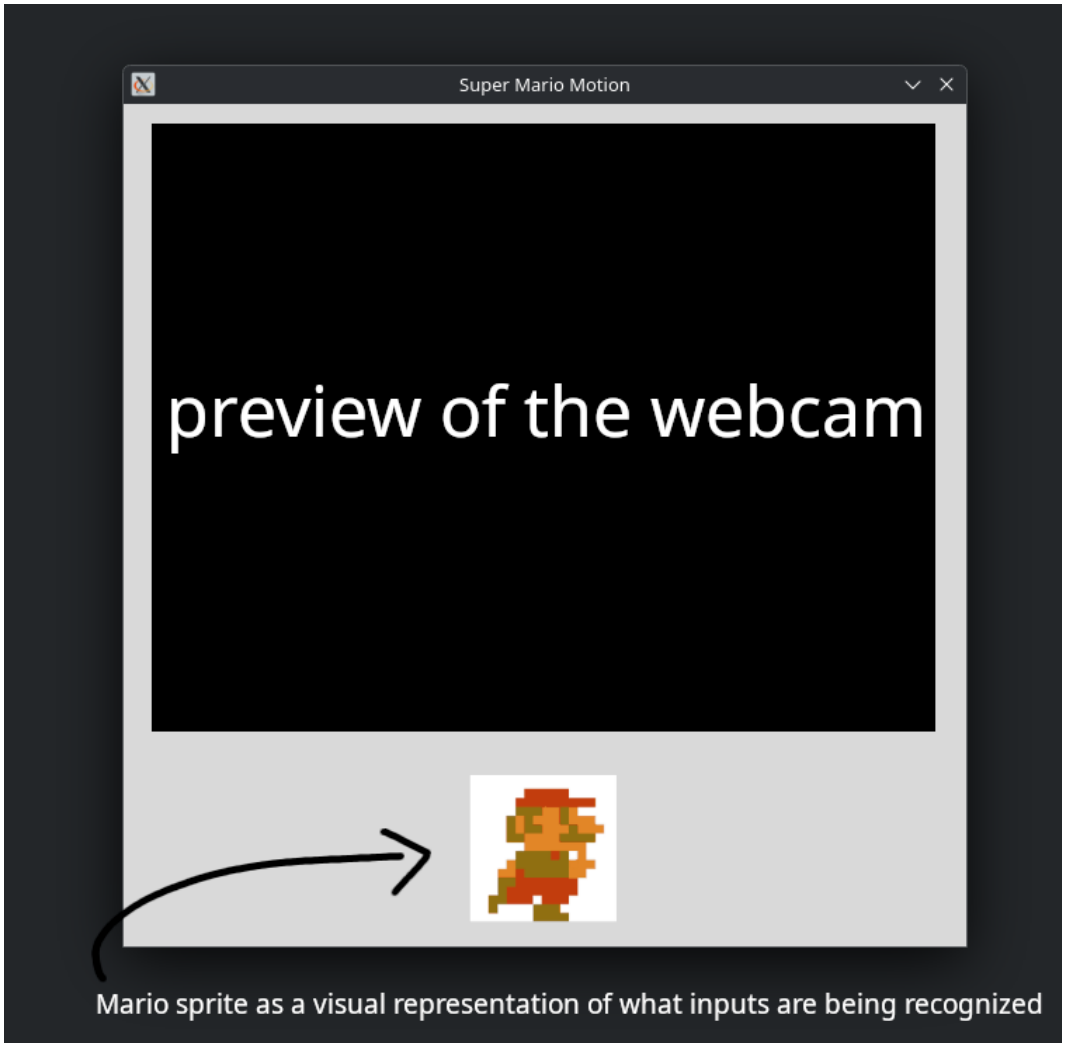


*https://ai.google.dev/static/mediapipe/images/solutions/examples/pose_detector.png*



*https://ai.google.dev/static/mediapipe/images/solutions/pose_landmarks_index.png*

## Very first mockup of the user interface


Super Mario Motion
preview of the webcam
Mario sprite as a visual representation of what inputs are being recognized