Meeting 5 – 4.11.25

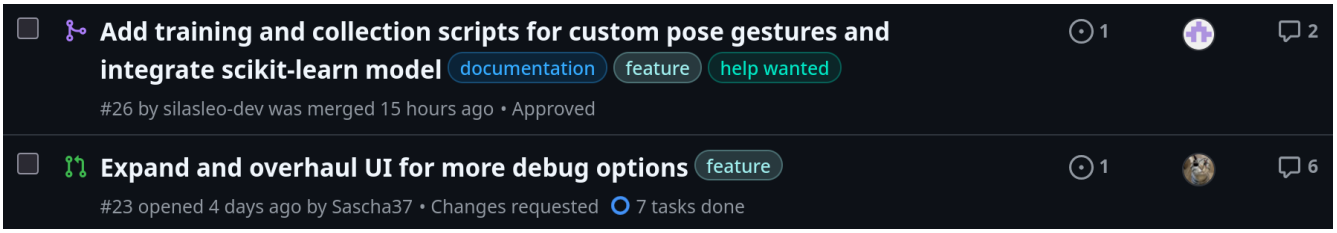
Progress Report

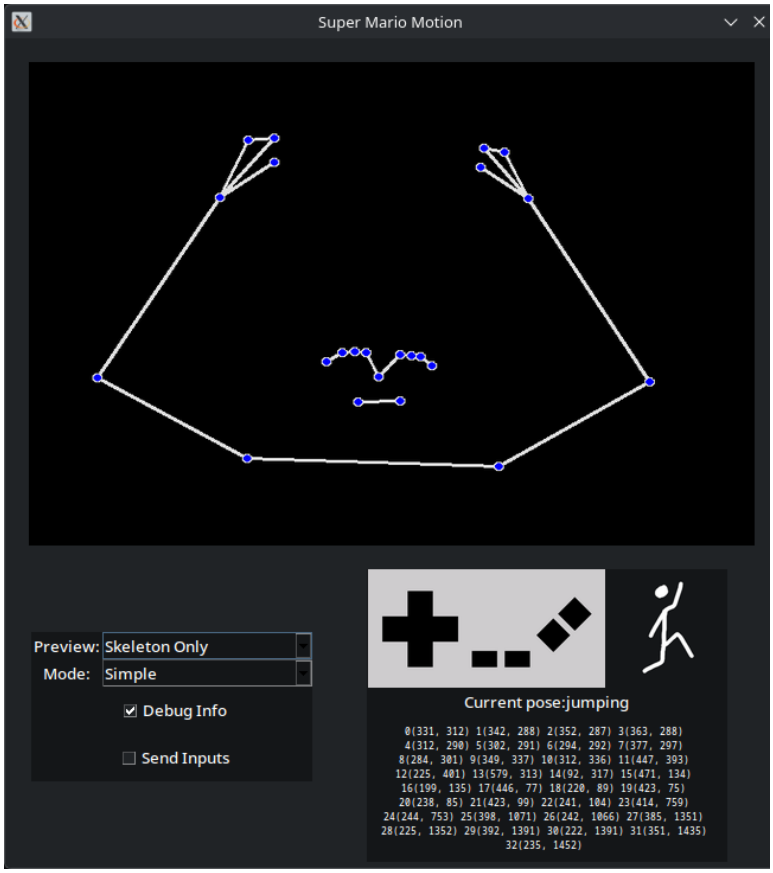
This week we tackled three things

1. Thread Management: We got together and worked out a solution to get threads to terminate correctly when the program closes. While doing so we identified bugs that we can fix.

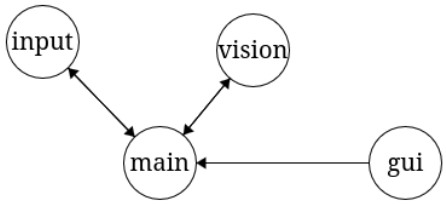
2. Full-Body Motion Controls: We now have expanded our program to support full-body motions, that can be trained by webcam recordings. Allowing us to go in to the main phase of the project.

3. New UI: We also extended our user interface to allow the user to select between both modes “simple” and “full-body”. Also the user can now set its webcam video to “Webcam”, “Webcam + Skeleton”, “Skeleton Only”. Lastly there is a new checkbox to display Debug information





Latency problem of our program



main.py: inits input and vision, starts the gui and runs the update loop
input.py: runs in a thread, presses buttons based on pose
vision.py: runs in a thread, main opencv/mediapipe logic

For now we have a loosely coupled approach, allowing main to handle getter and setter functions of vision.py and input.py.
Tkinter, our GUI library, needs to run on the main thread. The .after function of tkinter allows us to execute code while the UI is running.

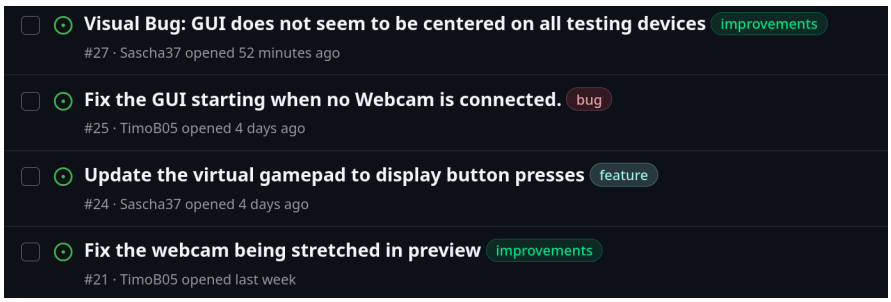
Currently now our loop, that we are using to update UI-elements and exchange information between Modules has a delay of 1ms. Tkinter only runs the function when it has the time to do so, meaning that 1ms is the **minimum possible delay,** in practice, the actual interval is even longer depending on hardware and other factors outside of our control.

This became a big problem when we had our program working. The delay while playing the game made it feel really sluggish.

Potential solution:
Only use our update function in main to update UI-elements.
Introduce a new module called state.py, containing variables for the current pose, images from OpenCV etc.
The module will have getter and setter, so that our modules can communicate with each other without going through our previous delayed loop.
One concern will be race conditions, but for the moment every module only writes to their dedicated variables, this should not cause issues in our current implementation.

Whats next?

- We have a lot of issues on github listed that need to be worked on



Main features to focus on next week include:

- Rewriting the way we pass on information between modules, eliminating delay
- Update README of our project (has not been kept up to date for three weeks)
- Once again extend UI to allow for easier capturing of training data
- Implement the virtual gamepad display in the GUI