Robot Control Server

Daniil Myronov
FIT CTU
myrondan@fit.cvut.cz

May 28, 2023

1 Task

The main goal of the server is working with distantly controlled robots and guiding them to the center of a simplified "map". Robots are controlled via the TCP connection, using predefined message structures. Server should be able to work with multiple connections simultaneously. Server should have an option of providing GUI, where it should display connections in real time, with the logs of communication and a map outline for each robot.

2 Methods for solving the task

2.1 Technologies

The task was implemented in Python programming language.

For the TCP communication, the `socket` package was used.

The logic for the communication, including authentication of the robots, was implemented as a finite-state machine. For defining the state machine, `transitions` library was used.

The graphical user interface was made using the `PyQt5` library.

2.2 Techniques

Asynchronous programming was largely used for this project to handle multiple connections concurrently. Separate threads are created for:

- GUI
- server, for accepting the connections
- each connection, for processing messages
- each connection, for listening to events and caching them in case the cannot yet be displayed by the GUI thread.

Regarding the last point, it is needed to say that the project uses *Event-driven architecture*. The *Observer pattern* is used for transferring events from the connection thread to the GUI (and possibly other consumers). In turn, worker threads that consume those events use *signal API* of the Qt framework to transfer the events to GUI.

As said earlier, a *finite-state machine* is used to keep track of current state of the robot and possible messages that the robot can send at the moment.

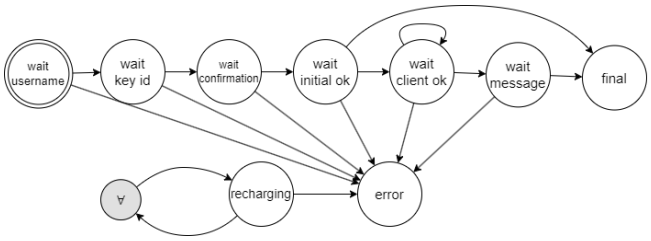

Figure 1: Simplified state machine

3 Results

As a result of dividing the application to the server and GUI modules and running the server independently of the GUI, it can be certainly said that the efficiency of the server isn't influenced by the capabilities of GUI.

Using the state machine makes the application more reliable and secure, as it prohibits unauthorized or invalid transitions between states and ensures that only allowed inputs are processed. This is especially crucial when dealing with data received from networks. Also, it makes expanding the app and locating the bugs much easier.

Using asynchronous worker threads deals with the race condition, when the server tries to answer robot's query as soon as possible, but the GUI is not yet ready to process the event of responding to a message.

4 Conclusion

The end goal of the assignment was satisfied. Still, improvements can be made regarding the user interface of the app, to make it friendlier and add the ability to control the server using it. The logic of the server could also be expanded if required by the specification.

References

[1] Martin Fitzpatrick. The complete pyqt5 tutorial. online, 2023. https://www.pythonguis.com/pyqt5-tutorial/.

[2] Alexander Neumann. Transitions repository. online, 2022. https://github.com/pytransitions/transitions.

[3] OpenAI. Chatgpt (mar 14 version). online, 2023. https://chat.openai.com/chat.

[4] Leodanis Pozo Ramos. Use pyqt's qthread to prevent freezing guis. online, 2020. https://realpython.com/python-pyqt-qthread/.