Self-evaluation

Name of project to be reviewed: Q-learning

Names of reviewers: Nico Åstrand, Marcus Myllyviita, Anton Eklund, & Axel Neergaard

The points we deem worthy for each list item can be found italicized under each item.

1. Overall design and functionality (0-6p)

1.1: The implementation corresponds to the selected topic and scope. The extent of project is large enough to accommodate work for everyone. (2p)

The implementation corresponds really well in our opinion. The project is built of four distinct parts, accomodationg clear and enough work for every group member. There was even more features that could have been, and was, added, giving even more work.

2/2 points

1.2: The software structure is appropriate, clear and well documented. e.g. class structure is justified, inheritance used where appropriate, information hiding is implemented as appropriate. (2p)

The structure of the software is consequent and easily understandable with a lot of comments and describing variable names. The documentation is extensive with drawings supporting the text. Class structures are well thought out and information hiding have the principle "only shown if needed". More information about this in the documentation.

2/2 points

1.3: Use of external libraries is justified and well documented. (2p)

The inline comments together with the documentation gives a good picture of how the external libraries are used. Many of the variable and type names also indicate what they do. All libraries used have a specific task for which they are needed, e.g. SFML for drawing, and no unnecessary libraries are added.

2/2 points

2. Working practices (0-6p)

2.1: Git is used appropriately (e.g., commits are logical and frequent enough, commit logs are descriptive). (2p)

Frequent commits to git with descriptive messages most of the time. Git was split up into multiple, but not too many branches. Old unnecessary branches were removed to make the important parts more clear and easy to find. Merge requests were used as the main contribution channel, proving extensive and proper use of Git version control.

2/2 points

2.2: Work is distributed and organised well. Everyone contributes to the project and has a relevant role that matches his/her skills. The distribution of roles is described well enough. (2p)

The work is distributed into four somewhat independent parts, one for each member. Although we still had the most experienced member keeping an eye on the whole project. Each members role is clearly described in the documentation. Much of the code was also worked on in pairs, increasing knowledge sharing.

1/2 points

2.3: Quality assurance is appropriate. Implementation is tested comprehensively and those testing principles are well documented. (2p)

Testing has been done throughout the development of the software. All modules has been tested both separately before implementing in the rest of the program as well as as a whole in integration testing phases. Testing has been done in most parts manually, but also with some unit tests. Please refer to the documentation for justification on heavy manual testing.

2/2 points

3. Implementation aspects (0-8p)

3.1: Building the software is easy and well documented. CMake or such tool is highly recommended. (2p)

Building is done with the help of CMake. The instructions for building is well documented and does not require many steps. The software is at least

compiling and running on Linux – tested on Aalto computers – and should also work on Windows and MacOS.

2/2 points

3.2: Memory management is robust, well-organised and coherent. E.g., smart pointers are used where appropriate or RO3/5 is followed. The memory management practices should be documented. (2p)

Smart pointers are used where appropriate and possible. RO3/5 has been considered, but ultimately not been necessary for the project. Memory management practices have utilized valgrind and work without memory issues on Aalto computers, documentation of this can be found in the documentations file.

2/2 points

3.3: C++ standard library is used where appropriate. For example, containers are used instead of own solutions where it makes sense. (2p)

The standard library is used extensively throughout the project, with no unnecessary extra data structures created. Some structs can be found, but these are justified by themselves.

2/2 points

3.4: Implementation works robustly also in exceptional situations. E.g., functions can survive invalid inputs and exception handling is used where appropriate. (2p)

Most functions are designed to handle input properly, throwing errors where such is necessary. There have not been proper needs for exception handling, so this has not been implemented properly.

1.5/2 points

4. Project extensiveness (0-10p)

Project contains features beyond the minimal requirements: Most of the projects list additional features which can be implemented for more points. Teams can also suggest their own custom features, though they have to be in the scope of the project and approved by the course assistant who is overseeing the project. (0-10p)

Please keep in mind that defining the requirements for this project is very difficult on our part as it is an own suggestion and there is no proper counterpart in the suggested projects list.

The minimal requirements list set up in our Project Plan has been met, and gone beyond. Our project contains a working project with bots learning on their own, with a lot of options during run-time.

The project contains a module for q-learning that does the actual machine-learning parts of the project. This works properly and is contained within one class for the sake of data and information containerization.

A module for bodies has been implemented that can easily be remodeled into an extendable class. Refer to the documentation of a short introduction to class hierarchy in this project.

A module for brains has been implemented that separates concerns between q-learning, the body, and the world itself.

The world module is small but powerful. It relies heavily on the physics of the Box2D library, but it is a nice interface to query, and is again a good way for separation of concerns.

The module for running the program itself and containing all parts has been implemented and does not concern itself with aspects that could be done with companion objects.

The most lacking feature – as per our Project Plan – is the GUI of the project. Unfortunately we were not able to create a proper graphical user interface with QT. However, this has been reworked instead to work with the modules that we currently have. For a more extensive explanation of this, please consult our documentations file.

9/10 points