Respondent: **Helena Båtsman** Submitted on: Monday, 18 December 2017, 1:09 AM

Mid-term review of C++ project

Name of the project group evaluated

micro-machines-5

This is a project self-evaluation

☐ Yes

C1.1: The implementation corresponds to the selected topic and scope. The extent of project is large enough to accommodate work for everyone (2 p)

This is the final review of the project, not mid-term review as wrongly stated in the headline

The project was done according to the instructions provided for Micro Machines -game: the purpose of the game is to race against other players and there are more than just one player and track available. According to the documentation and contributors statistics available in git, the workload was divided well among the group members.

C1.2: The class structure, information hiding and modularization is appropriate, and it is explained and justified in documentation. The file structure corresponds to the class structure (2 p)

The class structure and the overall software structure is explained extremely accurately. However, in order to explain the structure and class relationships more efficiently (and to get the 'big picture' quickly) the software structure section should be more compact. Therefore, the class relationships shown in Figure 1 could also be simplified.

C1.3: Use of at least one external library (in addition to C++ standard library). Comment the appropriateness of libraries and their use. (2 p)

Box2D (physics of the vehicles and collision detection) and SFML (UI, graphics) were used appropriately.

C2.1: Git is used appropriately (e.g., commits are logical and frequent enough, commit logs are descriptive) (2 p)

Git is used appropriately (86 commits) and the commit logs are descriptive enough to get the big picture.

C2.2: Make or Cmake (recommended) is used appropriately. The software should build easily using these tools without additional tricks. Nevertheless, instructions for building the project should be provided (1 p)

Instructions for building and using the software are documented clearly enough. Also, the steps are extremely user-friendly since they are easy to execute. Cmake is used.

C2.3: Work is distributed and organised well, everyone has a relevant role that matches his/her skills and contributes project (the distribution of roles needs to be described) (1 p)

As already answered to C1.1, the workload was divided well among the group members. The division of work was explained comprehensively in Table 1. It seems that everyone had 'some' work to do.

C2.4: Issue tracker is used appropriately to assign new features and bug fixes (1 p)

No issues reported! However, as stated in the project documentation, Telegram was used to 'discuss urgent matters'.

C2.5: Testing and quality assurance is appropriately done and documented. There should be a systematic method to ensure functionality (unit tests, valgrind for memory safety, separate test software and/or something else.) (1 p)

Testing was explained in the project documentation. As mentioned by the group, every part of the code pushed to git was first checked that it actually works. The group had also tested the program functionality with simple printouts. Furthermore, valgrind was used to find possible memory leaks. More information about the testing and outcomes could have been provided, though.

C3.1: C++ containers are used appropriately (including appropriate use of iterators), and justified (e.g., why certain type of container over another) (2 p)

C++ containers (e.g. vectors, pairs) were used appropriately. However, we did not find any justification why certain containers were used.

C3.2: Smart pointers are used in memory management, describe how (1 p)

Smart pointers are used. For example, cars are saved in shared pointers in order to use them efficiently across methods.

C3.3: C++ exception handling is used appropriately, describe how (1 p)

Exception handling is very rarely made. Countless of methods have been used in one try clause. Throw is mainly used for file reading functions.

C3.4: Rule of three / rule of five is followed, describe how (1 p)

Copy assignment operator, copy constructor nor destructor were defined.

C3.5: Dynamic binding and virtual classes/functions are used, describe how (1 p)

We were not able to find any virtual classes/functions. Dynamic binding (static_cast methods) was used in level.cpp and car.cpp.

Other comments and feedback to the evaluated project group.

The game was built according to the initial requirements set and the project plan supported well the actual project outcome. As an improvement idea, the code written could be more readable i.e. try to avoid massive while loops combined with recursive if/else conditional statements. Furthermore, in order to improve the readability, you could try to divide the code into different parts (e.g. functions.cpp). In overall, the game is nice.

If you did this review together with (some of) your group members, list the names of the group members here. Everyone needs to turn in a review, either separately or as a group.

Helena, Tuomas, Vertti