Respondent: **Vili Ketonen** Submitted on: Monday, 4 December 2017, 9:49 PM

Mid-term review of C++ project

Name of the project group evaluated

laser-squad-1

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)

The project seems to be divided well for the group members. Project plan has extra features such as A* path finding and the base project seems to cover many areas of the original Laser Squad game. There should be enough

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)

There are lot of files in the src folder so the structure is not that clear. The group should consider grouping the class files and moving the resource files to separate folders. The overall class structure looks good and information is hidden in the classes to private members. Maybe you should consider fixing the Riffle class name.

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)

the appropriateness of libraries and their use. (2 p)

Group is using external SFML library for graphics rendering which is good for this kind of small 2D game project with simple graphics and simple user interactions.

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

There should not be multiple commits with same messages. Many of these could be grouped to one commit. It seems weird that Class hpp is committed separately from Class cpp. Consider combining these to one commit. On the other hand, there are some commits with a lot of changes. These could be split into smaller commits.

Commit history does not really tell what has been done in the individual commit.

The commit history overall is quite ambiguous.

You should consider these things above in your future commits.

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)

It was easy to build the project using the CMake file. However, in Visual Studio generated file, the missing resources were quite hard to find find from the src folder because there is so many items. Consider putting the resources in their own folder.

Consider also testing in Debug mode since it shows some issues that Release mode does not show up (you have some code using uninitialized pointers).

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)

According to the Git history the work seems to be distributed between the group members - everyone has made commits - some more than others but that does not necessarily mean that the work is unbalanced because the changes between different commits vary.

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

Sadly there does not seem to be anything in the very handy Issue tracker.

Update as of 4 December:

There are 3 issues listed on the tracker. Good that you have started using the issue tracker. Just list all the issues there.

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)

There is a test file with few unit tests. Maybe consider using asserts to easily detect problems. Now the test just prints to standard output and it might be hard to see the problems fast.

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)

Maybe you should consider storing map coordinates to something else than std::pair<int,int> as it does not feel very natural to handle coordinates.

You use vector container in many places, this is good in places when you need fast random access.

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

You have many raw pointers to smart pointers after the last time we checked the project on the version control.

There are still some raw pointers left for example in Game class that you might consider replacing with smart pointers to avoid memory issues.

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

You are not still using any C++ exception handling. Consider adding these to catch exception for example when loading your level.

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

None of the classes seem to use all the elements rule of three (destructor, copy constructor and copy assignment). Some classes have destructors. Consider adding these to the classes that need destructors (free dynamically allocated memory).

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

All the different items inherit from Item class with seems good since they share common properties like weight and name.

Soldier, Scout and Boss classes inheriting from Character seem to differ only by one function. These functions look very similar and only one property seems to change. You might want to reconsider dividing this to so many subclasses.

Other comments and feedback to the evaluated project group.

Keep up doing the work and consider the feedback we have given here and you will do well in this project.

Fix the weird crashes we encountered when testing (vector iterators pointing to invalid locations) and ensure the stability by doing unit tests for special cases and testing in different environments.

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.

Group members that did this review:

Ketonen Vili

Soitinaho Jaakko

Koivusalo Matias

Mustonen Miika

Räsänen Tommi