COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Shray Patel + Siddh Patel

Team Members Evaluated Ishita + Faiqa

Provide your genuine and engineeringly verifiable feedback. Ungrounded claims will lead to deductions.

Part I: OOD Quality

1. **[6 marks]** OOD is about sensible code modularization. Looking at the header files of each object, can you easily interpret the possible behaviors of the objects involved in the program, and how they would interact with each other in the program? Comment on what you have observed, both positive and negative features.

Feedback:

After examining the code, we observed some positive as well as negative aspects to their OOD coding. The code was successfully modularized into different classes each of which handled a specific aspect of the game. They also named all their member functions and class members reasonably, so that by just glancing at the header file and the reading the name of the function we got some idea as to its use. However, some aspects were not ideal.

Some negative aspects we observed included the unorganized nature of their comments. There were random comments that were left in the project, which confused us as to what they were referring to or if it was left in a previous iteration.

2. **[6 marks]** Examine the main logic in the main program loop. Can you easily interpret how the objects interact with each other in the program logic through the code? Comment on what you have observed, both positive and negative features.

Feedback:

After examining the main loop and project.cpp file, we noticed some pros and cons. One pro was that everything in the main project loop was implemented in OOD programming, making it modularized. They also organized the main loop in such a way that their global variables and preprocessor constants were at the top of the main loop, as well as initialized after and then implemented. This made reading the flow of the code easy.

One con that was noticed was that, under the RunLogic loop, there was a piece of code that was commented, as well as debugging code that was commented. This was a bit confusing as the code had no explanation as to what that piece of code did or why it was commented in that loop. This could have been improved by either removing it if it was unnecessary or commenting as to its purpose for leaving it there.

3. **[5 marks]** Quickly summarize in point form the pros and cons of the C++ OOD approach in the project versus the C procedural design approach in PPA3.

Pros

- Able to encapsulate data and behaviour in objects making the code more organized and secured.
- Able to modularize the code making it easier to understand and maintain code.
- Classes and objects can be reused through inheritance.

Cons

- It takes a longer time to write code in OOD rather than procedural, as there are many classes and different files that have to be made.
- If not correctly planned and implemented, then it can be harder to debug the code, as there are many different classes, some of which may be interconnected. This can be avoided in procedural coding, as most of the code written in a procedural format is stand alone.

Part II: Code Quality

1. **[5 marks]** Does the code offer sufficient comments, or deploys sufficient self-documenting coding style, to help you understand the code functionality more efficiently? If any shortcoming is observed, discuss how you would improve it.

Feedback:

The code offers little to no comments or other self-documenting coding style to help understand the code functionality. The code includes only the comments given in the skeleton code. In order to improve upon this, we would recommend commenting all the header files for the newly created classes as well as the respective cpp files as well, so that other developers can understand what each specific class is meant to do.

2. **[4 marks]** Does the code follow good indentation, add sensible white spaces, and deploys newline formatting for better readability? If any shortcoming is observed, discuss how you would improve it.

Feedback:

The code follows good indentation of the code and has sensible white spaces. The code includes many commented-out sections that are not necessary or are ambiguous as they have no supporting comments as to why the commented code is there. We would remove the unnecessary commented code throughout the program and add comments to help the user understand the code better.

Part III: Quick Functional Evaluation

1. **[8 marks]** Does the Snake Game offer smooth, bug-free playing experience? Document any buggy features and use your COMPENG 2SH4 programming knowledge to propose the possible root cause and the potential debugging approaches you'd recommend the other team to deploy. (NOT a debugging report, just a technical user feedback)

Feedback:

Overall, the game works as expected and runs very smoothly. The frame rate that the programmers chose allows for a smooth experience when playing the game. There were just a couple of small things that could be improved on that we noticed pertaining to the UI of the game. Firstly, a lost message is printed when the player self-collides however when the game is forced quit, there is no message. Another observation is that the score is never printed in the lost message. There was also a debugging message for the lose flag being printed while the game was being played, as this should be hidden. These UI changes can easily be fixed by modifying and adding print statements in the main loop. Overall, the game was solid and had no major bugs.

2. **[6 marks]** Does the Snake Game cause memory leak? If yes, provide a digest of the memory profiling report and identify the possible root cause of the memory leakage.

Feedback:

After running Dr.Memory on their project, there was leak that was observed on the memory report. This indicated that the memory that was allocated on the heap in their program is correctly deallocated at the end of runtime.

Part IV: Your Own Collaboration Experience (Ungraded)

1. Tell us about your experience in your first collaborated software development through this project – what was working and what wasn't. If you are a one-person team, tell us what you think may work better if you had a second collaborator working with you.

   We believe that the collaborated software development on this project was a very useful way to go about completing the project. We each brought different skills to the table, which was great, but sometimes we struggled with communication and timing of our git pushes. However, we were able to split up the workload efficiently and still have a cohesive program as we were commenting the code as we were writing it. This helped us read each other's code and understand what each of us did. Overall, we loved the collaboration on this project and thought it was a great idea.