COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Maggie Xie Sungkyun Cho

Team Members Evaluated Ben Fabella Justin Feener

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 behaviours 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.

   Upon reviewing the header files of each object, the headers provide a clear and comprehensive view of the possible behaviors of each object and hot they interact with one another in the program. For example, the header file for game mechanics encapsulates all the relevant methods and variables related to the game, such as retrieving the board size, incrementing the score, and managing the exit flag. Having different headers for the player, food, and game mechs enhances readability and ensures that each object's responsibilities are well-defined.

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.

   The main logic in the main program loop provides clear and easily interpretable interactions between objects. An example can be found in project.cpp, where the statement myPlayer->updatePlayerDir() is used. This line concisely communicates that the player's direction is being updated, showing a straightforward and easily understandable flow of object interactions.

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 of using C++ OOD allows us to model different components of the game such as game mechanics, player, and food into their own classes which enhances the organization of the code and its maintainability.**
- **Cons of using C++ OOD is that there is increased memory usage and**
- **Pros of using the C procedural design approach is easier to follow**

- **Con of using C procedural design approach is that it is more likely to have code duplication which makes the program harder to maintain and increases the likelihood of errors**

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.

   The code is well-commented and provides explanations of its functionality throughout the program. Comments make the code easier for a developer to understand the logic behind different decisions and the purpose of different functions.

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.

   The code follows good indentation, adds sensible white spaces, and deploys newline formatting for better readability. In the project.cpp file, each function is appropriately separated spaces, contributing to a clear visual distinction. Additionally, spaces are thoughtfully inserted between for loops and if statements, making the code more organized and comprehensible code structure. Furthermore, in the for loops and if statements, the code is consistently and adequately indented.

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)

   The Snake Game offers a relatively smooth, bug-free experience without any noticeable problems. However, the bottom five rows of the snake game flicker constantly, making the gameplay look worse than it could be. A possible root cause for this could be that they are constantly redrawing everything, causing inefficient rendering, making the bottom few rows flicker. Some debugging approaches they could take is to try only printing the bottom half of the screen that was flickering and to see if it continues the behaviour. If it does not, the problem may lie in the loop efficiency.

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.

There is no memory leak in the Snake Game, as they deallocate all their “new” variables after allocating them. The variables that are allocated; foodBucket, aList, playerPosList, myFood, myPlayer, and myGM are all deallocated after they are used.

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.

In our first collaborative software development project, the experience was notably positive. Effective communication and a shared commitment to our goals were key factors that contributed to the project's success. Regular meetups allowed us to exchange ideas, addressing problems, and maintaining alignment on project milestones. Code reviews and continuous integration processes further enhances the overall quality of the program. One thing that was hard to work around was trying to work together on a certain portion of the code when we were not together in person. This was difficult as we could not see what the other person was doing, leading to confusion and inefficiency.