COMPENG 2SH4 Project – Peer Evaluation

Your Team Members	Carson Henderson, Noah Iwasaki
Team Members Evaluated	Mohammed Sabri, Naram Hirmiz

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.

There is one new object, "food", which has three public methods. The methods seem reasonable and straightforward. It is very to our team's implementation of a food object. It is easy to see how the various objects would interact.

The addition of getInput and clearInput methods for the GameMechs object follows the code modularization philosophy and is appropriately implemented.

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.

It is relatively easy to interpret how the objects interact with each other in the program logic. It is especially easy to tell for the Initialize, GetInput, and RunLogic blocks. DMA is used appropriately to reduce memory usage.

The DrawScreen method is not complicated but is very nested and the number of variables that are redeclared each block is quite large. The program flow could most likely be simplified.

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.

The benefits of the OOD approach is that the main project.cpp file is far less lengthy and verbose than the equivalent project.c file written in a procedural approach. The overall length is similar considering that much of the logic is offloaded to additional C++ files containing the methods of objects. The OOD approach protects sensitive variables that were exposed in PPA3, such as the input and state of the player. However, in OOD the sheer number of methods, fields, and object identifiers can confuse readers of the code.

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 does not contain many comments, but despite this it was straightforward to understand the program's logic. Better comments could help with easier code readability. For instance, on Project.cpp line 87, it is commented "Draw the top and bottom borders". This is not how the program draws the screen, as it instead checks (given that a part of the player was not printed) if the coordinates lie on the border.

Better comments in the DrawScreen function could help with readability, considering there are nested loops. The comments on the object .h files are adequate and provide a reasonable description of each method and field.

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 generally follows formatting practices. The formatting around the DrawScreen control flow could be improved by using consistent indentation and more spacing to visually separate the flow of logic.

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 game runs smoothly and functionally works perfectly. Visually it is very jumpy and the flickering between frames is very noticeable, especially with the text below the playable area. I didn't encounter food spawning on my snake or other gameplay glitches.

The game over text flashes on the bottom of the screen. I'd analyze the code before the draw screen logic clears the board, because it seems like its printing right before the screen clears. There are also a few debugging lines left, such as the position of some of the food on the board. Thats an easy fix with a couple comments though.

The "quit" button being slash is not displayed on screen. It should be. There is no acknowledgement of the player choosing to quit the game.

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.

Despite the large number of false positives given by drmemory, the memory profiler did not find any memory leaks in their implementation.

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.

Carson:

I learned that debugging code you didn't write is really really hard. Half my time spent on this project was debugging the code to generate a random piece of food, and it turned out to be one line of code missing. I liked the provided test suite for the objPosArrayList class, however I wish it was made clear that you should fully implement the entire class before attempting the test cases, as they rely on multiple functions that aren't initially obvious. I wouldn't have minded more of these test suites.

I don't love this style of OOD. A lot of the code you end up needing feels redundant, even though the point of this style of programming is to lower the amount of redundant code. Also Github was being super dumb. I'm still not 100% sure how it works, I think a more formal introduction to it would have been nice at the beginning of the course.

Noah:

The project was challenging due to the great number of steps, as well as the forced OOD paradigm. I disliked the number of "dummy" or temporary variables that we had to use just to replicate similar logic from a procedural style. Having to create so many new methods etc. was not very enjoyable either. There were many issues with git push and git pull commands, so a class going over how git and version control in general works would probably do the class wonders. Also the C/C++ extension in VS Code would continually flash on the corner of the screen and was very distracting.