COMPENG 2SH4 Project – Peer Evaluation

Your Team Members ____Aurora Boone___Maeve Wheaton___

Team Members Evaluated _____Erion_____Robert_____

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.

The header files are mostly easy to interpret, however there is a lack of comments to explain unclear behaviours. Some comments are mis-placed and not relevant, especially when looking at the player header file. The code itself does appear to be edited well and does not contain unnecessary declarations. The header files are indented and organized in a logical way. Function names are logical and self-descriptive.

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.

Yes, the main logic in the program is easy to understand. It does not appear to contain unnecessary declarations and is organized efficiently and concisely. Much of the main function calls to other areas of the program but the names of functions and variables make it easy to understand. More comments in the main loop of the code would make it even easier to understand, as there was little to no comments in this section. Some code seemed redundant and unnecessary, such as the "if (drawn)" statement which didn't seem to impact the logic of the program in any way.

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	Cons
- OOD in C++ is cleaner and easier to read. Code is "packaged" more efficiently. - OOD has more capacity to handle larger projects. Procedural programming is not able to effectively handle larger and more complex projects. - OOD is easier to work on teams as code can be worked on and tested in parallel.	- OOD requires more background knowledge and practice to write. It's not as simple as procedural programming. - OOD programming can be more complicated to debug. - Memory leakage is more difficult to resolve in OOD programming.

Part II: Code Quality

1. **[5 marks]** Does the code offer sufficient comments or deploys sufficiently. -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 applies somewhat sufficient comments but does not specifically comment on more complex functions. More description, especially in the player object would make the code a lot easier to read. Comments also are out of place at times and do not seem relevant to the file they are written in. Developer code was also included in the final deliverable, which was unnecessary and made the program more difficult to read.

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.

Indentation and organization were done well. The code was visually appealing and fairly well spaced. Code was not cramped, and whitespace was used effectively. There were only a few spots in the code that could have been a little more spaced out to clearly show each step of the process.

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 largely works correctly, however, the designers failed to correctly check if new food was generated under the snake. This means that while playing, food may be generated under the snake and is not viewable until the snake moves off that spot. In addition, the designers left the debugging report printing on the screen which contributes to visual clutter and could have been removed/cleaned up.

It appears the designers attempted to prevent food from being generated beneath the snake, but the check does not work as intended. Through a visual analysis, it appears the continue statements aren't mirroring the pseudo-code's expected behaviour. This leads to the snake body not being checked when new food is generated. The designers could consider using GNU de-bugger on the loop to validate the co-ordinates of the food generation. They could also consider implementing a flag value instead of using continue and break statements.

To contribute to a more user-friendly interface, the designers could have removed the de-bugging messages in the terminal. The de-bugging messages lead to more visual clutter and do not provide any necessary information to the user. If the designers wanted to display the coordinates of the snake, they could have used better labels and white space to make it more understandable. They also did not include information on how to exit the program.

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.

A Dr. Memory report showed no memory leakage from the Snake Game.

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.

Our experience in our first collaborative design project was quite positive. The design iteration steps provided on the manual helped us divide up the tasks and work effectively in parallel. We somewhat underestimated the amount of time iteration 3 would take, leading to a rush near the end. It was also difficult to fairly divide up the work for iteration 3, as we had merging issues with GitHub and could only work on one computer at a time. We had to physically meet in person to work on those steps which made that iteration even more rushed.

In the future we would allocate more time to final iteration steps so we wouldn't be so rushed near the end and would be able to "tag-off" when working on final steps.