COMPENG 2SH4 Project – Peer Evaluation

Your Team Members	Nina Samson, David Segal-Pillemer
Team Members Evaluated	Emilie and Mustafa

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.

Looking at their header files we could easily interpret what each object class does and how they interact with each other. For example, in the Player.h file the Player constructor takes in a GameMechs and Food pointer as parameters. This indicates that the Player class uses the game mechanics and food classes. On the other hand, in Food and GameMechs there are no parameters in their constructors showing us they do not use Player or any other class. Additionally, the function names are all indicative of what each function does. This allows us to follow the functions quite easily. One negative feature of the header files was that the team did not remove comments provided by the teaching team, which could be confusing to the reader.

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.

After going through the main program loop, we could easily follow and interpret how all the objects interact with each other to produce the snake game. They made use of the classes that they created and took advantage of object-oriented design. For this reason, when reading the code, it is very easy to follow. Another positive about this code is that it is very concise. There does not seem to be any unnecessary function calls. One negative aspect of the main code is in the draw screen routine there are significantly less comments which made it a little difficult to follow the logic from an outside perspective. We were able to understand because of our extensive knowledge of the project, but we're concerned that someone looking at this for the first time may have trouble interpreting what each function is meant to do.

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.

In general, OOD is the preferable method to the design approach in this project. Some pros include the main program being very straightforward and reads like an essay. This allows for someone who has not

written the code to be able to interpret its functionality. Additionally, OOD allows multiple programmers to work on the same project simultaneously since they can each work on different classes. Further the OOD approach allows for reusability of functions and objects. This also allows for easy modifications to the code.

Some cons to this approach include increased confusion about interaction of all classes and understanding how OOD works. There is a steeper learning curve compared to the C design employed in PPA3. Another con is the memory usage in this approach. In OOD we use more memory since we are creating so many more files.

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.

Overall, the code includes sufficient comments and is very easy to follow and understand. Within each cpp file each function has adequate explanation which allows us to understand very clearly what each function is meant to do. Something we noticed that went above and beyond is that they included errors being thrown within objPosArrayList class if the memory would go out of bounds, blocking a potential segmentation error. This makes their code less prone to bugs. No shortcomings were observed.

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.

Overall, this code follows good indentation, has sensible whitespaces, and is formatted for optimal readability. One example of this is within the food constructor their variables are all grouped together in a logical sequence. This is followed by multiple blocks of code, where within for loops or if statements, all code related to each other are grouped together and separated by whitespaces. This allows us to easily understand each block of code having a distinct function. One shortcoming could be observed in the header files where some confusing comments were not removed, and more relevant comments could have been added.

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)

This snake game runs smoothly and does not have any buggy activity. When food is collected, immediately the length of the snake is altered, score is incremented, and new random food is generated. There are also 2 values of food on the board, one worth 10 points and one worth 1 point. There are clear

instructions underneath the game board outlining the rules of the game. Once the player reaches 50 points the game will end and display a winning message. If the player presses the esc key or collides with itself, the game will end and display the appropriate game ending message. Overall, this was a phenomenal code and pleasant game experience.

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.

The snake game runs without any memory leakage (as seen in the image below). This is because they used new and delete keywords properly in the code when using memory on the heap.

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.

Overall, we enjoyed working collaboratively on this project. At first there was a learning curve which was frustrating, but we now see the value of OOD.