COMPENG 2SH4 Project – Peer Evaluation

Your Team Members	Abaan Khan	Someshwar Ganesan
Team Members Evaluated	Brooke C	Samantha S

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**.

   POSITIVE – The header files are neatly organized. They give us a good hint on the possible behaviour of the objects involved as they are supported by proper commenting too. Moreover, the Food class is created in a different header file rather than integrating it into Gamemechs.h which helps in easily interpreting the functions of different objects.

   NEGATIVE- The commenting in Player.h could have been a bit more informative.Comments mention upgrading certain parts in iteration 3, but there's no context or details about what these upgrades entail.

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**.
   POSITIVE- The Initialize function initializes the game-related objects and sets up the initial game state perfectly.

   NEGATIVE- The GetInput() function is declared but not implemented. This leaves a gap in the understanding of how user input is being handled in the game.While the code has some comments, there's still a lack of comments explaining the purpose of certain sections such as the DrawScreen Routine, making it slightly challenging for someone new to the codebase to understand its intricacies.

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:

1) Improved Organization and Implementation
2) Enhanced Code Maintenance and Debugging.
3) Reduction of Redundancy through Inheritance.
4) Allows developers to model the software based on real-world entities

CONS:

1) Increased Memory Usage.
2) Certain simpler problems may be more easily solved using a procedural approach.
3) We need to be able to create and handle multiple files together to properly organize each class.
4) Managing dependencies between classes and objects can become challenging. Changes in one part of the code may have cascading effects on dependent classes

.

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.**

   After reviewing their code, we concluded that all attempts made to implement self-documentation in the code is clear. They used naming conventions which we are all familiar with and this is consistent throughout the code in all files. This made the peer-review process easy for us, as we were immediately able to understand the purpose of every variable and function.

   In terms of comments used in the code, there were not enough comments, if there were any at all, but this was not a hurdle for our review as the variable names and function names used were enough to help us understand their work. A point to note however, is that they used comments to help themselves during the project development, and did not remove them after completion.

   If we were insisted to provide any feedback to improve the readability of code, we would suggest adding comments on the logically heavy functions such as the generateFood() function, and their checkFoodConsumption() function, as most other functions are self-explanatory. The comments would have to be clear and descriptive. This would be very helpful for someone who is new to the concept of Object-Oriented Design or Programming in general.

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.**

   After looking at and carefully reviewing their code, it is sufficiently readable as they used proper indentation and white spaces in between functions which makes the code blocks visually appealing and improves readability up to a sufficient extent. Moreover, they used a new line for their UI.

   However, there are few minor changes which can be made the readability of code. Firstly, a lot of their individual blocks of code such as for loops in their DrawScreen() function, and checkFoodConsumption() have no whitespace in between prominent sections of the function, making it hard to understand the logic in a clear manner.

   In conclusion, the code has appropriate white spaces and indentation, but could use some whitespaces within certain functions. The improvements suggested above would make the code look substantially neater.

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 code offers a smooth working snake game and is bug-free. The appropriate Game ending messages are displayed properly upon the snake crashing.

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.**

   After running Dr.Memory, the report indicates that there are no memory leaks in any of the files, which means the code correctly and effectively allocates and deallocates memory wherever required.

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.**

   It was comforting to have someone to divide the work with, given the relatively small time period when compared to the work load. However, it is definitely possible to have completed the project as one developer, if there was a later deadline.

   However, team members might have different coding styles. Ambiguities or misunderstandings can occur, leading to wasted effort or misaligned goals. Regular check-ins and openness can help mitigate these issues. Moreover, merging code changes from multiple contributors can be challenging.

   However, we feel great that we were given the opportunity to collaborate. Even better that it was not a group project, since that would have been a logistical nightmare to go from solo development to group collaboration at once.