COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Annie Phan and Roxxannia Wang

Team Members Evaluated Zhihao Zhou and Xuan Chen

Provide your genuine and engineeringly verifiable feedback. Ungrounded claims will lead to deductions.

Part I: OOD Quality

1. **[5 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.

Pros: When looking at each of the header files, each class such as Food, GameMechs, objPos and objposArrayList show well defined responsibilities and provides a clear interface for interaction. For example Food class is responsible for handling food-related functionalities, the GameMechs class manages game mechanics and Player class encapsulates player-specific behaviour. These all contribute to well-maintained and modular code while additionally enhancing code organization and readability.

Cons: There are some areas that could be improved for better clarity. For example, the Food class takes an objPosArrayList as a parameter in the generateFood method. However, the header files did not explicitly indicate that objPosArrayList was used in the food class. The high interdependence between classes can sometimes increase difficulty making modification in the original class without affecting the linked functions in other classes, and maybe over complicate the code design.

2. **[5 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.

Pros: The main program loop shows several positive aspects for game logic handling and object interactions. When looking at the initializing, it appropriately sets up game objects, making sure that myPlayer is correctly linked with reference to myGameMechs and myFood. Additionally, the object interactions are clear as seen by the invoking methods on myPlayer and myFood based on the players actions and positions when playing. The scoring and food handling mechanisms are easily interpreted with distinct cases for regular food, good food and bad food, with each case affecting the player score and snake length.

Cons: There are a few areas that could be improved such as variable names and redundancy. Some variable names lack descriptiveness, affecting the codes readability in some areas. For example, foodType could be more descriptive and named more explicitly to enhance understanding such as goodFood, badFood, and regularFood. Additionally, while there are comments, some lack detail and

could be more informative especially in the complex logic sections. There are also some instances of redundancy such as exitFlag and loseFlag, which could be streamlined for more clarity and numbers could be replaced with named constants for improved code maintainability.

3. **[3 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.

Positive: The utilization of C++ OOD approach allows the main logic of the main program much cleaner. Different classes are allowed to work with each other and use the pre-established functions from other classes. In PPA3, all the functions are positioned in the main file, decreasing the readability and increasing the redundancy of certain functions. Additionally, C++ OOD approach allows private classes, prevents the users to change crucial variables in the game whereas in C procedural design, the variables are not as well protected.

Cons: C++ OOD can be complicated at times when the class is not well designed especially when the classes are interacting with each other. C procedural design offers a more straight-forward and simple design process when building a first-time project.

Part II: Code Quality

1. **[4 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 shows all the necessary comment to understand the brief functionality of each code block. No excessive commenting was spotted. However, the implementor did seem to be confused by the "loseflag" in gameMechanics() (with //???? as the comment) as it should be used as a part of the self-consumption termination process.

2. **[3 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 food indentation. Sensible white spaces were added when necessary. Newline formatting for better readability can be seen in the game playthrough, showing detailed instruction and explanation on different food generated on the board.

Part III: Quick Functional Evaluation

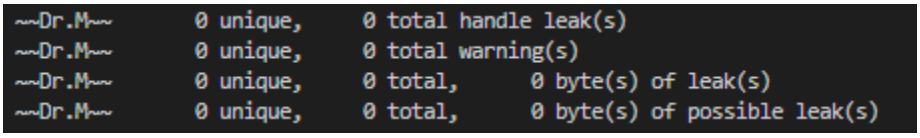
1. **[6 marks]** Does the Snake Game offer smooth, bug-free playing experience? Document any buggy features and use your COMPENG 2SH4 programming knowledge to propose the possible root cause and the potential debugging approaches you'd recommend the other team to deploy. (NOT a debugging report, just a technical user feedback)

The Snake Game offers smooth, bug-free playing experience. Moreover, the implementors of the project has shown great understanding of the project, not only fully completing the project without issues, but also including 3 unique food-snake behaviours. Snake length increment and score increment is both true to the description. Snake movement is smooth, and the wrap-around logic is correct.

No buggy features were spotted throughout the few playing sessions. Game-ending functions also work as expected. However, there was no differentiating ending between normal user termination and self-consumption. There should be a different game ending to set apart the two game endings.

2. **[4 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 Gama did not have any memory leakage as shown in the screenshot of drmemory report below.



Deletion of the heap functions can be easily located in different classes.

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.

The experience in this first collaborated software development project was quite enjoyable as each person was assigned to a specific part to work on instead of scrambling throuhg the whole project by oneself. Having a partner also allows timely support when running into critical problem or asking for quick feedback. The level engineering process provided in the project manual was very helpful and the overall experience is great throughout the project.