COMPENG 2SH4 Project – Peer Evaluation

Your Team Members	Simone Tabile	Asheel Dowd
Team Members Evaluated	Michael Mota	Riley Chai

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.

   After reviewing the header files of each object, one can easily interpret the possible behaviours of the objects involved in the program, and how they interact with each other in the program. For example, prior to iteration three, within the *player* header file, the primary *playerPos* fell under type objPos. It was easy to determine what the purpose of the playerPos variable, even from just the header file. Also, it becomes even more self-explanatory when the variable is switched to a pointer called *playerPosList* of type *ObjPosArrayList*. One can consider this a positive feature as it's essentially transcribed from the UML diagram, however, the sheer number of classes and variable types can be overwhelming. When reviewing Michael and Riley's header files, their variables are named appropriately and descriptively. However, for the sake of consistency, the public and private classes are flipped in some files, so for feature reference, it should have the same convention for all.

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 examining the main logic in the main program loop, one can easily interpret how the objects interact with each other in the program logic through the code, especially regarding the game mechanics and player objects. The object names are descriptive, and when they're initialized, the arguments passed aid in understanding what the objects do. For example, Michael and Ryan created a separate food class, and initialized everything accordingly, as well as their temporary variables (tempBody and tempFood).

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.

   When using C++, you can use encapsulation and modularization, which is structured and organized. In turn, this makes debugging and coding a lot easier, however it will take more time. One thing that makes C++ OOD good to use is constructors and destructors, as the creation of one inevitably leads to the creation of the other. Conversely, C's procedural design is faster to get started with as there's no prototyping involved, and all your code is consolidated in one source file. Consequently, it's easy to lose track of your variables, which can lead to disorganization.

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.

   This code provides a sufficient number of comments where it is neither too little nor excessive. These comments appropriately describe and explains the functions of the code using the minimum number of words required. Specific comments such as one found under the generate food function in the food file, highlights the use of the passing by reference method which further improves readability. Despite using a different approach for generating food items than our own, the different approach is easy and simple to understand due to these well laid out comments. No shortcomings were observed and therefore no improvement is found to be necessary.

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.

   This code follows good indentation, spacing and newline formatting. A great demonstration of good spacing can be found under the player data member in the player file. Here, lines of code that are relevant to each other are grouped together, and those that do not relate such as the initial player position and food reference, are separated by a new line. Loops and if statements are also nicely indented and organized. Though not always necessary, the curly brackets consistently included with the if statements help in visually seeing and understanding the separation between if statements. No shortcoming is observed and therefore our team does not see any need for improvement in terms of formatting.

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 Snake Game offers a compliable and a smooth bug-free player experience. The static border is printed with no issues in its alignment. The wrap around feature works without the player going inside the border before wrapping around. The player also does not overlap the border either. No food is generated on top of the snake body nor the border. Also with food generation, the expected number of normal food and bonus food is displayed with every collision. The snake body movement/direction following the inputs from the wasd keys is smooth and as expected. The score correctly increases corresponding to what food item is consumed. The suicide function also works efficiently and immediately sets off the exit flag and message. Overall, no issues are experienced after a thorough understanding of the expecting results and its in-depth comparison to its actual results.

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.

   Michael and Riley have 0 bytes of leak, which can be attributed to their use of constructors and destructors.

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 first collaborative software development experience was great and successful. This project allowed us to explore and improve our collaborative and communication skills in a way that is different from other courses such as the 1P13 course. This experience emphasized the importance of having an in-depth understanding of the goal of the coding project and the approach. Specifically, it demonstrated the need to outline what steps of the approach are dependent on other steps, and which ones can be done independently. This was important for time management and efficiently separating tasks. Working in different sections separately to build the code proved to be efficient and simple. On the other hand, working simultaneously together in iteration 3 was doable but difficult because either partner needs to push their updates instead of the code updating immediately. This made it a little more time consuming and difficult to equally divide up the work.