COMPENG 2SH4 Project – Peer Evaluation

Your Team Members	Vivian Nguyen	Xinyi Wang
Team Members Evaluated	Owen Martin	Anushka Goswami

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.**

    In terms of header files, it is generally quite easy to interpret what each object is meant for. The names chosen for each method in each class tell us about its purpose in a clear and concise manner. As a result, we can easily figure out each object's primary purpose. For example, it is clear that the player class is meant for keeping track and updating the program based on user input and food-player interaction due to methods like "trimPlayer", "checkSelfCollision", and "checkStarvation". Another good example of this is the objPosArrayList class, that has methods titled "getTailElement", "insertHead", and "removeTail". From the names of these methods, we can deduce that this class is primarily used to aid in drawing the snake.

    There are few, if any, negative qualities in this code in terms of OOD. Everything is organized well—there are no confusing or messy structures. There is no evidence of poor naming, making it confusing to understand what each object is meant for. However, one could argue some methods like trimPlayer are a bit redundant, since it takes an existing method and puts it inside a for loop. This defeats the purpose of using OOD for modularity. However, without the trimPlayer method the code in the main program would be less succinct, which is also a goal of OOD.

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.**

    The main project code can almost be read like a novel. It is very clear how the program works, and it is particularly demonstrated in the Run Logic function in the main program loop. For example, the lines:

    ```
    collected_symbol = snek->getFoodCollision();
    trimLength = gameData->processFood(collected_symbol);
    ```

    illustrates that if the player (snake) collides with food, it will collect the symbol. It will then use the symbol to see how much of the snake to trim. It would not be easy to understand what is occurring if we purely had what is written in each of these methods in the main program.

A minor negative feature of this code is that it is unclear that the object "snek" is the player, since (1) it is spelled wrong, and (2) since we are not calling it some variation of "player" someone without any context may not think that the snake can be controlled by the player.

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.**

	Project C++ OOD	**PPA 3 C Procedural Design**
PROS	• Main project code is very clear and easy to understand; reads like a novel • Easier to debug than PPA3 since we know which class and method the issue is from • Easier to test; can just test part of the code without impacting other parts • Easier to reuse parts of code—modular • No global variables needed! - less of a security concern	• For simple small-scale programs, it can be easier to use • Less difficult to set up, since everything is in one main program • Since everything is in one main program, it is easy to figure out how to access each variable for beginners like us
CONS	• Can be difficult to set up. Many classes, and methods, and for beginners like us it was hard to figure out how to access them in each class. ○ Not straightforward	• The purpose of each part of the code is not immediately clear • Harder to debug; don't know the source of the problem • Harder to test a small section of the code • You may need to duplicate parts of code, that could have been put in a method/class in C++ OOD • Many global variables :(

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.**

In general, we thought there could have been more comments to help us understand the code. Items that were self-explanatory like getter methods had less comments (which was good), however there was some code that was a bit more complex, or new to the program, that should have been commented on.

For example, the for-loop in the checkSelfCollision method in Player class could have been explained with a single comment like "//checks if each index in the snake class (except the head element) overlaps with the head element".

Another example is the new processFood method in the game mechanics class. The name of the method is quite general, so comments about its purpose, or what it does would help (ex. //uses the character passed in to determine how much of the snake to trim).

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.**
   The code follows a consistent indentation style. Whitespaces between operators and characters are also used reasonably and follow a consistent style. Newlines are also used to block off logical sections of code. Overall, the code has good indentation, sensible whitespaces, and good readability.

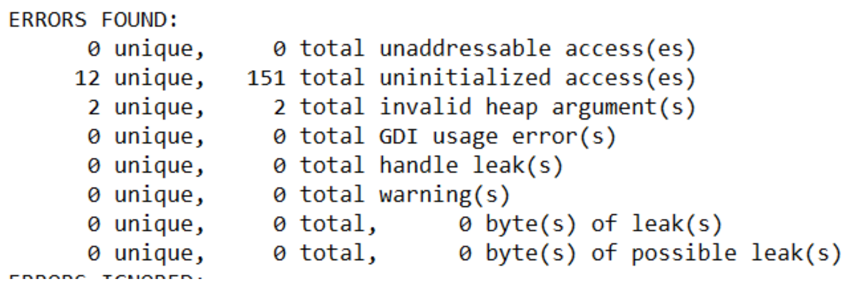
Part III: Quick Functional Evaluation

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)

   Overall, the Snake Game is very smooth. We encountered no bugs when running the program. The program had relatively smooth animation, the snake moved in correct directions , and the snake adhered to the wrap-around. We tested each kind of food: the typical * food (increases snake and score), the ? food (decreases snake and score), and the $ food (decreases snake, increases score). Each one worked correctly. We tested if there was a difference between exiting the game and losing the game. We also tested all the lose conditions. In conclusion, there were no bugs, the game ran smoothly, and everything worked as intended. The game even stated the controls for the snake.

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.

There are no memory leaks in the Snake Game!



```
ERRORS FOUND:
      0 unique,     0 total unaddressable access(es)
     12 unique,   151 total uninitialized access(es)
      2 unique,     2 total invalid heap argument(s)
      0 unique,     0 total GDI usage error(s)
      0 unique,     0 total handle leak(s)
      0 unique,     0 total warning(s)
      0 unique,     0 total,      0 byte(s) of leak(s)
      0 unique,     0 total,      0 byte(s) of possible leak(s)
```

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 a good experience to work with another member on the project; we were able to fill each other in on gaps in our knowledge and work simultaneously with one another to save time. It was only when we reached iteration three that we began to struggle, since we were not sure if we would save over each other's files. We had to let the other person know when we would be working on the project/coordinate times to prevent this issue.