COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Viraj Bane & Tai Li

Team Members Evaluated Dev Patel & Rajveer Sandhu

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.

We were able to deduce functionality of all objects in the program thanks to a combination of comments, and proper naming. They also properly used helper functions in their Player object to help increase clarity in the code and reduce copy paste code. Alongside that, it is evident that this team put significant effort increasing efficiency of the code, even in the PPA code that would be functional even if copy pasted over from the previous assignment

One small error we noticed was that two of the helper functions created were improperly labelled as public functions, when their only use was inside an object class file, while this is a minor issue that doesn't impact functionality in a program of this size, it could quickly get out of hand in a more complex project.

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.

Something we noticed in the readability comes from the design choice of *where* code is located. For example, the print statements which output the game ending messages are in the draw screen method, locked behind an if statement, as opposed to in the CleanUp function. Another example is that all the input processing takes place inside of the player class as opposed to the RunLogic function even if the processing does not directly relate to the player class. Similarly to part 1, this is a minor issue that does not have a negative impact on this assignment, however in a larger program it could pose an issue.

However, once we were able to locate the key functions, interpreting the code was straight forward, this is once again thanks to comments and helper functions improving the clarity.

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:

- OOD allows for improved clarity and larger scale projects without namespace contamination.
- C++ also has significantly improved OOD support compared to C, which while OOD is possible it isn't the intended use case.
- Improvements in QOL features make CPP code much easier to read and debug compared to CPP

Cons:

- While CPP can support larger programs with much more ease. C excels at small scale programs where memory management is of the utmost importance.
- The lack of directly implemented features such as OOD and Exception handling makes CPP generally slower compared to C.
- While OOD can enable a progammer to do more, it is a double-edged sword as poorly planned OOD can hold back program efficiency.

<u>Part II: Code Quality</u>

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.

We believe that there are enough comments to offer sufficient understanding of the code, although we also noticed that there could have been more comments on certain parts of the code to offer an easier read to someone who was new to the code. We think that there are enough comments added in the code, and the comments are easy to understand and straightforward.

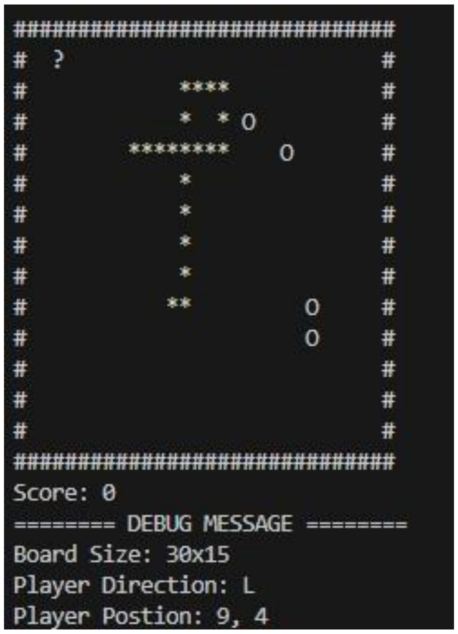
2. **[4 marks]** Does the code follow good indentation, add sensible white spaces, and deploys newline formatting for better readability? If any shortcoming is observed, discuss how you would improve it.

We believe that the indentations were well done, but the white space that was added was sometimes excessive. For the most part, the white spacing is sensible and makes the code easy to read, but sometimes there is too much white space. This can be changed by removing some of the white space and removing the curly braces for one line if/else statements.

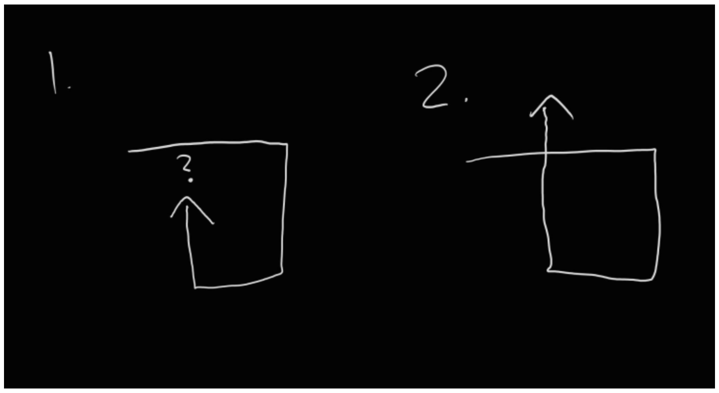
Part III: Quick Functional Evaluation

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)

In 99% of scenarios the Snake Game offers a smooth bug-free experience, however, there was one bug we had found that took us a lot of time to reproduce that would be considered a "critical bug". The special food '?' which was implemented has 3 random possibilities when eaten. One of these possibilities increases the snake length by 4. The bug we had found was that if the snake ate a '?' while within 2-4 spaces from the body, the code would insert 4 snake nodes into the head of the ArrayList and cause the snake to warp through its own body. We have included a screenshot of the bug in action, and a diagram to better explain what is happening



```
##############################
#  ?                         #
#            ****            #
#            *  * O          #
#        ********    O       #
#            *               #
#            *               #
#            *               #
#            *               #
#           **         O     #
#                      O     #
#                            #
#                            #
#                            #
##############################
Score: 0
======== DEBUG MESSAGE ========
Board Size: 30x15
Player Direction: L
Player Postion: 9, 4
```



This is a complicated issue to fix, our first idea was to instead of inserting the additional nodes into the head, to instead insert them into the tail, however, albeit in a fraction of the cases, could still cause this warping behaviour to occur. Due to the elusiveness of the bug, this could be a viable fix as it would result in this behaviour only happening in very few cases. Ultimately, we believe the best case of action would be just to include a check after a mystery food is consumed that ensures no body segments are on top of each other, causing game loss. This is because this bug stems from the fact that we only check head on body collision, not body on body collision.

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 did not result in any type of memory leak as per the DrMemory report.

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.

Working on a collaborated software development is a learning experience, you get to learn how people think differently from you and how you need to compromise those differences to make a code that is easy to read and functional for the other person. This project would not have been possible without GitHub. GitHub allowed us to work on separate machines and separate operating systems, it also allowed us to work remotely and not have to be present in person. An important thing to keep in mind was to add comments for the other partner so we would know what the code meant and would not have to spend time figuring it out for ourselves. Something that was not working well was having one partner on Windows while the other one on MacOS. Changing the makefile was tedious and would frequently cause issues with compiling even when changed correctly, as a result we often found it better for us to just debug on the Windows computer.