COMPENG 2SH4 Project – Peer Evaluation

Your Team Members ____Haolin Du________________ _Jing yuan Deng ____________________

Team Members Evaluated __Aman __________________ __Sameer__________________

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.

Their design follows a correct OOD approach, with clear separation of concerns between UI, player logic, and game mechanics.
Positive features: The name of functions and variables are reasonable and all functions interacting with each others as intended. In their program, functions are located in the correct class.
Negative features: There's no significant negative features in their code.

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 examine their main program,
Positive feature: we found that the game mechanism and the player object are created on heap and these objects can be interact with each others easily inside different functions and get passed on through the whole program.
Negative features: Their clean up function involves a Drawscreen method.

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.

For the C++ approach the OOD approach cuts the codes in different classes, which makes the code simpler to read, but it may cause larger files since not all functions in all classes are used. For the C procedural approach, there is less memory usage since there are less pointers involved and also makes the code shorter. However, the procedural design approach is harder to read.

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 their codes, they added some sufficient comments in each document. No doubt, it sometimes help us to understand the code's working behavior, however, the amount is not enough and detailed to understand the meaning of the each line of their program.
To improve it, we would add more detailed comments in each functions, especially in those large chunk of the code like the eatFood function, movePlayer function, or the drawScreen function in the main program, so that the others can understand easily.

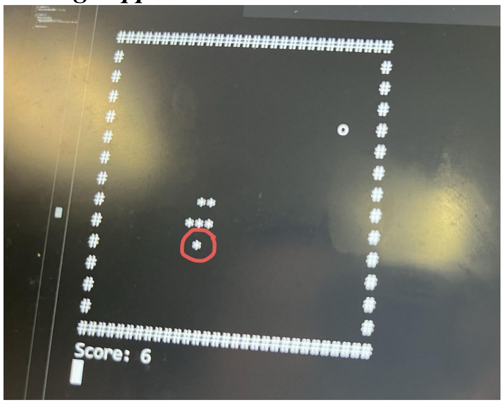
2. **[4 marks]** Does the code follow good indentation, add sensible white spaces, and deploys newline formatting for better readability? If any shortcoming is observed, discuss how you would improve it.

The functions with large chunk of codes have good indentation, but does not have sensible white spaces or newline formatting, which makes all the codes stays together and hard too read. To improve this, we will add newline after variable deceleration and for loops.

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)

Most of the functions of the code behavior correctly, but the detection of eatself function has a small problem where the snake moves one more step after it eat itself and then the game over message appears.



```
Score: 6
```

The problem may exit because that the eat self detection happens inside the move player function after the snake moved. To fix it, they can write the detection step before the move step.

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.

No, we use the normal steps to check the memory leakage(Drmemory.exe), after checking, it shows that there’s 0 byte(s) of memory 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.

   The experience is not bad, and we works our own parts. However, when we debugging the codes, as we did different part and write different codes, that’s a little hard to fix the other one’s bug. But we did it and really excited to solve out the problems.