COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members	devrij12	wauchops
Team Members Evaluated	pendyalj-2005	chaia1

Provide your genuine and engineeringly verifiable feedback. Ungrounded claims will lead to deductions. Completing the peer code evaluation on time will earn your team a total of **25 marks**. Do not exceed 2 paragraphs per question.

Peer Code Review: OOD Quality

1. **[3 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 way the main game objects are initialized and used is very clear. One shortcoming is that the way player interacts with food is hidden within the player class. While sometimes this type of abstraction is good I find it harder to interpret the main game loop. The main loop simply calls two functions from the player class. These functions in turn call more functions from player class and the food class. Some of these functions I believe should be called in the main project file rather than being hidden in the player class.

2. **[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.

   The advantages of the OOD approach are very apparent in the main program class. The main loop is very easy to follow as it simply employs functions from other classes which each take care of a specific part of the game. PPA3 was not like this, the main loop contained many functions all defined within the same file which made the main game loop and file much more difficult to interpret.

Peer Code Review: Code Quality

1. **[3 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.

   There are more than sufficient comments for the general as well as in-depth understanding functionality of their code. This included comments for creating variables and pointers, what for loops were to be iterating through and what each portion of the function definitions were doing. In the cases where there was lack of comments the variable names and function calls were easy to follow such that I had no issue in understanding certain snippets.

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 indentation and spacing of the code overall was well done with there being 1 line between all function definitions and lines between most code logic and the return statements for a given function. There were some instances where they did not include curly brackets for a few of their if-statements as seen in the objPosArrayList.cpp file for the removehead function. For all if-statements I include the curly brackets in case I need to add more logic to the if-statement after initially making it. Additionally, it makes it easier to read for peers especially when sharing code between a partner.

Peer Code Review: Quick Functional Evaluation

1. **[3 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 technical user feedback)

   The game plays very smoothly, no visual or functional bugs. The emojis offer an immersive gaming experience.

2. **[3 marks]** Does the Snake Game cause memory leak? If yes, provide a digest of the memory profiling report and identify the possible root cause(s) of the memory leakage.

   The Snake Game does not cause a memory leakage. All heap objects seem to be managed correctly.

Project Reflection

Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team's implementation in addition to your own, reflect on the following questions:

1. **[3 marks]** Do you think the compound object design of objPos class is sensible? Why or why not?

   I do not think the compound object design of objPos class is sensible since it uses a c struct which is redundant in a c++ class. The members of Pos struct should simply be fields in the objPos class. This would simplify the interface with the class. I think c structs do have applications in c++ however in the context of the project and the objPos class, they are redundant.

2. **[4 marks]** If yes, discuss about an alternative objPos class design that you believe is relatively counterintuitive than the one in this project. If not, explain how you'd improve the object design. You are expected to facilitate the discussion with UML diagram(s).

   An alternative objPos class that would simplify and improve the implementation is one that stores the Pos information as integer fields in the class. The objPos class only holds x and y information, the symbol information is static and has no logic based around it, it is mostly just a visual function. For this reason, it would make sense to have the Pos information (x and y) stored as fields of the class since these are the main functional parts of objPos, there is no need to abstract them.

objPos
+ x: int

+ y: int + symbol : char
+ objPos() + objPos(int xPos, int yPos, char sym) + isPosEqual(&objPos) : bool + getSymbol() const char + getObjPos() : const objPos + setObjPos(int xPos, int yPos, char sym) : void + setObjPos(objPos o) : void + ~objPos() + objPos& operator=(const objPos&)