COMPENG 2SH4 Project – Peer Evaluation [30 Marks]

Your Team Members doylej18 soleh (team name Hannah and Julie)

Team Members Evaluated Nabils team (memonn and bayario)

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 **30 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.

From the main logic, the objects are quite easily discernable. It is evident from the code thaty certain classes control certain aspects of the game. For example, in this line here foodPtr->generateFood(playerPtr->getPlayerPos()); it is quite easy to tell that the player class was used in the food class, specifically the playerPos, likley to ensure the good does not generate where the player is.

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.

Pros:

- Because we utilized an OOD approach, we were able to reuse certain classes and implement them in multiple different classes instead of recoding the same logic in each function. An example of this is ObjPosArrayList, which was implemented in both the Player class and the Food pos class. However, reusability can be seen as a con as well (will expand on this in cons section).
- OOD implementation is also beneficial for the expandability of your code. This is especially useful for game development as there is always more features that could be implemented (scoring, different levels, etc).

Cons:

- Using OOD can be more complex than using a procedural approach, which introduces more room for errors, which can take a lot of time to debug.
- In OOD you can reuse certain parent classes for other classes, but as mentioned in pros this can also be a con as if you are not 100% certain your parent class runs properly, it can be a huge task to debug and trace back the error. If you build large sections of code on a class that does not work properly or has memory leaks this can affect the whole code and can be difficult to trace back.

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.

The code is very clean with sensible comments and easily identifiable variable names. There are a couple components that could be changed to increase readability. For example, in the DrawScreen function in project.cpp, gameMechsPtr->getBoardSizeY() was used inside the for loop to generate the board. While this is completely correct and an acceptable way to generate the board, for code readability I would suggest setting a variable named BoardSizeY = gameMechsPtr->getBoardSizeY() and using that. There are other examples of this in the code as well, where playerPtr->getPlayerPos()->getSize() could be replaced with PlayerSize. Again, the way the code is implemented is correct but for code readability, I would recommend. Overall, while there are a few minor adjustments to be made, the code itself is well commented with easily discernable naming conventions.

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.

Yes, the code for the most part displays sensible whitespaces and newlines. The code itself has good indentation, separating different logic blocks which is good for the readability of the code.

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 snake game offers a good gaming experience, with easy to understand symbols and automatic input. The game speed creates a fast-paced game that creates a good level challenge. However, the game speed causes the game to glitch a great amount.

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.

No there are no memory leaks. There are proper memory deallocations in player (deallocating playerPosList) , and objPos (deallocating pos).

Project Reflection

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

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

The objPos handles the position variables of the positions. This allows each of the higher classes to utilize its set up without changing it. This allows player and food to easily utilize the class separately to ensure organization and readability. It also allows the snake to be easily manipulated as needed. The class adheres to Rule of Six (minimum four). It also has two ways to get positions (getx, gety or getobjPos-> x

getobjPos->y), ensuring flexibility. One improve that could be made is not storing Pos* as a pointer which makes it susceptible to memory leaks.

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.

For a counterintuitive design approach, x and y can be stored in different classes, in global variables, or any other option that does not include encapsulation. Furthermore, the class could be further split up into x and y instead of having everything in one objPos. Or, objPos could turn into only global variables and completely eliminated the classes. Finally, one more design approach could be to simply create global variables in project.cpp, and update and use the variables from there. Note that all these approaches would be much less effective then the well designed objPos class that has been implemented.