COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members: Varun Gande, Mustafa Haider

Team Members Evaluated: Jeremy Shi, Benjamin Semmlerb

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.
   - After examining their main program loop, it was relatively easy and straight forward to understand the different variables and functions. The team added a good number of comments and explanations in the main program loop which helped to understand the relationship between the various objects.

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:**
     1. Due to the structure of OOP, it helps with code reusability, which as a result helps with reducing unnecessary code, and increasing readability.
     2. Compartmentalizing code also makes the program more organized and easier to understand.
     3. Different team members can work on different sections of a program without and then later combine their work.
   - **Cons:**
     1. In some cases, it can make it harder to debug, since a section of a program may depend on other classes or objects in different files.
     2. It can also cause performance issues since the program will consist of more information to compile and have more tasks to perform.

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 team has added various comments throughout the program which made it easy to understand what was happening in the program and how different objects interacted. However, we feel that the Food Class (Food.cpp) file did lack some comments. To enhance clarity, we recommend adding comments within the generateFood() function in Food.cpp to help someone unfamiliar with the project better understand the logic and reasoning behind the implementation.

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.
   - After looking through all the files, we came to the conclusion that their code does adhere to these standards, which is another reason why their program was relatively easy to understand. They achieved this by using appropriate indentation, and separating different function definitions with appropriate spacing.

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)
   - After thorough testing of their program, we did not experience any buggy issues or behaviour. The program ran smooth and fast.

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.
   - After running the drmemory command, we did not see any memory leakage.

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?

- No, the inclusion of the Pos struct was unnecessary to the objPos class. In the program, we only used the struct to retrieve the coordinates of objPos objects. This could've been implemented as just the coordinates into the objPos class.

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).

objPos
- x : int - y: int - symbol: char
+ objPos () + objPos (xPos: int, yPos:int, sym: char) + setobjPos (x:int, y:int): void + setobjPos (xPos:int, yPos:int, sym: char): void + getobjPos (): objPos const + getSymbol (): char const + getSymbol() : char const + isPosEqual (refPos: const objPos*): bool const

+ getSymbolIfPosEqual (refPos: const objPos*): char const

- In this implementation, the Pos struct is completely removed and the coordinates are implemented in just the objPos class. This makes it easier as an unnecessary part of the program is removed.