COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members _____ wangd148_____ _____rizvia21_____

Team Members Evaluated _____anila4_____ _____choprm9_____

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 main loop's structure is clear and easy to understand. Each object in the program has played a proper role during the interaction in different functions. "snake" manages the snake's behavior, "food" handles the food and bomb generations. "game" reads the input and game state. The dynamic allocation of these objects is handled well, and collision logic between the snake and food is implemented logically.

   [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.

   C++ OOD

   Pros:

   1. Encapsulation
   2. Reusability
   3. Modularity
   4. Dynamic allocation

   Cons:

   1. Complexity
   2. Potential memory leakage

   C procedural design

   Pros:

   1. Simplicity
   2. Fewer dependencies between objects

   Cons:

   1. Lengthy coding
   2. Lack of reusability

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 lacks a good number of sufficient comments but filled with numerous commented-out codes that are not necessary there. If it was reviewed by others who did not do the same project, it could cause some confusions. The work seemed to be submitted in a rush as there were too many parts that could be improved besides readability. I would better organize time well to ensure that there could be enough time to reinforce the project after completion.

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 code follows good indentation and has sensible white space for readability. However, the unnecessary unused codes that are commented out has caused some confusion and made the code lengthy and unorganized.
   Deleting unused codes and comments will improve the readability.

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 runs smoothly without any significant bug shooting problems. However, the snake is not able to immediately stop when the self-collision or hitting bomb occurs. The bug could be caused by where it sets and checks the exitFlag or loseFlag conditions. If the condition is checked after the snake's movement, the game will not be terminated immediately.

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.

   This group's project does not cause any memory leak which reflects a good attention of deleting any on-heap objects.

Project Reflection

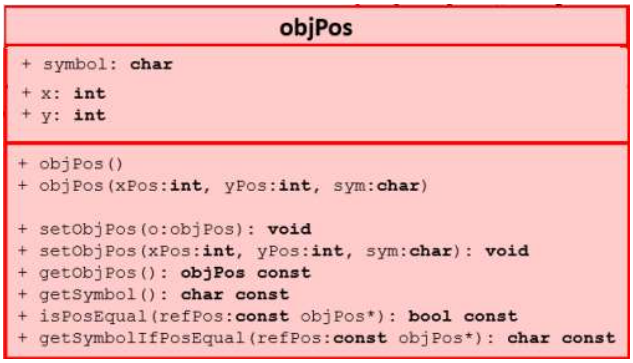
Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team's implementation in addition to your own, reflect on the following questions:

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

   First the "Pos" struct groups related elements including 'x' and 'y' into a cohesive unit which keeps the position and symbol data easy to access and modify if needed. Secondly, The use behind "Pos" struct is a classic signature of C++ composition that aligns with OOD principles, allowing "objPos" to combine "Pos" data with other methods. Last, it helps with the readability of understanding that "Pos" as a fundamental building block. In conclusion, I believe it is sensible.

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).

   Another design that is believed to be *counterintuitive* is to eliminate the struct and scatter 'x' and 'y data as separate, ungrouped and public members of objPos.



   In this alternative design, 'x' and 'y' are members of objPos that has public scope which reduces the cohesion it has in the original design. In addition to that, objPos would need to manage 'x', 'y' separately, increasing the inconsistent updates or potential bugs. The struct in the original design allows modular handling of positions, enabling reuse in other contexts. Whereas, it does the opposite here which ties those properties directly to objPos. Reusing and modularity is reduced.