COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members elkhaih patem215

Team Members Evaluated suljaka bernan3

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 logic in project.cpp is well-structured, with distinct roles for the Player, GameMechs, and Food classes. The interaction between these objects is easy to understand: Food integrates with the players movement to check for collisions, while the player depends on gameMechs for inputs and game conditions. Strong Object-Oriented design concepts are reflected in this design.

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
        - Classes like Player, Food, and GameMechs have clear responsibilities, improving readability and maintainability.
        - objPosArrayList dynamically tracks the snake's body and food, simplifying logic compared to static arrays in PPA3.
        - Centralized logic (e.g., Food::generateFood) avoids scattered functionality, making debugging and future changes easier.
        - The design supports easy feature expansion, like adding special food items or new gameplay mechanics.
    - Cons
        - Memory Management Complexity: Dynamic allocation (e.g., new objPosArrayList) adds risks like memory leaks, requiring careful destructor implementation.
        - Rule of Six implementations, such as for objPosArrayList, increase code length and complexity compared to simpler procedural functions in PPA3.

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 reasonably commented which makes the code a lot easier to understand.

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 formatting and Indentation was very good throughout the code, making it easy to read.

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)

   Their snake game has a smooth playing experience with both correct movement and wraparound logic. The idea to make the special food increment the score by a random number between -10 and 10 also made the game more fun and challenging. Looking over the code and playing the game multiple times we couldn't find any specific bugs or glitches.

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.

   Both of our Dr.Memory is not working and so we were unable to run it to ensure that there is no memory leakage in their code. However, just from reading their code they do have destructors that seem to have been implemented correctly to deallocate heap memory which shows that they have tried to ensure that no memory leaks occur.

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?

   A better solution would embed x, y, and the symbol directly in the objPos, this would eliminate the heap allocation. As a result the overall memory overhead would be reduced and it would also simplify the destructor logic. A UML diagram of this alternative would have integers x and y, and character, symbol, as direct data members of objPos. There would still be helper functions for things like setPosition and isEqual.

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).

   We think the objPos design is a reasonable choice for implementing the positional data as it ensures that there is consistency across the different objects. But the use of the heap-allocated Pos struct seems a bit unnecessary as it introduces more complexities without giving us any outright benefits.

As such, storing both x and y as direct members would simplify the overall design while still ensuring its functionality.