COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members ZimoFu rui zhang

Team Members Evaluated Joud Al Mowesati Vivian Le

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.

   I can easily interpret the interactions between the objects. The naming of all the objects and its functions are easy to understand.
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.

   By using an OOD approach, we can implement the body of the snake more easily by just coping the object, and using the function that was created for the objects we can mass manipulate them with ease and repeatability. Also using a well structed OOD design, it makes it possible to add new features quickly and easily as we only need to add the functions to the class instead of adding the same function to each different part of a program.

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 does offer comments. Each major parts have clear comments that describe the function of the code.
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 is formatted clearly and uses white space to separate different parts.

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)

   Zero problem when running the program. The game moves smoothly. But the same symbol is used for both head and body, makes the game play a bit confusing.

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 drmemory, the report shows no 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, first of all it added unnecessary complexity. For a relatively simple use case, adding a separate struct introduces unnecessary indirection and complexity. Secondly, the Pos struct allocation on the heap (via new) introduces unnecessary memory overhead and increases the likelihood of memory management issues. And all of this struct can already be implemented in the private part of the class directly and with virtually no down side.

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).

   I would just more the x,y and symbol variables into the private part of the class where it would be a lot easier to access:

objPos
- int x - int y - char symol
+ objPos() + objPos(x, y, sym) + ~objPos() +objPos(const objPos&) +operator=(const objPos&) + setObjPos(x, y, sym) + getObjPos(x, y, sym) + isPosEqual(other: objPos): bool