COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members Stephanie Hughes, Francesca Buckley

Team Members Evaluated noushadw, shahiz8

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.
    a. Going through the Project.cpp file, I found it was relatively easy to understand how the objects interact with each other with just the code since the functions and variables were named well making it easy to tell their purpose.
    b. Regarding positive features, again the functions and variables were appropriately named making the code easy to understand, the spaces between sections of code with different purposes also made it look less crowded, however the drawscreen function was a bit harder to read as there wasn't any space between loops making it look crowded. Another thing I observed was that there was a print statement for snake collision both in runlogic and cleanup, which I think could have been cut down to only having one print statement for snake collision to avoid unnecessary code repetition.

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.
    a. Pros:
        i. Less repetition of code.
        ii. Easy to recycle previously written code.
        iii. Makes potential code expansion easier.
        iv. Better organization.
        v. Any mistakes are centralized (e.g., if a mistake is made in C but you didn't realize it and decided to copy and paste the code, correcting it becomes a hassle).
    b. Cons:
        i. Can be overwhelming with the large number of files and code.
        ii. The code becomes more complex which can cause problems when coding and mistakes will likely occur more.

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.
    a. There are a good number of comments in most of the files, however in the player.cpp file for the check food consumption function, I believe a comment or two telling the reader what the if and else statements are doing exactly would have been better to make things more obvious even though it seems self-explanatory. Everywhere else seemed appropriately commented without overdoing it.
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.
    a. There is a good indentation throughout the project, making things more readable. More whitespace would have been used in the drawscreen function for main and in the check food consumption function for the player class to enhance readability otherwise it was good. In the case of the draw screen and check food consumption functions, I would have added more whitespace between each if statement or loop to make it clearer for the reader.

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)
    a. The Snake Game offers a very smooth and bug-free playing experience. Although, the above and beyond feature is not fully implemented properly. In the project manual, it says, "Bonus Feature – Generating more than 1 food objects at random positions with special food functions". While the program has two different food functions, only one food is generated on the board at a time. The root of this is due to only one food being chosen from a list and determining random coordinates for the one food in Food.cpp, and printing it in draw screen of Project.cpp. Some debugging approaches that could be deployed include setting up logical print statements so that the programs behaviour is monitored and creating test cases to ensure smaller parts of code work as intended based on the manual before implementing it into the rest of the program. Additionally, hypothetical solutions to make the program more efficient such as adding break statements to loops and switch cases to allow for faster runtime, as well as minimizing the number of variables and unnecessary code.

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.

    a. The snake game causes 0 bytes of memory leaks and 0 bytes of possible memory leaks. Everything on the heap is deleted accordingly.

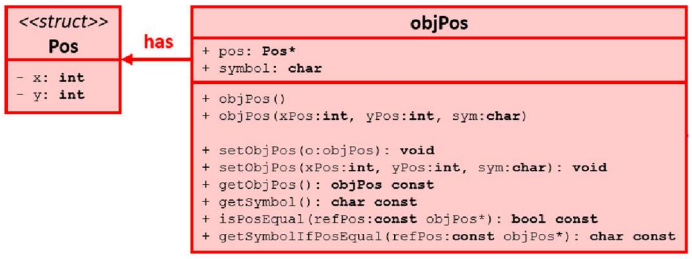
Project Reflection

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?

The compound design of the objPos class involving the Pos struct is sensible. The compound design of objPos organizes the code logically, decreasing code repetition and improving code readability. Additionally, since the struct only includes two integers, x and y position, a struct provides simplicity to organize variables without making it its own class. This makes it easily accessible from various objects. Additionally, the design strengthens its maintainability as any changes to code can be easily implemented without affecting other parts of the code in other files. Pos is frequently manipulated, allowing an efficient way for the data to be accessed.

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. <u>You are expected to facilitate the discussion with UML diagram(s).</u>



An alternative to the current design as shown in the UML diagram above, is if Pos was a class instead of a struct. This would require various changes from the original design. For instance, the Pos class would have its own constructors and destructor to initialize and clean up the data. Members could also be added to access x or y or evaluate certain conditions. An example UML diagram for a Pos class would be:

Pos
- x: int - y: int
+ Pos() + Pos(xPos: int, yPos: int, sym: char) + setX(xPos: int): void + setY(yPos: int): void + getX(): int const + getY(): int const

As shown in the UML diagram, if Pos was a class, it would need more code for constructors, destructors, methods, etc. Since Pos is relatively simple, with not much need for many methods, a class would not be as sensible. Since most of the manipulation and accessing occurs in the objPos class, there is no need for Pos to be a class as well, since the objPos class achieves this all appropriately. A struct is a much simpler way that is easier to read and less complex. Since Pos only has two data members, a struct is sensibly able to achieve its intent in the Snake Project.