COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members	riscas05, whitee24
Team Members Evaluated	sharm15-mac, muraliya

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 program is well-organized, with different objects like GameMechs, Player, and Food each handling specific tasks like game rules, the snake's movement, and food generation. However, some parts of the code, like the drawing function, are a bit too long and mix different tasks, making it harder to understand or change. Splitting these tasks into smaller pieces and making the game board more flexible would make the code easier to read and improve.

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:

- Encapsulation allows different objects to handle different parts of the game. For example, Player object controls the players movement, and food class controls the food generation and storage
- Classes can be easily reused or extended
- Modularity means program components are independent and easily tested

Cons:

- More complex than the simple linear and easy to read procedural design

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.

In all the class definition files, the code is commented sufficiently, with about 1 line giving a brief description of what every function does and the functions themselves being sufficiently self documented. In the main project file, more commenting was included, especially in code blocks that contained many nested for loops and if statements. This was not excessive since there are far more variable names to keep track of in these code blocks and the comments made it far more legible and 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.

Overall, the code has good indentation. Lines of code within loops and if statements are clearly visible with curly brackets indicating where the code starts and ends. The code is also spaced well and different member functions in the class definition files are distinguishable. The code is easy to read and different components of the code are well spaced out and defined with proper indentation and good use of curly brackets and new lines.

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 does offer a smooth, bug free playing experience. The player interacts with the food exactly as described in the on-screen instructions, and the correct exit messages are displayed corresponding to each exit condition.

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.

The game has no memory leaks.

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?

It is not very sensible because it adds an extra memory allocation element to worry about and requires more code writing to access the x and y coordinates of each element on the game screen. Having to extract a public Pos struct from an objPos class seems unnecessary.

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>

In order to improve the existing design of the objPos class, we would incorporate the x and y coordinate variables as public data members alongside the symbol in the objPos class, instead of creating a Pos struct on the heap within each objPos class. The reason for this decision is that the objPos class is relatively simple as is. With only 3 private data members in the improved design, there is no reason to encapsulate the x and y data members within the Pos struct. This improved design will allow for easier access to the data members and implementation of special member functions such as the copy assignment operator, as well as more intuitive code writing when working with objPos objects in the main code. The only downside to this improvement is that the new code may be slightly more difficult to read to someone unfamiliar with the objPos class. This disadvantage can be completely eliminated by naming the new public data members "xPos" and "yPos" instead of simply "x" and "y".

Here is a UML class diagram showing our improved implementation of the objPos class:

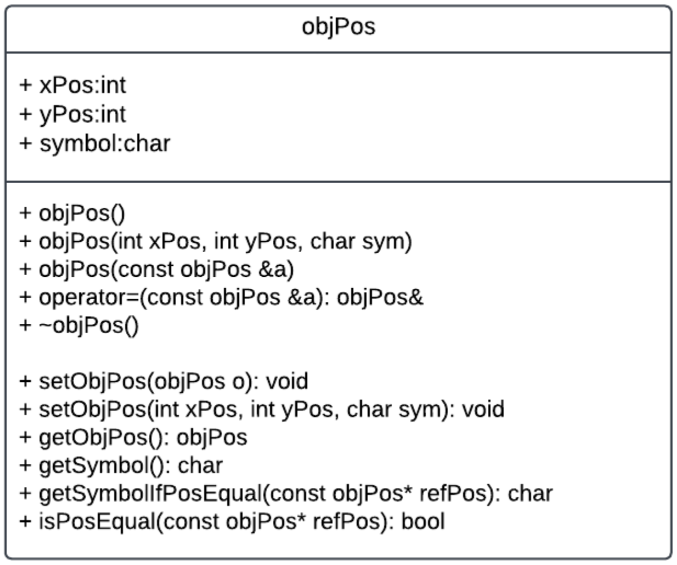
objPos
+ xPos:int
+ yPos:int
+ symbol:char
+ objPos()
+ objPos(int xPos, int yPos, char sym)
+ objPos(const objPos &a)
+ operator=(const objPos &a): objPos&
+ ~objPos()
+ setObjPos(objPos o): void
+ setObjPos(int xPos, int yPos, char sym): void
+ getObjPos(): objPos
+ getSymbol(): char
+ getSymbolIfPosEqual(const objPos* refPos): char
+ isPosEqual(const objPos* refPos): bool

Figure 1: Improved objPos Class UML Diagram