COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members Bilal Adem, Zixiang Zhou

Team Members Evaluated karrayb , rakinm

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 program loop is easy to follow, with a clear flow of GetInput(), RunLogic(), DrawScreen(), and LoopDelay(). Using objects Player and GameMechs helps keep things organized. However, relying on global variables like myPlayer, myGameMech
and foodGenerated makes the code more tightly connected than it needs to be, which could cause issues later. The way objects share state also ties them together too much, making the program harder to expand or debug.

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.

C++ Object-Oriented Design (OOD) Approach (This Project)

Pros:

· Encapsulation: Objects like Player and GameMechs manage their own data and behavior.

· Modularity: Clear separation of responsibilities makes the code easier to read and maintain.

· Scalability: Easier to extend the program with new features or objects.

· Reusability: Classes can potentially be reused in other projects.

Cons:

· Increased Complexity: Introduces additional layers like classes and methods, which might not always be necessary.

C Procedural Design Approach (PPA3)

Pros:

- Simplicity: Code is often more straightforward and easier to understand for smaller projects.

- Efficiency: Typically less memory and runtime overhead due to lack of object instantiation.
- Direct Control: Functions directly manipulate data without intermediary abstractions.

Cons:

- Poor Modularity: Functions and data are more tightly connected, making the code harder to maintain.
- Limited Scalability: Adding new features can lead to a tangled codebase.
- Lack of Reusability: Functions and data structures are less portable compared to objects.

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.

Although the code is well made it could use a few more comments just to make the viewing experience better. Like telling us what each block of code does instead of making us read the logic and figure out its purpose. This would allow anyone new working on the code an easier time getting started.

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 mostly follows good indentation and uses newline formatting and white spaces to make it reasonably readable. Blocks within functions, if statements, and loops are consistently indented, which helps in understanding the program flow.

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, the snake collision with the food works as well as its collision with itself resulting in a game over works perfectly. The snake wrap around logic works as intended and the game registers the user input properly.

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 snake game does not have any memory leakages.

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?

Yes, I think it can work. It encapsulates position x and y within a POS struct. It provides better modularity and it can separate the concerns. And in this case, the objPos class can focus more in managing symbols and interactions and make them as a group instead of dealing with only the coordination stuffs.

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).

Here's an example:

```
Class objPos {
Public:
Struct Pos {
int x;
int y;
}  position;
Char symbol;
```

In this case, everything is public, the struct pos is public. So, the value inside Pos can be changed directly by outside like obj.postion.x=-1; . So what's the best way is that we have keep objPos public, but the Pos structure should be private so that keep the safety of the data, and keep the rule of the OOD.