# COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members dennic4 - Cheryl Dennis kianmeha - Athena Kianmehr

Team Members Evaluated slytherin

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.

This code provides an organized and clear structure of implementing the objects in the classes and shows evidence of an efficient coding process as this main logic has been made and refined. The GameMechs, Food and Player pointers are immediately initialized in the global scope to be used throughout the program, but the exitFlag for the program is also initialized here, which can be redundant to the program. This is because the GameMechs object is already tracking the status of the exitFlag using the methods getExitFlagStatus() and setExitTrue(), so exitFlag is not needed to be directly accessed. Therefore, exitflag does not need to be initialized or used in this main program loop. Also, the gameMechInstance pointer initializes the size of the game boarder, but the team could of utilized the getBoardSizeX() and getBoardSizeY() from the GameMechs class to retrieve this information. In the RunLogic function, the function MacUI_clearScreen() is called when the snake head eats the food. This call is redundant as MacUI_clearScreen() is already called in the DrawScreen and clears the screen at the start of the function. In the CleanUp function, this team provided the lose status for the player here. The logic of these statements utilizes the objects well, but would preferably be better in the DrawScreen function, where the other screen implementations occur. I also think that it avoid any accidental exits, the “else” condition should clarify to if the getExitFlagStatus() is true, so it only runs of if the escape command is pressed. The overall object interactions in this main program remain consistent and are utilized effectively.

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:

- All coding logic is organized in separate classes so they can be called separately within each function
- Avoids redundancy within each function, as you don’t have to rewrite the logic in each function but instead call the class objects to keep everything organized
- Setting private and public scopes in the class allows to restrict what data can be modified, providing protection from unwanted changes from collaborators
- Allows to incorporate more complex applications of coding in an organized manner, hiding excessive details and showing logic that is necessary to understand
- Can collaborate with others more easily by making changes in separate classes independently without interfering with the main functions

Cons:

- Makes code harder to understand for outside viewers if they don't understand the logic within the classes and pointers classified in the main logic, while C programming is more straightforward to understand
- the more classes incorporated can cause excessive running time and more memory leakage as more information is being implemented at a time
- can be difficult to debug the code, as you must look into all the classes to find issues in all coding logic and be confused when multiple objects are incorporated
- code would become heavily dependent on external libraries and workflows from other classes, which can make compatibility issues

## 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 provides sufficient comments when defining new variables and pointers and explains the logic for all the functions that were not intuitive. The code is very consistent in commenting the assigned variables in all class files and provides comments next to 'if' statements and all loops to explain the what the code is intended to do and logical reasoning behind these implementations, demonstrating proof of a well done self-documenting coding style. There are no improvements that I would make in their code documentation.

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.

This code was curated with great organization and mindfulness of space, ensuring that all variables that are initialized are grouped closely together. All code logistics (statements and loops) are indented in a neat and organized matter, providing great readability. All coding logic written within a function is separated in an appropriate manner so that it is differentiable from the other functions in the file, and the functions are separated by indents, so the space is utilized wisely. There are no improvements that I would make in their code indentation.

## 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 provided a smooth and bug free experience. The mechanics of the game, including the snake movement, food generation and collision detection worked really well without any buggy implications in the coding interface.

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 cause any 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?

The compound object design of objPos class is sensible because it allows the separation of the position logic from the rest of the object's methods. It also improves the readability and reusability of the code.

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).

Instead of the struct Pos design an alternative objPos class design that would be relatively counterintuitive is defining each variable separately instead of putting them in a struct(ungrouping the variables). This would reduce the readability and the reusability of the class as it removes the encapsulation of the position variables.