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

| | | |
|---|---|---|
| Your Team Members | Adam Suljak (suljaka) | Noah Bernardin (bernan3) |
| Team Members Evaluated | patem215 | elkhaih |

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 **30 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.

It is overall fairly simple to determine how the different objects interact with each other in the program logic. Pointers to both the game and player mechanics are created adequately and used properly in the Draw Screen function. The draw screen function was overall well implemented, and we both liked the way they did the generation of characters by setting a flag to determine if a certain position had already been drawn, and then printing any characters that have not been printed yet as a spacebar. This is the opposite of our implementation, as we used local objects to check if either the snake or food had been selected, followed by using an if/else structure to print rather than integrating Boolean variables into the design. We do believe that their implementation is overall the better design. The group's implementation of using local objects to simplify code readability, such as when defining a local objPos object to resent the current character of the snake, was also beneficial to code readability.

However, there are some negative aspects of this group's design. When drawing regular foods, the group does not appear to use a food list, and instead hardcodes a value for the number of regular foods into the normal food loop in DrawScreen(). Similarly, it does not appear that the group implemented a foodBin array list, which complicates the design of DrawScreen(). The group also relied on global variables for exitFlag, boardWidth, and boardHeight, and while they retrieved these values from methods in the gameMechanics class, we believe it would be a better objected oriented design to call those methods directly when they are needed rather than using a global variable. A similar issue was observed with getting the input, as input was collected in the Project.cpp rather than gameMechs.cpp, with the input collected being passed into gameMechs. We believe it would be a stronger objected oriented approach to implement this function directly into the gameMechs class.

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 of C++ OOD Approach:

- Better organizes the code and makes it more clear what is being referenced
- Allows for better code re-use (eg. Foodbin Array List and ObjPos Array List can be instantiated using the same class)
- Easier to alter the code if changing variables throughout the design (eg. boardSize, etc)
- Very good for large scale projects with multiple developers.
- Easier to facilitate parallel development.

Cons of C++ OOD Approach:

- More complex to implement for new programmers
- Not suitable for smaller project sizes with a singular developer, as it will result in more work than needed.
- Requires additional sanity checks to prevent improper access and memory leak.
- Can be more confusing to analyze for new developers.

## **<u>Peer Code Review: Code Quality</u>**

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 offers sufficient comments that help one to understand what is being done at each section in the code. This made it much easier for us to determine where different functionality is being completed, which helped us when completing the functional evaluation section. For example, in the drawScreen() method, it is clear what characters are being printed in what section of the function, which made the code very straightforward to analyze.

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 offers good indentation, white spacing, and newline format, improving readability. Comments are adequately created either beside of above the section of code they are describing, such as in Player.cpp and objPosArrayList.cpp, and this was overall observed to be very well done. Similarly, the commenting done to understand objPos.cpp was very well done.

## 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)

   While the program ran smoothly, various bugs were observed during program runtime. One of the first bugs we noticed was that the when the regular food was collided with, the location of the special food would not update. This is likely caused by an issue with the generateFoods() function, as in order to generate a special food, it needs to previously be inactive, which is not set in the checkfoodcoll() function. Similarly, there were times when the regular food would randomly generate in its previous location, which can be altered by investigating generateFoods(), adding an additional check to ensure that previous food positions are not repeated. The snake or the regular food was also seen to overlap the special food at certain points, which caused it to disappear from the display as it is printed last in DrawScreen(). We would recommend that the generateFoods() function in GameMechs.cpp be thoroughly investigated, as various sanity checks are missing that could resolve these issues.

   Similarly, upon pressing a key that is not w, a, s or d, the snake will stop moving, which we believe is caused by the use of a default statement to hold the stop state in movePlayer(). Instead of this implementation, the stop state should be its own state that can only be invoked upon program runtime. This results in buggy implementation of player movement, as once the snake is stopped, the user is given the ability to move the snake in all directions, including back into its own body, immediately resulting in the snake's death and ending the game. Our overall recommendation to the group would be to stress test the program with longer snake sizes, as well as to implement a food class with an array list containing all foods, both regular and special, as well as the respective data of each one.

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 was tested using Dr. Memory, with no memory leak being detected. However, this was difficult to profile, as within the DrawScreen() function, cout is used instead of MacUILib_printf(). This prevents anything from drawing on the screen on a non-Windows environment, leading to issues as we both use Linux and Valgrind to profile memory leak.

## Project Reflection

Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team's implementation in addition to yours, reflect on the following questions:

1. **[3 marks]** Do you think the compound object design of objPos class is sensible? Why or why not? We don't believe that the compound design of the objPos class is very sensible as it needlessly complicates the process of accessing the position data for the snake, and foods. We noticed that every time we wanted to access the position of the object, we always had to go through objPos,

however, no other classes use the objPos struct at all. As such, it is not necessary to have this struct, as it is only being accessing by one class.

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.</u>

   As the only class that uses the Pos struct is objPos, a more simple implementation would be to implement the x and y positons, as well as the symbol of the character, as private fields in the objPos class, according to the UML diagram below.

| objPos |
|---|
| - int x<br>- int x<br>- int symbol |
| + objPos()<br>+ ~objPos()<br>+ objPos(a: const objPos&)<br>+ operator=(a: const objPos&): objPos&<br><br>+ setObjPos(o: objPos): void<br>+ setObjPos(xPos: int, yPos: int, sym: char): void<br>+ getObjPos(): objPos const<br>+ getSymbol(): char const<br>+ getSymbolIfPosEqual(refPos: const objPos*): char const<br>+ bool isPosEqual(const objPos* refPos) const;<br><br>+ setX(x: int): void<br>+ setY(y: int): void<br>+ getX(): int const<br>+ getY(): int const |

The new functionality implemented compared to the base design of objPos.h are private variables for the x position and the y position, as well set and get method for both X and Y. Then, when the values of x and y are needed by other classes or functions, they can be easily set by using objectName.setX(value), and retrieved by using objectName.getX(value). The implementation for the setting and retrieving the Y value would then be identical. This eliminates from a struct, and while it does add fundamental set and get methods to the program, it better reflects data encapsulation, as the x and y values are now in the private scope, as compared to the struct, which was in the public scope.