

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

Your Team Members                      hussam89-comp                      singhy18

Team Name: Snake\_AandY

Team Members Evaluated              marcuscc2024                      zhue13

Team Name: Children of Chen

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.

Yes, we were able to easily interpret how different objects in the main logic interacted with each other. For example, if the food position and the player position are equal to each other, actions such as incrementing score in the GameMech's class, growing the snake's length in the objPosArrayList class were correctly taken. The food object was correctly encapsulated within the GameMech's class, which allowed for these interactions to be taken easily, and for the interactions to be easily understood by anyone who is viewing the project for the first time. There are, however, some negative features in the main logic of this program. First, the objPosArrayList snake object is not included/referenced within the Player class as instructed. This has led this team to include their snake movement and snake length growth logic, game ending conditions, etc., within the main logic of the project. This takes away from the modularization purposes of a class and clutters the main program file. Other than this one negative, the main program of the project correctly initializes different class objects, and sets interactions between them that makes the game possible to play.

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 large advantage of the OOD approach used in the project is encapsulation. This allows us to combine data and functions into classes and improves modularity of the code. The OOD approach also allows for easier scalability of the program allowing the programmer to easily manage more complex functions and data. OOD is ideal for large scale projects that require modularity.

The procedural design approach allows for easier debugging since functions and data is separated making it easier to pinpoint issues on a smaller scale. The OOD approach is also more complex to implement requiring knowledge about creating and accessing classes. Procedural design is useful for small scale and less complex projects.

### **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.

Yes, not only does the code offer sufficient comments, but also has a self-documenting coding style that leaves no area for misinterpretation or confusion. For example, all variable names are meaningful and self-explanatory. The instance of the `objPosArrayList` object, which represents the snake, is named "snake". Within the food generation algorithm of the `GameMechs` class, the random x and y coordinates generated for possible food positions are named "randomx" and "randomy". Additionally, there are a plethora of comments dispersed throughout the project that explain the various class methods, main logic algorithms, etc. For example, within the main logic of the program, there are comments explaining the most important algorithm of the game – moving the snake and growing it's body depending on certain conditions. The developers explain that for the snake to grow, for example, the coordinates of the food must match the coordinates of the player. All in all, the code developed is very easy to interpret; there are no shortcomings in this project criteria.

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.

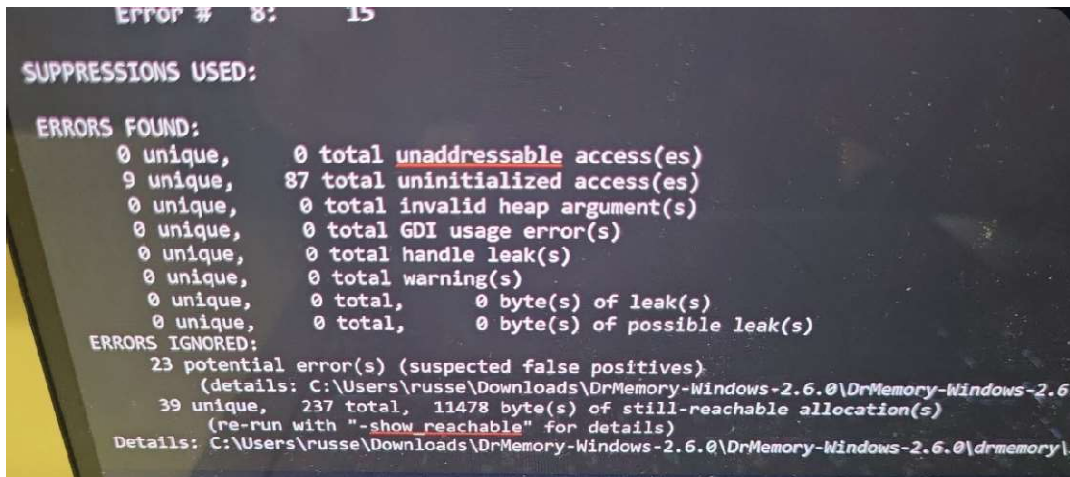
Yes, the code has good indentations and whitespaces across the entire program's scope, and deploys newlines whenever game messages are displayed for a pleasant user experience. However, the developers could have deployed newlines within the code for better readability in a few areas. For example, in each for loop, the starting curly bracket is left on the same line as the actual for loop statement. This makes the code look "messy" and can be improved. Other than this, the code is completely readable and follows good formatting practices.

### **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 snake game completely offers a smooth, bug-free playing experience. There are no buggy behaviours. The developers have gone above and beyond by implementing speed variability of the snake's movement. The wrap around movements are smooth, and the game correctly follows game ending conditions upon the user pressing the exit key, or running the snake into it's own body.

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.



```
ERROR # 8: 15
SUPPRESSIONS USED:
ERRORS FOUND:
0 unique, 0 total unaddressable access(es)
9 unique, 87 total uninitialized access(es)
0 unique, 0 total invalid heap argument(s)
0 unique, 0 total GDI usage error(s)
0 unique, 0 total handle leak(s)
0 unique, 0 total warning(s)
0 unique, 0 total, 0 byte(s) of leak(s)
0 unique, 0 total, 0 byte(s) of possible leak(s)
ERRORS IGNORED:
23 potential error(s) (suspected false positives)
(details: C:\Users\russe\Downloads\DrMemory-Windows-2.6.0\DrMemory-Windows-2.6.0\drmemory-ignore.txt)
39 unique, 237 total, 11478 byte(s) of still-reachable allocation(s)
(re-run with "-show_reachable" for details)
Details: C:\Users\russe\Downloads\DrMemory-Windows-2.6.0\DrMemory-Windows-2.6.0\drmemory-ignore.txt
```

There are no leaks in memory.

## **Project Reflection**

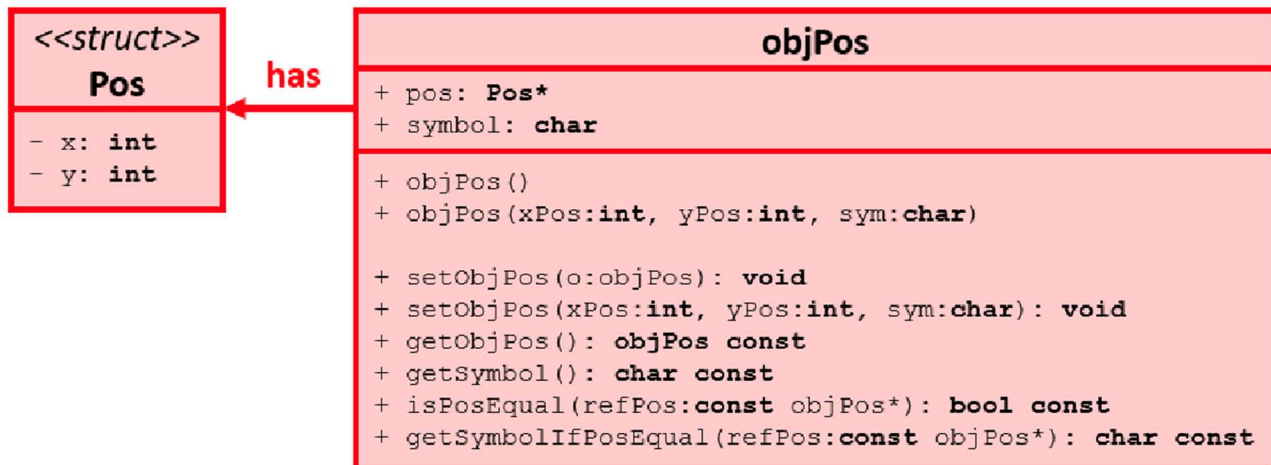
Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team's implementation in addition to your own, reflect on the following questions:

1. **[3 marks]** Do you think the compound object design of objPos class is sensible? Why or why not?

No, we do not think that the compound object design of the objPos class is sensible. While having a struct field member within a class is not wrong, there is really no need to include it within the class. A struct is a C's version of a class in C++, except that all data members in a struct are public. C++ is a language that supports object-oriented design, such that there is no purposeful use of a struct within a class. We may have easily just taken the x and y player position data members outside the struct and have declared them as normal data members. This makes complete sense, since the objPos class represents the position of a player, there is no need for further encapsulation via the struct. If there were many data members within the class and the use of a struct helped modularize the coordinate data apart from a variety of data members, perhaps then it would make sense to include it. However, this is not the case and leaving the x and y coordinates as explicit data members makes for a simpler class design.

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).

UML diagram (How it should look after combining:)



These are combined due to heavy memory usage of the struct Pos.

