COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members	Arushigan – 400505741, Shwethan 400515492
Team Members Evaluated	chathena

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 very well structured, and we were able to easily interpret how the objects interact with each other. As far as we could tell, we found no negatives or errors after reviewing the main logic in the main program loop. Everything was perfect in terms of how they integrated the objects effectively and it was easy to tell how they interact with each other, as well as excellent clarity and robustness. Overall, from our observations this group did exceptionally well in this aspect of the project.

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.**

   There are various pros and cons of using the C++ OOD approach for this snake game versus the C procedural. The negatives is that C++ is more complex and it often uses more memory too due to the complexity. However, the pros overweigh the cons. Due to the OOD approach, it's easier to reuse and add on to various functions and classes to add extra features if needed to the game. It is also more organized through encapsulation where classes group related functions. In summary, due to the main pillars of OOP, encapsulation, polymorphism, abstraction and inheritance.

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 does offer sufficient commenting that allows the viewer to clearly understand what the major points of the code were. Each major method within classes had comments explaining what it did, and various points within those methods where it may be confusing also was provided comments, too.

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 also follows good indentation and sensible white spaces, too. It provided enough white space between methods so that it didn't seem too crammed.

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)**

   After playing their game, we came to the conclusion that there were no buggy features and the program ran smoothly. It was evident that all their wrap around logic was implemented correctly as well as growing the snake, counting points, spawning food, etc. We liked how they kept it simple with one symbol for regular food and one symbol for the superfood. Overall the playing experience was super fun, smooth, and enjoyable with no buggy behaviour even after several tests.

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.**

   Upon completing a drmemory check, there was no memory leakage. After visually inspecting the code as well it was evident that this group allocated and freed memory properly.

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 the objPos class is sensible as the Pos struct correctly encapsulates the position related data separately. In other words, it seperates the position logic struct from other attributes in the objPos class. This makes so that its easier to reuse or modify. You can do this without affecting the rest of the class which makes it more modular.

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>**

Another way to implement the objPos class design is that they could integrate the position attributes, the x and y, directly into the objPos class. This would mean that the Pos struct would be eliminated. This also means that it would simplify the structure so for easier project, we would be avoiding an extra layer of abstraction. An example of an UML diagram could be that the class name is objPos, the attributes/data members are x:int and y:int which are private, then the public methods are setPosition (x:int, y:int) which is a void function, then the getPosition function with parameters of the int would return these values. Finally, the move method adjusts the positions based on their relative movement values taking in int values, too.