COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members Deev Patel pated201 Ray Wu wur99

Team Members Evaluated tripav2 ismailj05

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 code that was written was very easy to follow and understand. Through comments and class setup I was able to easily follow what the program does and how the developers wanted the program to work.

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:
   The OOD approach to this project comes with many advantages and allows the user much more power when it comes to different aspects of the game. Since different aspects of the game are each different objects, it allows for easy code reusability and greater control. For example, the food class can easily be modified to have multiple pieces of food just by instantiating another food object. If I want to change the size of the game board, I can just change the number in the object initialization of GameMechs. When all the code is encapsulated within itself and left general to accommodate a variety of different design ideas. On top of that, an OOD approach makes it easier for other people to read and understand how the code is meant to work.

Cons:

The amount of code throughout the process is longer than what it could've been if it were all done procedurally. Along with that, it would also take more time to develop since OOD has a more defined and strict structure. Another disadvantage is the learning curve for newer developers. It takes time to learn and understand how to implement OOD properly and efficiently.

OOD is a very powerful tool that is essential for large scale software development but has a learning curve and may be difficult to pick up at first.

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 comments accurately describe the function and method of each of the different functions and classes. There are comments that describe what each function does and comments throughout explaining how they are done/implemented.

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 does follow good indentation and employs sensible whitespaces. It's very clear where each function, for loop, and while loop starts and ends. Additionally, the use of whitespaces in variable definitions and function and loop parameters makes it very easy to read and understand them. However, at times, there could have been better and more consistent use of newlines to separate parts of the code to promote better readability. Doing so would decrease crowding on the screen and make it easier to locate parts of code you want to find. I would implement this improvement by adding 4 lines of white space between each separate function.

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 did play very smoothly and no major issues were encountered but there was one slight issue that sometimes occurred when playing. Sometimes the spawned on the body of the snake. The way this can be fixed is if the coordinates of the food was checked with the coordinates of the snake before it is placed, if there is any overlap, the coordinates of the food will be regenerated.

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.

   After running drmemory on the snake game, we find that it does not cause any memory leaks or possible memory leaks.

Project Reflection

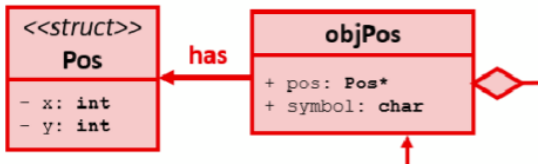
Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team's implementation in addition to your own, reflect on the following questions:

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

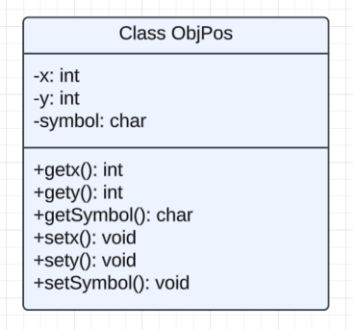
I think that while it does follow OOD principles and works, I believe that having the pos struct was unnecessary and redundant.

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).



The current UML design for objPos and Pos does work but has redundancies. The objPos class always contains a Pos and symbol. I think that the Pos struct is redundant as it can be easily combined into one class that stores a symbol and coordinates without affecting any functionality of the code.



If the class was made like this, the functionality would remain the same but would be more compact. Normally in OOD, classes can be reused in other classes so they can be implemented to do different things. This is shown in our snake project where objPosArrayList was used to make the snake and to make food. But in the case of Pos and objPos, there isn't much you can do differently when they are separated, so you can combine them into one, simpler class.