COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members Gobind Kailey, Abdullah Asilar

Team Members Evaluated Alsamml & Seetk

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.

Positives	Negative
• All lines are in the correct order • I can see interpret how all the lines of code interact. • Their drawing function looks organized and easy to understand.	• More comments needed in some parts • They called their updatePlayerDir() in the movePlayer() function, but I rather place it in the main project.cpp under the RunLogic, so that if someone is trying to get the gist of the program, they will know that updatePlayerDir() was used earlier, than having to go into the movePlayer() function and seeing that it has been placed there. • I would've placed lines 79-80 of the Project.cpp in the initialize function, instead of placing it in the Drawscreen because we are initializing the playerPos and foodbucket() in a location where we should just be drawing.

Figure 1: Table for pros and cons

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 OOD

- Creates a simple and organized code with all of the classes
- Good for large scale applications such as the snake game.
- Classes can be reused, for example the arraylist class can be used for player snake and

the food
bucket.
Cons of OOD
● OOD is a harder concept to understand and implement than procedural design.
● It can become hard to debug when a problem arises as you have to debug through multiple classes and pointers.

Pros of Procedural Design

- Writing code in a sequential order is much easier, and especially if you are starting out, it can be a great tool to learn and understand code, and helps you focus on logic.
- Procedural design is easier to implement for smaller projects such as the iterations we did for
PPA as it can be faster.

Cons of Procedural Design

- When you are creating a project with a team, it is not a good option to go with procedural because the code will just be in one file and without encapsulation it can be hard to assign tasks for group members to work on.
- Can become difficult to manage in large 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.

   In some parts of the code commenting in great, but in some files, there is near to no commenting, for example in objPos.cpp file, in 83 lines there is 1 line of comment or in objPosArrayList.cpp there is 1 line of comment in 91 lines. To improve this, I would quickly summarize some of the more thinking required functions in one line. This would result in a better understanding of the code. The self-documenting in this project is perfect, as in files like GameMech.cpp most of the functions consist of one line of code, which is self explanatory, so not adding comments to that is good.

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 follows standard indentation rules and is easy to read and follow. The white spaces and newline formatting all look good.

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)

   There is no problem with the gameplay or main logic of the snake game, there is just an error due to the borderline shifting at the end of the game ending with the collision of the snake with itself. This behavior is most likely caused by a logic error in the checkSelfCollision or movePlayer functions, where the game continues to update even after a collision is detected. Potential debugging approaches include making the game stop completely after the collision, which can be solved by adding return or break. These approaches will improve the game gameplay.

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.

   By using Drmemory, I noticed that they had 2 bytes of leakage, but when reading through the code, I didn't find any deallocation errors, everything seemed deallocated correctly.

Project Reflection

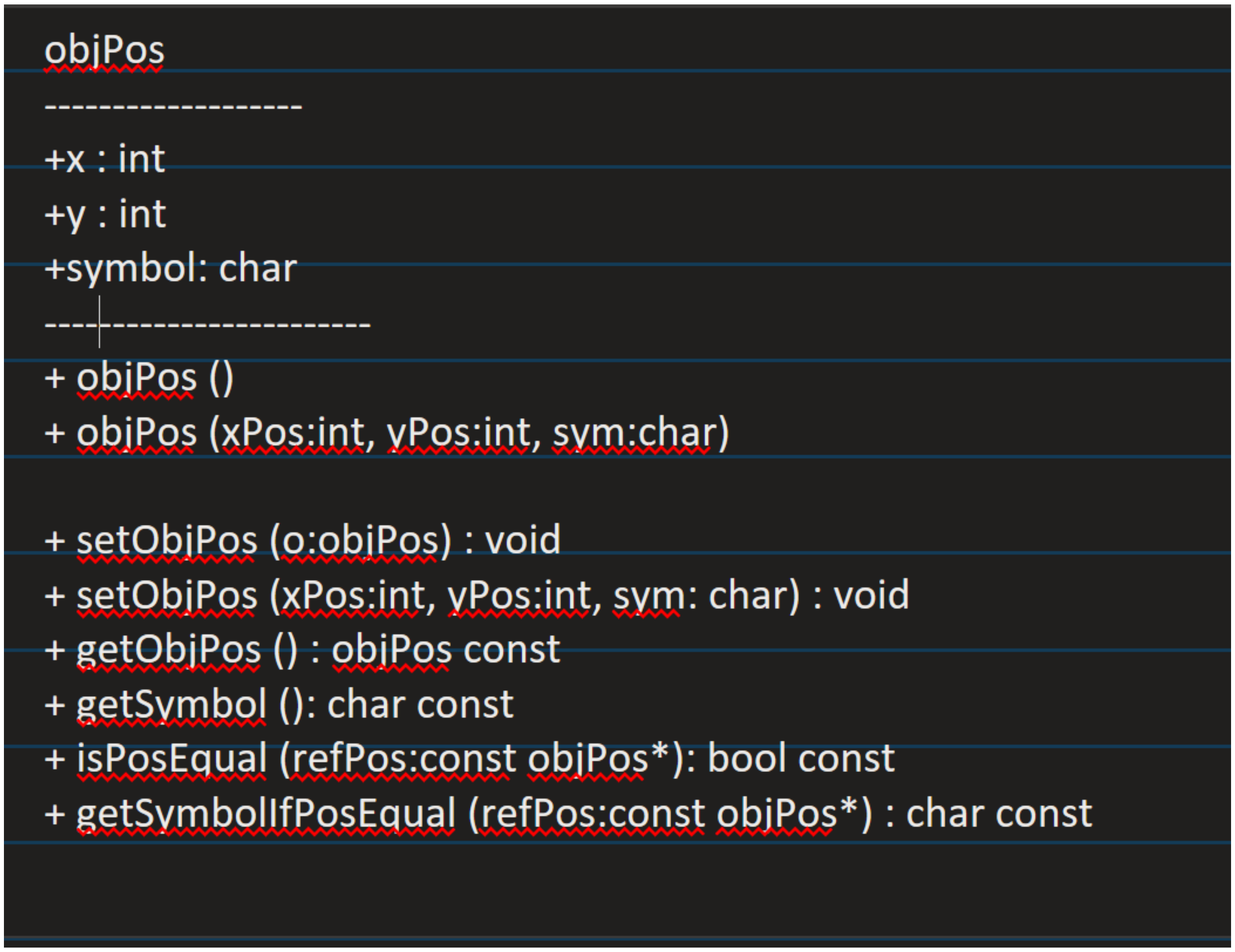
Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team's implementation in addition to your own, reflect on the following questions:

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

   Yes, the compound design of the objPos class, with its additional Pos struct, is a sensible approach as it encapsulates positional attributes (x and y) separately from other properties like the symbol. This allows the data contained in this class to be handled in a much easier and more organized manner. It also provides a proper memory management by preventing memory leakage that may occur due to writing this implementation in accordance with the rule of four. The design manages the position variables of the object separately which provides easily maintainable and extensible 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).

An example of a less effective design than the one we used in this project is a design where x,y and symbol are declared separately directly to the objPos class in the public scope. Even if this simplifies the modification code, it will make it more difficult to manage these attributes if the project expands in the future. The current design is more effective because it groups these attributes in Pos structure which makes the code cleaner and easy to handle during the process of improving. As mentioned before, Pos structure simplifies extending the class as adding new methods without complicating the class.



```
objPos
-------------------
+x : int
+y : int
+symbol: char
------------------------
+ objPos ()
+ objPos (xPos:int, yPos:int, sym:char)

+ setObjPos (o:objPos) : void
+ setObjPos (xPos:int, yPos:int, sym: char) : void
+ getObjPos () : objPos const
+ getSymbol (): char const
+ isPosEqual (refPos:const objPos*): bool const
+ getSymbolIfPosEqual (refPos:const objPos*) : char const
```