COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members Joshua Guo, Suchir Ladda

Team Members Evaluated _____Kun Xing_____ _____Ibtisam Alhasoon_____

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.

Our MAC ID's:

laddas

guo184

Other team MAC ID's:

xingk8

alhasooi

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 project logic interacts well with each other. The code in the main project.cpp file mostly follows a straightforward OOD design method; however, there are a few issues. The DrawScreen logic has a lot of printing logic that I believe can be implemented in one of the objects used in the program. Their printing method manually prints the objects onto the board using loops and each object's position. This causes several nested if statements and nested for loops, which severely decrease the readability of the code and make it much more complex. I believe that the best way to reduce the complexity is to create a double-character array on the heap and place all of your objects in there. That means printing the output will be much easier to do and reduce excess logic in the project.cpp file. Furthermore, the CleanUp function also prints a very large message that I think can be moved into one of the methods in any of the objects present in the project.cpp file.

I think the code did very well in the other parts of the project. Other than those two functions, the project is very lean in logic, and all of the logic is stored in each of the objects. For example, the runLogic function comprises only two lines, both of which are methods from the player object.

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.

The OOD used in this final project has several benefits. First of all, it allows for easier iterative design, with the project being very long and having lots of features. We were able to implement each step modularly and easily, without being confused by each other's changes if we were to develop it procedurally. The modularization of the design into multiple files allows for easy readability, as not all of the features are jammed into one file but are instead spread out. Furthermore, the design used in the final project allows the designers to debug much more easily. In this project, we were able to easily spot which part caused issues by running the debugger, and it would identify where the logic fault was caused, whether it be in either Player.cpp or the GameMech.cpp file.

However, OOD also causes a lot of bloat and increased file size. The memory required to run this program if I were to code it into one file would require significantly less space, due to the inclusion of several libraries over and over again in multiple object files. Furthermore, in general, you have to write more code because of the nature of the rule of 6 when it comes to creating objects. If I were to complete it in one file, the total lines of code required would decrease.

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 offers an inadequate number of comments. For example, there are a good number of comments in most of the files explaining what is occurring. However, the project.cpp is entirely devoid of comments. Project.cpp is the file that requires the most comments because in the DrawScreen function, there are a lot of nested for loops and if statements, which require documentation for ease of reading. Furthermore, even though it is straightforward, the GameMechs.cpp file does not have any comments, except for the comments that were provided by the skeleton code. The given comments, furthermore, are very barebones and could use some extra explanation. For example, movePlayer having the comment “PPA3 Finite State Machine Logic” is a bit vague and could use some extra explanation.

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 is very easy to read in terms of formatting. There is an adequate amount of white space, and the code doesn’t look cluttered. Indentation is also good as you can clearly see where all the if, else statements start and end as well as the various for loops used. Overall, the code is very easy to read and doesn’t need much improvement.

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 runs very smooth and meets all the requirements for the base game. The bonus was also implemented as there is a special food item that increases the score by 5 but does not increase the length of the snake. The UI looks nice and clearly outlines the rules of the game. There is also an appropriate message generated if you lose or quit the game. One feature that seems to be missing from that game is that upon startup the food items are always generated in the same spot. When you collect the first food item the generation then becomes random, but I don't think having the food be in the same spot at the start follows the rubric guidelines.

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 a drmemory report, there were 0 bytes of leakage found. This indicated that the team made sure to use the delete() command was used anytime memory was allocated on the heap.

Project Reflection

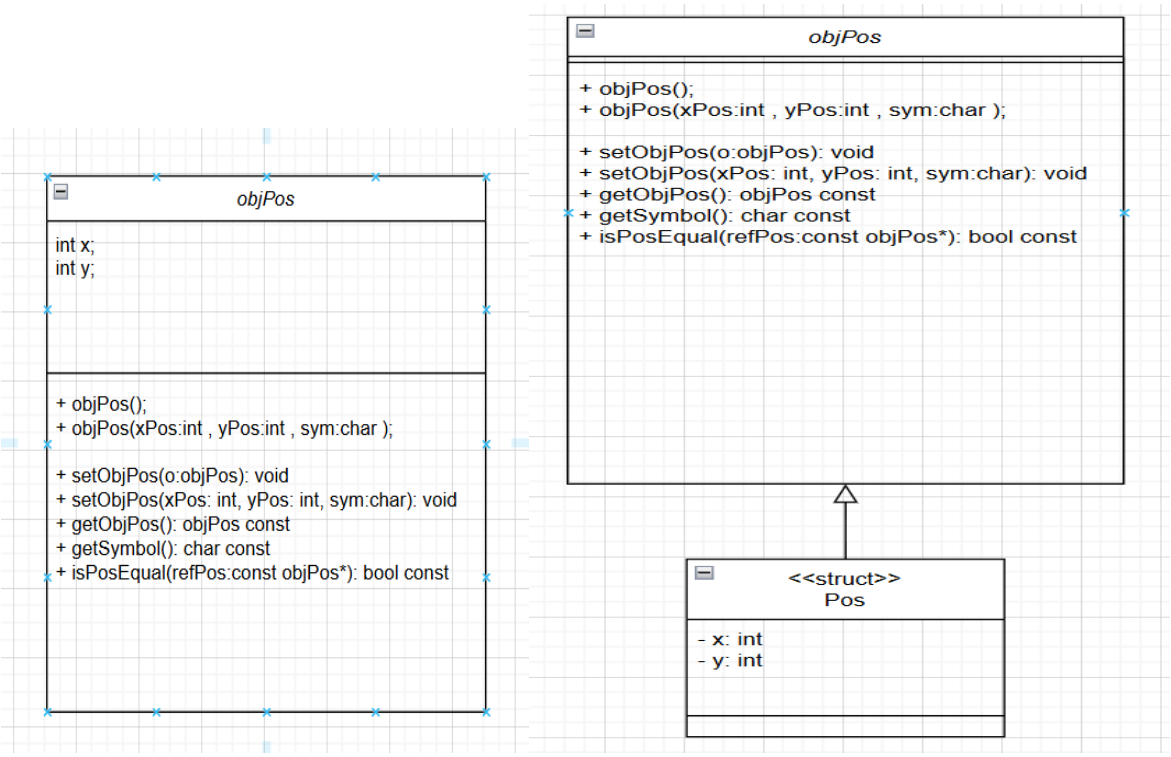
Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team's implementation in addition to your own, reflect on the following questions:

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

The compound design with Pos* allows for future code expandability if more features are to be implemented. However, given the simplicity of a game like snake, this is slightly over-engineered. I think it would be more sensible to add x and y as members in the objPos class as that would make the code simpler and less complex than it has to be. There is no reason to use the struct Pos other than printing something, and that requires a character. It doesn't make much sense to have it if you are never going to use the struct Pos for anything else other than objPos.

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).

In this case, the best way to implement this is to move the struct Pos into the objPos object by creating two pointers to the x and y declared on the heap. This would decrease the complexity of accessing the x position and the y position in our code, as it would be much simpler to find an object's position than in the current implementation.



In this UML diagram, rather than having a separate struct, it would be better design to declare int x and int y in the objPos class. The design on the right is what was originally given to us. We prefer getting rid of the struct and placing it in objPos.