COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Thanush Jayanthan Asim Nisar

Team Members Evaluated Vaibhav and Omar

Provide your genuine and engineeringly verifiable feedback. Ungrounded claims will lead to deductions.

Part I: OOD Quality

1. **[6 marks]** OOD is about sensible code modularization. Looking at the header files of each object, can you easily interpret the possible behaviours of the objects involved in the program, and how they would interact with each other in the program? Comment on what you have observed, both positive and negative features.

   After looking at the header files of each object. I was able to easily interpret the possible behaviours of the objects involved in the program and how they would interact with each other. From the objPosArrayList.h we observed the private members of the class to be straightforward as they have variables for the size of the array, size of the list, and a pointer interacting with the objPos class for the movement of the snake which was all straightforward to understand and how those objects are interacting with each other. The Food.h file also seems straightforward as the variables within the class are named well to understand what their purpose is within the code. Also, the Food.h file interacts with the other objects, and they are included by calling the #include for the objPos.h, objPosArrayList.h and GameMechs.h as all those objects are interacting with the Food.h file. The void generate() function incorporates the objPosArrayList by pass by pointer with the player and GameMechs is incorporated within the Food() function to interact with game mechanics class with a pass by pointer. Nevertheless, when looking through all the header files in their code it seems as they have done a great job with the objects interacting with each other. However, some negative features can be how they didn't really comment on some of the functionality of the functions as to what they would do which could have made the process of understanding the code even easier and more helpful for the reader such as us.

2. **[6 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.

   In the main logic within the main program loop which is the RunLogic() function. It is easy to understand how the objects are interacting with each other as there are comments for every object calling a function from their respective class. The positives are the global objects have variable names that are easy to follow within the code as the player object is just player and much more. In the initialize () function the player class was initialized with two objects as its parameters which were the GameMechs class and Food class which were also commented before when they were initialized which helped understand what was being used in the parameters. Furthermore, in the GetInput() logic it is straightforward as to how the GameMechs class is used to set the input of what the input retains. In the DrawScreen() it is easily interpreted as the GameMechs class is used in game which works with getting the borders of the game and when to end the game, the player object is printing characters as it loops through the array list for the position and the food object is implemented like the player object. The CleanUp() function looks good as every object that was instantiated on the heap was deleted to avoid memory leaks. Nevertheless, some negative features can be the tight initialization of the food class as it contains a GameMechs object, however, this not entirely bad but can lead a limited use and inflexible usages. A better way to print the boarder of the game can be used by implementing the game object with the given member functions of the GameMechs class.

3. **[5 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	Cons
• Encapsulation in C++ allows direct access to the data and easier through their member functions • Dynamic Memory on the heap • Much less code written in the main project code as everything is called through objects • Code looks more organized and much easier to follow as for every part of the code a function is involved • Not many preprocessor constants are used compared to C as they are handled in specific classes making the project code simpler	• Much more files to handle instead of one main code file • Much more complex as there are many different classes being called and header files created. • Much more prone to erros as there are default constructors in C++ that the program has that can lead to errors • Dependency of different classes can also lead to errors

Part II: Code Quality

1. **[5 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 provides sufficient comments as all the cpp files have comments explaining what the code does especially when objects are involved and what they are doing in the code. A general overview of every strategy is given in the code as to what is the purpose for huge chunks of code. Also, gave a heads up on the project.cpp file on how to test their code as they added the bonus. There are comments for every member function of the code. One way to improve over commenting Is that to leave some self-documenting code as some of the functions are straightforward.

2. **[4 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 formatting of the code seems to be well-structured and easy to understand. The white spaces and new lines in the code are strategically placed to enhance readability. For example, in the project.cpp, specifically inside of the initialize function, the initializations of the classes (GameMechs, Food, Player) are all line-after-line and separated from the other lines of code in an effort to group them based on their similar function. Most of the functions are like this, where code that has to do with each other is blocked together. The indentations, especially inside of nested conditional statements, are meticulously placed for better readability. Most importantly, the way that the code is written is consistent. The one thing that I would suggest to improve readability is to add the initial curly brace in a new line for loops and conditionals. Although not necessary, in my opinion it is easier to look at.

Part III: Quick Functional Evaluation

1. **[8 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 a technical user feedback)

Overall, the gameplay of the snake game is smooth and bug-free for the most part. However, there are some things that can be enhanced. Firstly, it seems that the wrap around logic for the top and bottom border is implemented incorrectly. At the sides, the body of the player goes into the border and appears *out of* the other side. However, for the top and bottom, it doesn't go *into* the border. It kind of just teleports from top to bottom. The possible root cause certainly relies in the draw screen mechanics. In order to possibly fix this, a minus one should be introduced inside of the if statement for the wrap around logic. To properly debug this, the first thing would be to recognize which function the issue relies in, which is obviously draw screen in this case. After that, try your best to narrow down the lines of code that are applicable to the problem. Next the debugger can be used, although in this case I feel as though print statements would be more efficient than the debugger because you can make small changes and print them to quickly see the effects on the gameboard. Another thing that I noticed was off had to do with the bonus part. In the code, it is mentioned that the "%" symbol is supposed to remove some length of the player body and subtract the score. It does this accurately. However, the food on the board does not reset once this symbol is "eaten" by the player. Theoretically, the player can continuously travel to the spot where the "%" symbol is printed (granted nothing else is touched) and cause a loop where the game never ends. The likely root cause lies within the generate food function inside of the food cpp, specifically the for loop that checks whether the random position collides with the previously generated food. If this did the check correctly, the issue would not exist. To properly debug this, I would suggest to first try and narrow down where the issue is occurring (just as I did when narrowing it down to the specific for loop in the generate food function). After that, the debugger should be used and the variable for the special character should be monitored. Print statements can also be used to take a quick peek at unusual behaviour.

2. **[6 marks]** Does the Snake Game cause memory leak? If yes, provide a digest of the memory profiling report and identify the possible root cause of the memory leakage.

The snake game does *not* cause any memory leak. In lines 23-29 of the project cpp, pointers for game mechanics, player, and food classes are created. Later, in lines 62-64, they are dynamically allocated on the heap using the "new" keyword. To counteract this, the correct implementation of the cleanup routine is inserted where each of the three pointers have been dealt with using the delete key word in lines 174-179 inside of the project cpp.

Part IV: Your Own Collaboration Experience (Ungraded)

1. Tell us about your experience in your first collaborated software development through this project – what was working and what wasn't. If you are a one-person team, tell us what you think may work better if you had a second collaborator working with you.

All things considered; the experience gained from working on the project was valuable. Something that we really liked was how the lecture content was directly applicable to the project. This forces students to be attentive, and then activate active recall by attempting similar yet different scenarios on their own projects. In a sense, this project simulated a work environment where you are given a task to tackle in a team. Therefore, not only did our understanding of lecture content increase, communication, problem solving, and time management skills were honed and tested. One thing that I feel could have been done slightly differently is how the project is split up between partner A and B. One partner becomes a specialist of the food class without knowing too much about the player class, and vice versa. Since this is a learning setting, it would make sense to find a way to create iterations in which mostly everything is being thoroughly understood by both partners without making it too disorganized/random.