COMPENG 2SH4 Project – Peer Evaluation

Your Team Members __Malek Abouelkhair__ __Mohamed Alkouni_
Team Members Evaluated ___Ahmed_____ ___Tharshigan_____
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.

   Positive features: *The code that the evaluated team wrote is very sensible code structure modularization and aligns with the OOD principles. The header files of each object is quite identical to all the UML diagrams provided in the project manual instructions with the classes written in the right structure in the right files and is very easy to comprehend the behaviour of each object since the code is written in variables with certain naming conventions and structures that make it clear how different object interact with each other. The code contains functionalities within each object, to show that each is made to independently manage and perform a distinct set of duties.*

   Negative features: *Unclear of any debugging techniques or limited error handling strategies that have been used in the code, other than that the code is pretty well structured.*

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.

   Positive features: *yes, it is very easy to determine object interactions functionalities as each functionality code is written under the right class with correct use of parameters, and code optimization by employing efficient algorithms to maintain functionality as well as readability at the same time. For example in the {void generateFood(objPosArrayList* blockOff);} they proceeded to use single line if statements followed by else if statement for each condition but was all written in one line to maximize code optimization. They also used a Do-while loop to eliminate redundant iterations when for generating random coordinates for the food. Furthermore, in the move player function the updatecurrenthead() function is called to generate new head for the snake when a collision occurs by inserting the new head to the head of the list by using a pointer arrow operator. This is just one of many examples used in the code that shows the feasibility of object interaction in their code.*

   Negative features: *There are parts that may be improved. These include adding reducing the need for global dependencies to improve code maintainability and putting in place stronger error-handling procedures.*

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 of C++ OOD:

- *Objects simplify and improve the organization of code by hiding implementation details.*
- *Reusing code is supported by classes and objects, which facilitates system expansion and adaptation.*
- *Classes and interfaces give a higher level perspective of the system by abstracting away the underlying complexities.*

Cons of C++ OOD:

- *When compared to procedural code, the introduction of classes and objects may result in some performance and memory overhead.*

Pros of C design:

- *Particularly for smaller projects, procedural code is typically easier to understand and less complicated.*
- *Procedural code may be more efficient in terms of runtime performance for certain types of projects.*

Cons of C design:

- *For larger projects, procedural code may encounter scalability and extensibility problems.*

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.

   Yes this code offers sufficient use of comments and self documenting coding style. Their comments are provided through out the project extensively explaining functions logic and their decision making process, which helps in understanding the objective of each code better. The code also outlines the deigns process of object interactions and game design actions/how it's preformed.

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.

   Yes the code follows good indentation and sensible white spaces as each line/code preforming a related function are put together line by line and if not there's a white space in between, indentation of functions are present in loop/control flow statements such as for, while, do-while loops. Same thing is observed for newline formatting. No shortcomings were observed for this question.

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)

Yes, the snake game they provided offers a bug free experience and the game runs very smoothly with functions like checkSelfCollision and moveplayer showcased to operate smoothly with no bugs. However, when playing the game there's an observed delay and whitespace issue*. The way the insertHead operation and the updateCurrentHead method handle their respective inserts may be the cause of the observed delay and whitespace problem when the snake turns in the given code. The head is updated and instantly inserted at the start of the playerPosList whenever the snake changes direction. If the insertion and update processes are not synchronized, this could cause a brief delay and possibly add whitespace to the snake's tail. I would recommend them to try changing the movePlayer and updateCurrentHead methods to make sure that the head insertion and update happen simultaneously in order to fix this problem. Update the head inside of the insertHead method rather than updating it beforehand and inserting it separately. By doing so, synchronization may be preserved and observed. Something else I would add, would be in the draw screen where I would add a print statement indicating which button to use to exit the game for the user's experience.*

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.

   No, its doesn't cause memory leakage, 0 leaks for 0 total leaked bytes. Which means that they have deallocated the memory properly by using delete/free for the right objects created on the heap.

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.

   It was pretty good as me and my partner equally divided the work amongst ourselves and were able to hand in the assignment on time with no errors.