COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Faiqa Anjum Ishita Rana

Team Members Evaluated Shray Patel Siddh Patel

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.

   All the header files have a great deal of organization. They all begin with the crucial #define statements for other necessary files, as well as a clear easily understandable class name. Following the names of each header file, all of them followed a readily logic with a private and a public member section. The behaviours of each member are comprehensible, as the coders provided detailed comments on each role for every member throughout all the header files. For example, all constructor, getters, and setters were labelled.

   An improvement would be in the **objPosArrayList.h.** For someone potentially unfamiliar with the code, a brief comment on the members and functions of the class would be beneficial. Also, the **Food.h** and **GameMechs.h** header files are missing a destructor. Including a destructor is vital for preventing memory leakage, highlighting the significance of having both a constructor and a destructor in place.

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.

   After reading the main program loop, the structure of the code is excellent. The code begins with global initializations of a pointer to the designated classes. Following all the sub-functions of the main program, the needed concepts were implements throughout each function. Firstly with the **initialize()** and **cleanup(),** the coder remembered to include the new – with delete concept, in order to prevent as memory leakage. Each call to the members of the classes, were implement in the logical sub-functions, using a clear variable name. The coders also provided detailed comments on the reasoning behind the usage of every member call which made code easy to interpret.

   The coder forgot to include their debugging message. As a part of the project manual, it was asked to include debugging use key messages, for example with lose flag status. The coders must implement this to show the debugging – it could've been commented out.

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.

    - C++ has polymorphism, inheritance, abstraction, and encapsulation due to OOD while C does not.
    - C++ contains the uses of classes and objects and while C is only broken up into functions.
    - C++ is organized and structured making it is much easier to follow, while C is a lots of code all over the place and harder to follow.
    - C++ supports function overloading in classes while C does not have this feature.
    - The simplicity for the Project with the implementation of C++ makes the code much better, compared to C procedural in PPA3.

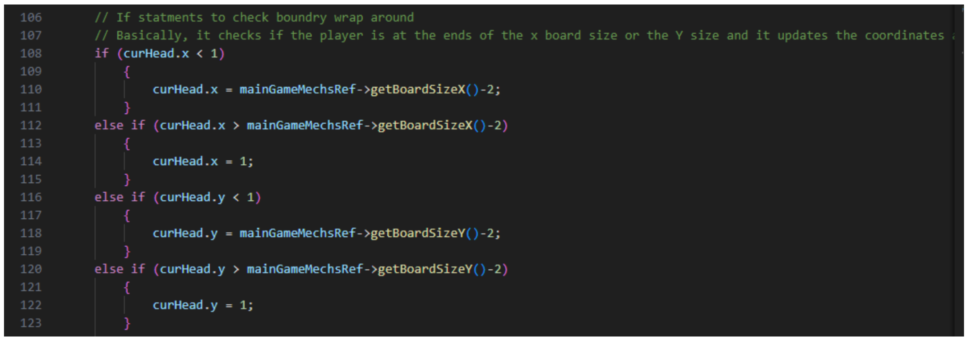
Part II: Code Quality

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 offered enough code comments to guide the viewer through understanding the structure and thought process. When mentioning any reference to a class it will specify which class it came from to direct the viewer to the file to understand why this reference is being implemented.
- When calling anything from a class, the comments also provided context as to why it was being utilized and what it helped to achieve.
- The only improvement that can be made Is including comments in the for loops as they are long processes and having comments to guide the viewer through the code may be beneficial.
- Overall, the coding quality depicted a high standard and effectively guided the viewer to gain a deep comprehension about implementation of the classes.

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 code follows good indentation and sensible whitespaces, everything is sectioned into appropriate processes and allows the viewer to understand. There were indentations as expected for the for loops, if statements, and switch cases. There were white spaces to break apart the code to separate the processes. One recommendation would be to improve the indentation for the boundary wrap if statements in the Player.cpp file on lines 106-123 as the curly brackets and the action being executed in the if statements have an extra indentation.

- 

```
// If statments to check boundry wrap around
// Basically, it checks if the player is at the ends of the x board size or the Y size and it updates the coordinates
if (curHead.x < 1)
    {
        curHead.x = mainGameMechsRef->getBoardSizeX()-2;
    }
else if (curHead.x > mainGameMechsRef->getBoardSizeX()-2)
    {
        curHead.x = 1;
    }
else if (curHead.y < 1)
    {
        curHead.y = mainGameMechsRef->getBoardSizeY()-2;
    }
else if (curHead.y > mainGameMechsRef->getBoardSizeY()-2)
    {
        curHead.y = 1;
    }
```

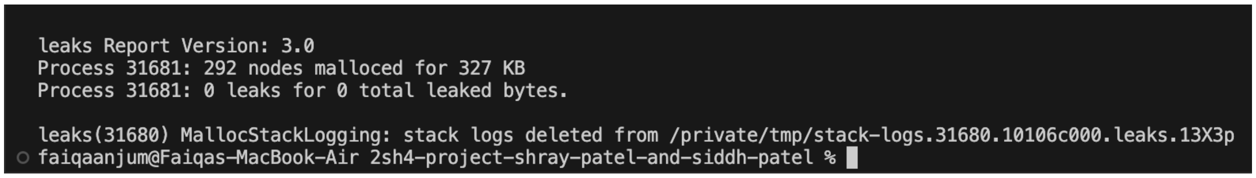
Part III: Quick Functional Evaluation

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)

- There were no buggy features detected through the code. The game ran smoothly, and the snake grew in length when it ate the food, this signifies that the code for collision detection works properly (as expected from inspection and analysis of the code), and the snake effectively commits suicide when it hits itself. There was no lagging, the snake did not impact the borders and it stayed within bounds signifying that the wrap-around method worked properly.

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 causes no memory leakage. The coders implements the correct destructors, along with the constructors, and always included a **DELETE**, when implementing a **NEW**.



```
leaks Report Version: 3.0
Process 31681: 292 nodes malloced for 327 KB
Process 31681: 0 leaks for 0 total leaked bytes.

leaks(31680) MallocStackLogging: stack logs deleted from /private/tmp/stack-logs.31680.10106c000.leaks.13X3p
○ faiqaanjum@Faiqas-MacBook-Air 2sh4-project-shray-patel-and-siddh-patel %
```

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.

- What worked for us was coming prepared to meetings to work on the code in parallel (especially iterations 1 and 2), so that we have a deep understanding of how to complete the iteration and can explain our though process and code implementation to each other as we go. Coming prepared to meetings would help us progress faster since we wouldn't waste each other's time to watch videos or reading through the manual for the first time.

- A key player to our success was setting mini-goals for each other, such as having half of this iteration done by this day.
- Overall, this was an enjoyable experience and emphasized the importance of communication and team collaboration to efficiently produce a final product!