**COMPENG 2SH4 Project – Peer Evaluation**

Your Team Members Mohamad Hadi karthic vasan sankar

Team Members Evaluated Ryan b qusay Q

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 proper definition and utilize “ifndef” functionality to ensure it is defined only once,this is found in all header files like player.h gamemechanics.h,food.h(custom file). CPPfiles interact properly with header files, private class functions are properly approached by using functions and havent been tried to access directly. functions are hooked to meaningful and correct pointers in main file. Moreover, I can easily understand whats happening inside each cpp file by going through its respective header file. There is no con with the modularization of the code, and all the header files are organized.

1. **[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.

We can easily interpret the code in main program loop and clearly understand what is going on, the flow of program is coherent. Going through the main file, we see that everything is properly organized and looks neat. For a huge portion of the main file there are comments provided which is a bonus. Additionally, the main function doesnt have much code inside of it because the code is done inside of other files. In main the team essentially called on the different game files and kept it as concise as possible.

**[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.

cons: hard for beginners to understand

**pros:**

* Makes it easier to debug in chunks in comparison to C
* C++ provides more modularity inside the overall code and makes it much more easier to read and understand.
* Advantageous while altering code
* Allows us to reuse Class in other parts of the code.With OOD we could build off class for future iterations instead of copy and pasting everything.
* Creation and destruction are alot more efficient in C++ than C because of the entire Constructor and destructor format.

**Cons**

* A beginner would not be able to understand the OOD format compared to the way we coded PPA3 in C due to the initial learning curve between C and C++.
* Initially requires more time to understand and implement PPA3 using OOD.

**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.

There are comments found inside all the files and they explain the functionality of almost all the Files, However objPosArrayList doesn't have any comments implemented which means someone who isn’t in this class would have a hard time understanding what is being done in this file. Furthermore, We can see the use of comments inside the main file, however comments are missing for the draw function and process following. There is no comments explaining printing procedure, score presentation and uninitialization

the comments are mostly sufficient, however, we would personally add more details and add comments where ever possible (Not overdoing it) to ensure someone who is a new programmer would be able to understand what’s happening throughout the code by the comments.

1. **[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.

Their code follows good programing practice, has proper indentation, utilizes new line function while displaying messages to ensure everything is legible and there is essentially no actual improvement required because the overall code is very fluent and organized.

There is some minor inconsistency in {} braces positioning in 2 functions and the player constructor function has an odd function input present inside most parts of the code is properly spaced.

However, overall there is no actual criticism that we can give since everything is done nicely.

**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)

the game provides a smooth bug free gameplay, however there are a few issues we encountered.

1. if we try to end the game without suicide condition no keys will terminate the game
2. after the game ends by suicide condition screen displays score and message: ”press any key to shutdown” but no keys happen to terminate the application except enter key

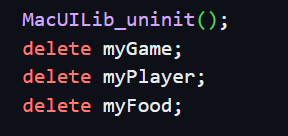
other than this there is no other bugs present which could affect functionality

these bugs can be avoided by using proper quit game condition functions to set exit flag and get input function to call termination function.

these bugs can be identified by comparing gameplay expectation and experience, since these functionality doesnt involve any specific calculation it can be only identified by executing and through testing manually.

1. **[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.

We did not find memory leakage in the other teams code. Going through each of the code files thoroughly, we can see that proper memory management was implemented. Moreover, after analyzing Gamemechs, Player, and objPosArrayList we can see that inside each file memory allocation and deallocation was used correctly. Conclusively, at the end of the main project file (CleanUp function) we see the use of the delete methods to ensure there is no memory leakage, as referred to the picture below.



**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.

collaborating was a good experience and it made implementation alot easier and time efficient.completing this project individually would require a lot more effort and time so working in team was much better than completing this project alone, however we feel that a team of three would be much better than a team of 2,we liked the dynamics of of working together and discovering eachother’s strength and weaknesses.