COMPENG 2SH4 Project – Peer Evaluatio

Your Team Members Mohammed Zeeshan Ansari (400480720) & Aditya Hura (400480022)

Team Members Evaluated Abrar and Daniel Real__________________

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.

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.

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.

Ans 1.

Solely using the header files of each object, you can understand the gist of the function or declared variable. Someone who isnt familiar with OOP though, that person may get an idea but would have no idea what public or private mean, or what declaration of objects would mean inside a header file.

Pros:

- Easy to read.
- Appropriate names make it easier to know the purpose in a glance. (How is a different question).

Cons:

- No comments in header files
- To a person with only basic knowledge of C, * would mean nothing let alone ->. If only one header file had briefly explained why -> and not .

Ans 2.

Yes it's very well commented out, super easy to understand the reason for everything. No real cons to be honest.

Pros:

- Very well commented. A person with basic knowledge of C would have an idea of everything that happens and why it happens (given he has understood the functions inside the classes).
- The Object declarations, print statements and everything are segregated from everything else. Like elements are together and makes it easier to understand.
- Use of MacUILib and not std::cout is also a huge plus point, because it follows the same syntax and principles as print statements in C.
- Declaration of global pointers is also appropriately done.

Cons:

No real cons if I'm being honest as far as main program's commenting goes.

Ans 3. There are many pros of using OOP over procedural programming like we did in PPAs. Let's go over a few that I could really tell when going from iteration to iteration.

Pros:

- Encapsulation of functions into classes provides a modular structure, improving code organization and maintenance. It makes the code easy to upgrade later on.
- Classes and objects facilitate code reuse by using composition which implements the "has-a" appraoch. Essentially, each class can have an instance of another class inside it and refers to that other class.
- Readability: Well-defined classes and methods enhance code readability, making it easier to understand the relationships and responsibilities of different components.

Cons:

Students unfamiliar with OOP principles may face a learning curve when understanding or modifying the code. Mainly because the references and pointers may be too much to comprehend.

Another possibility is creation of too many classes. Just over complexifies it.

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.

Ans1.

Comments are included in the provided code, they are definitely sufficient as far as learning the purpose of functions or a loop or a reference goes. I would definitely add doxygen comments as well (I would have if I didn't run out of time on my own project, which is unfortunate). Doxygen comments have tags, which can be used to explain more on classes and the return values of functions and where it would be used and more. Definitely not a requirement though, I'm just being nitpicky here. For Self-documenting purposes the comments provided are great. One more thing is that the header files can explain the parameters that are used inside of a function.

While the main code is well-commented, the header files lack that same level of detailed explanation. Introducing comments in header files, linked to relevant code blocks, would bridge this gap. Then again, the function naming makes it easy to understand what each function does.

The only part which I feel *needs* commenting is the gamemechs class, there's 0 comments explaining the generatefood function which could be a handful to understand to some people.

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.

Ans2.

Yes the code follows good indentation principles and there's decent whitespace left for better readability.

Good coding practice, I mean to say the amount of "whitespace" they have to differentiate between loops and conditional statements are good. It makes the code much more readable.

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)

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.

Ans 1. Yes, it is smooth and bug free. The game flashes too much in my opinion, as in the border and everything. The cause for this is many print statements, I would just put all print statements in a string or something and print that out once. Debugging approaches I can't say much, they've done a great job at implementing the game and it seems to run fine.

Ans 2. Memory leaks there seem to be none reported by drmemory. Although the number of uninitialised accesses is absurd (I got 3376 uninitialised accesses with 17 unique errors). Causes reported by drmemory trace back to the player's updateplayerdirection(), gamemechs get_input(), runlogic and main. This isn't particularly a cause for concern in this course. Might be later on, and I'm ignoring uninitilaised accesses from macuilib and minGW.

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.