COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Chehan Marakawatte_ Allie Launder

Team Members Evaluated Marryam_ Aadeeba

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 header files are organized, and methods are separated into respective groups for constructors, destructors, and more. All of the methods are named clearly for reader understanding. The code is missing methods for "check food consumption" and "increase player length" in the header files which would be preferable for a more object oriented approach.

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.

The code implements the "check food consumption" and "increase player length" logic in a procedural manner in the RunLogic function of project.cpp. This logic should be done in an object oriented manner and should be implemented into methods in Player.h or at least be procedurally coded into the movePlayer method. It should not be procedurally coded into the RunLogic function. Additionally, the suicide check is done in the project.cpp file when it is supposed to be in the movePlayer method in the Player.cpp file.

The MacUILibrary_ClearScreen() function was taken out of the cleanup function in order to print custom end-game messages in the DrawScreen function which isn't necessarily a bad design, but could alternatively be implemented in the cleanup function.

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:

- Easier to edit
- More organized
- Easier to interpret from outsider perspective
- Easier to test while creating code
    - Can test individual methods before proceeding

- The created classes can easily be reused if some functionality was required in a different code being written
- There is more security with private and protected data members

Cons:

- More code is required which means it will take longer to both write and read through
- Designing the code implementation is complex, thinking of how the different methods will work together
- It's more difficult to access some data because it may be private or protected
    - Specified methods must be created to access this data

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 implements good commenting practises generally. Blocks of code are generally modularized, and each block has a respective comment. Anything that may be confusing is provided with a comment for better understanding. All comments are clear and concise.

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 is made very readable. There are no large blocks of code and appropriate spaces to improve readability. Long function or method implementations are broken up by empty new lines to prevent having too much code grouped together.

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 is a lag between when the snake collects food and when the snake's length increases. This may be because the logic for increasing player length and food collection were both in RunLogic in the project.cpp file. If it had been made in a more object oriented manner with the use of their own methods, the design would be more efficient. Besides the one buggy behaviour, everything else in the program runs properly without bugginess, since the coding was implemented with an object oriented approach.

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.

There is absolutely no memory leak. There is proper implementation of destructors in the code. In GameMechs.cpp, there was nothing created on the heap, so there was nothing in the destructor method. In objPosArrayList.cpp, an array was created on the heap which was deleted properly in a manner that deletes all parts of the array in the destructor method. In Player.cpp, an array of player positions was created on the heap, so it was destroyed in the destructor.

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.

The project was good since we usually worked on the code together and revised each other's code implementation. GitHub had issues when pulling code, which was quite difficult when trying to collaborate. To get code from a partner, one partner would have to push to GitHub and the other would have to copy-paste the code from GitHub into their local files. Working with more than one person also made it much easier to catch errors and think of a different approach to some implementations if the first one didn't work out. Being able to look at the code from two different perspectives was very beneficial.