COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Ryan Farekh and Vraj Patel

Team Members Evaluated Christ Pan and Ryan Li

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.

After analyzing the header files of each object, which includes Player, Food, ObjPosArrayList, it is easy to interpret the possible behaviours of the objects in the program. The name of such getter methods is very straight forward and just by reading them, it is easy to interpret what the specific method is for.

Positive Features:

- Private Members

All private members are initialized correctly from interpretation. This means that the “important” variables are outside access to anyone else which promotes encapsulation.

- Purpose

Every header file has a very clear purpose which can be easily analyzed by taking time to read the getter methods of the respective class. For example, in Player class, it is obvious what the class is responsible for.

- Header Guards

The use of header guards was implemented perfectly in the sense that it prevents multiple inclusions of the same header file in a translation unit.

Negative Features:

- Comments

Adding comments would be better especially since it helps other developers understand on how to use the class and on what some of the methods are intended to do.

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 checking the main logic in the main program loop, it is fairly easy to interpret how the objects interact with each other in the program logic through code. Obviously, most of the code (logic) is being done “under the hood” in the separate classes, but just by interpreting the names of the methods called, it is easy to understand on how to objects interact with each other.

Positive Features

- Good Implementation

Within most aspects of the main logic in the main program loop was done correctly, in the sense that OOD was used. For example, in runLogic() function there are only 3 methods being used, which is a positive since is increasing simplicity.

Negative Features

- Global Scope

Within the global scope, there are multiple extra variables declared. One of the pros of OOD is to reduce global variables because of their side effects but within the code, there are a significant amount of globally declared variables which simply defeats the purpose of OOD. Instead, they should be calling on the methods implemented instead of declaring more variables globally.

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

- Simplicity

Everything scaling from Readability to simple logic for some functions is simpler. In Procedural the aim is to construct everything into 1 big file but in OOD, big problems can be broken down and implemented onto simpler problems like this project here. Also, to an outside programmer, the code is very simple to read and would know what the code is doing.

- Global Scope

In PPA3, there were a considerable number of global variables which isn't good. OOD gets rids of the global variables which makes the program more flexible and increases modularity.

- Maintenance

Within Procedural Programming, if there is a code snippet which needs to be changed, it can affect the entire file, and with bigger projects changing a small section of code means to change other parts of the code (which can be 1000+ lines). OOD reduces this annoyance by a drastic measure. If there is some code which needs to be changed, instead changing all the lines in a very big main.cpp file, the changes would only be done within the class itself.

Cons

- Learning Curve/Complexity

For beginner programmers, the idea of OOD can be very challenging since they are changing the entire approach to a project. Beginner Programmers are more accustomed to procedural programming since that is what they have been practicing since their career as a student so a change in Design can be hard to grasp at first.

- Increased Compilation Times

With bigger and bigger projects, there would be more classes and more lines of code. This means that the program would have to jump between a lot of classes which may increase compilation time of the program.

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.

- Their code is understandable, and comments are clear and concise when they appear. I would encourage the team to add more comments explaining the functionality of certain references, specifically within the project.cpp file, to make it easier for future programmers to understand what is being executed.

- Although the titles of each class member are already quite self-explanatory, I would encourage the team to add more comments in the .h header files as well to clearly explain all the available class members and their functionality.

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 very good indentation habits and with no excessive white spaces and deploys proper new line formatting for better readability. The code is very well-organized and has no shortcomings in terms of formatting. The code takes up a sensible amount of space with no excessive scrolling necessary. All functions are deployed neatly, and in a sensible order.

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)

    - This game offers a smooth and straight forward user experience. The game runs without bugs or delays. I noticed especially that with some other groups projects, the game was flashing and the snake would lag instantaneously when consuming food, this is not the case with this group's project, the wrap-around features, consumption features, and suicide detection work smoothly without bugs, and the proper end-game messages are displayed at the right time without lagging.

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.

    - This team's snake game causes no memory leaks. An instance of their Game Mechanics Class as well as an instance of their Player Class was initiated on the heap and were properly deleted in the cleanup routine allowing for no memory leaks.

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.

Vraj: In previous projects, I've had experience in collaborative software development through hackathons and courses. This project was my first time working collaboratively while implementing Object-Oriented Design (OOD). Transitioning from procedural to OOD was initially challenging, but with the help of Dr. Chen's tutorials, the concepts became clearer. The most challenging part was during iteration 3, specifically feature 1, where changing objPos to ObjPosArrayList had a significant impact on the existing code. I've known my partner, Ryan, since our first year and he exceeded my expectations in this project. His strong work ethic, proactive communication, and early completion of iterations were impressive. However, I believe he could enhance his feedback provision, especially in evaluating the code quality and potential improvements.

Ryan:
Collaborating with Vraj on this coding project marked a transition into Object-Oriented Design, a departure from my previous procedural approach. We encountered numerous challenges, especially during iteration 3, but we collaborated and made it work in the end. Vraj worked hard and collaborated efficiently, completing his tasks before our designated deadlines allowing for a smooth collaborative experience. I have known Vraj for quite some time now and I knew it would be a pleasure working with him, as I have not had a bad experience with him throughout our numerous collaborations. He proved again that he is a team player, and willing to put in the effort to make it work, even when the project becomes 3rd or 4th on the list of priorities with everything else going on in different classes. Overall, this project was a positive experience and presented me with challenges that will help me in future similar endeavours, and Vraj made it that much more of a positive experience.