COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Charanjit Rikhi Matthew Gobeil-Chianchai

Team Members Evaluated Matthew Rakic James Cameron

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.

Looking at the header files of each object you can easily interpret the purpose of each. This group decided to create the food object, within this, functions associated with Food are found such as food constructor and deconstructor. In addition to generate food and get food features. The other objects like game mechs, object position array list, and player also have the associated functions. To explain, the player object holds all functions related to the player, the game mechs object holds all functions related to the mechanics of the game, and the object position array list object holds all function related to the object position array list. This is a significant positive compared to placing all our functions within the same file, this keeps the code very organized, independent in terms of their separate functions, and makes it easier to implement changes due to separation of objects and readability. Though a downside of using OOD is that it may lead to over complication and inefficiency compared to more simplistic designs.

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.

From the main program loop, it can be identified how some of the objects such as `GameMechs`, `Player`, `myFood`, and `objPosArrayList` are used to interact with one another in the program. For example, it can be observed that there are pointers such as `myGM`, `myPlayer`, and `myFood` pointing to their respective objects. The interaction can be seen through `myGM -> getInput()`, which is an input handling mechanism. A major positive of object interaction can be seen in the initialize void function, where objects are created using the new keyword, the Player object is initialized to `myGM`, and `myFood` objects. Furthermore, Food also has references to `myGM`. With the objects interact with one another, it is unavoidable if there are not any dependencies among the objects. In this case it would be beneficial to define the dependencies through comments. Such dependencies can also lead to logical errors that are faced during error handling.

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.

There are many pros to using OOD, such as a more organized program leading to easier debugging, the ability to reuse code with inheritance and being able to have a more flexible program through polymorphism. In terms of modularity, the code is organized into classes which improves the code readability thus making it easier to understand the interactions between the objects. Though some downsides to OOD are the increased complexity of the program, and the possibility of reduced efficiency compared to a more simplistic design.

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 offers sufficient comments in some areas such as in the initialize void function, where comments such as "board size" and player food item generation, explains the functionality of the following code. However, the code functionality, can be improved upon by explaining areas that are more complex. For example, in the `DrawScreen` void function, the for loops, that utilize looping parameters, along with `myGM` objects; can become complex to understand for a new viewer. I would improve this by commenting the usage of a function, and explaining complex lines of code, where possible objects are referenced.

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.

This code has great indentation, everything is intended logically, and the program makes good use of white spaces. Though it is of note, that their usage of white spaces is slightly inconstant, though this is not a major problem. Another minor formatting critique is that, not a consistent method is chosen for the usage of the curly brackets. For example, sometimes a curly bracket is placed in front of the line of code, and sometimes on the next line. Again, this is nothing to lose marks over, it is just a hint of inconstancy. Their usage of '\n' is good, making the on-screen information readable, and easy to understand.

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 most prominent error when running the code is the displayed score in the terminal window. Through testing it was found that the displayed score does not begin at zero, and every time the code is run at a different instance, a different number is displayed for the score. Meaning that upon the initialization of the game and output in the terminal, a value is already present for the score before user interaction. However, whilst playing the game, the score begins to be incremented correctly. Looking through the code it was identified that this could be caused by the score not being initialized to zero. This would explain why the score still increments correctly and why the number changes every time the exe file is ran. To catch this error in the future, I would recommend that the other team utilizes incremental engineering practices and approaches.

Another implementation that was noticed whilst playing the game was that there was a slight delay from when the snake consumes the food and the length of the snake increments. A possible reason for this delay is the insertTail function, within the function we found that the position in inserted into the array list before the array list size in incremented. We also found that the position was inserted without the use of the setObjPos function. It was beyond our comprehension due to the different technical approaches taken in this code, to understand the reason this program is still able to run, however the program runs nonetheless.

A user accommodation/preference error that was found was that after the snake collides in its body and the game is lost, the game lost messages displayed in the terminal at the bottom of the board appear for a short time, until the exit screen is displayed automatically. The game lost message is very easy to miss, and it can be concluded that the message should have been added to the exit message instead of being displayed before the screen is cleared. Improving this helps user interface and game experience.

It is of note that the following group used Linux, and our group utilized windows, this possibly could have created some of the errors with their program, but we followed the procedure for the switch in operating systems, so this is unlikely.

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.

Through the usage of Dr. Memory, it was found that their code exhibited zero bytes of leakage. All allocated memory is properly deallocated with the use of “new” and “delete”. The usage of destructors after constructors can deallocate the memory used for functions such as array lists.

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.

Charanjit Rikhi:

Throughout this project, the collaboration between the both of us was good. Methods to complete the code, included in person and online meetings helped accomplish the project. Along with other communication to designate tasks and work. Through the help of another collaborator, I found that debugging, and problem solving was much more productive as another member was able to provide their insight and solution. Division of the work was fair, and we both worked together to solve errors we were running into. Overall, this project helped my learning experience, and communication skills, as we both worked on the same project!

Matthew Gobeil-Chianchai:

I found that throughout the duration of this project, I found that the inclusion of a partner allowed us to reduce our total workload, and greatly improve our learning experience allowing another perspective on our code to fill in any gaps in our knowledge. My partner for the project was very diligent, hardworking, and easy to work with. She was always quick to respond to any messages concerning our project, and together we were able to pace our selves in the creation of our program, in spite of our minimal availability.