COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Mani Nabovati Ryan Seyedan

Team Members Evaluated Jiayi Yang Xiang Li

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 reviewing the header files of each object, overall, I believe the behaviors of each can be interpreted easily. All of the functions in each of the classes are named appropriately according to its specific function and the team leaves no room for ambiguous interpretation. For example, the objPosArrayList class offers methods like insertHead, insertTail, removeHead, and removeTail, which are clear names for their specific function if read further into via the cpp file. This design allows for easy understanding and manipulation of spatial elements in the game. The team utilizes whitespace well in all the header files, spacing out each member function or data member appropriately to improve readability. One thing I believe that could be improved on is organizing the function members according to getters and setters, through placements or comments. This would allow collaborative coding to occur more seamlessly, giving both yourself and teammates a clearer idea of what function does. Furthermore, it makes cross checking function names and existence more easy via the header file

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.

Overall, the main program loop in the provided code showcases a well-structured interaction between various objects, all of which adheres to the principles of Object-Oriented Design discussed through the semester. Instances of the Player and GameMechs object were initialized in the global scope and named appropriately such that whenever called, it was easy to understand which class was being accessed. As a result, it is easy to understand that myPlayer updates and moves based on input processed by myGM, which handles all the games logic. The team also

methodically utilizes object interaction. This is highlighted in the runLogic function where myPlayer relies on myGM to process inputs and update its state, demonstrating the relationship between each other. These interactions are easy to follow as a result of the well named function members. Another positive is that the team doesn't utilize any hard coded values, rather uses functions from myGM like getBoarderSizeX and getBoarderSizeY to define loop conditions within drawScreen.

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.

- The OOD approach brings encapsulation, where data and functions are bundled within classes, enhancing data integrity and security.
- C++ is a much more modular approach vs the C procedural design. This object oriented design gives the user the ability to break down complex behaviors in specific classes with its respective file.
- C++ OOD approach can be easily maintained if utilized with intelligent naming convention due to scanability.
- C procedure might be more straightforward for smaller scale projects while C++ OOD might be better suited for larger scale projects like the snake game.
- C procedural design is not very scannable or readable, especially when the amount of code and complexity becomes large scale.
- C procedural design lacks code reusability through inheritance and polymorphism which is present in C++ OOD.
- C has structures (struct) and they do not support access specifiers like public and private while C++ does.
- All members of a struct in C are public by default and encapsulation can be mimicked by defining functions that operate on these structures and only exposing what is necessary in the header files which can be tedious.

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.

Overall, this group did an excellent job of thoroughly commenting on their code which made their logic and workflow very clear. They provided an explanation for their logic in the project file and also added informative comments in their class files. Also, the code generally deploys a

sufficient self-documenting style for variable names. For instance, the variable named myDir clearly contains the player symbols movement direction.

However, a particular shortcoming is that they did not delete the pre-existing instructional comments from the project template which do not assist in code readability or the reader's understanding. Also, the group could have added function header comments to give an overview of the function behavior and purpose. Furthermore, this group included their food methods implementation in the GameMechs class. However, they should have included comments to distinguish the methods pertaining to the food item for greater organization and readability.

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.

Throughout their code, this group uses good indentation. This is specifically seen in loops as well as conditional statements that are effectively indented. This helps the reader understand the nested structures and the order of execution. Specifically, the code uses a mix of 4-space and 2-space indentation. They are consistent with this indentation format and uniformly apply it throughout the code. Furthermore, white spaces are used effectively between keywords, operators, and function parameters, contributing to readability. The white spaces between different lines also help separate the different sections of code. There are some areas of the code, specifically, in the GameMechs.cpp file where the newline formatting is inconsistent between functions. This can be simply improved by ensuring that the line spacing between the functions is kept consistent throughout the code (For example 1 line of space between each function).

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 snake game runs smoothly and shows no bugs. This is also supported by the correct implementation of the classes and calling of the methods in the main project file.

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.

The snake game has 0 bytes of memory leakage as shown in the image of the drmemory report screenshot below. This is further supported by examining the code. The heap instances declared at the beginning of the main project file: myGM (from the GameMechs class) and myPlayer (from the Player class) are both deleted at the end of the game program loop in the CleanUp function. Moreover, in the objPosArrayList class implementation there is also a destructor used

to deallocate the array on the heap. Overall, the correct usage of delete throughout the code ensures that there is no memory leakage.

```
ERRORS FOUND:
      0 unique,     0 total unaddressable access(es)
     15 unique,   176 total uninitialized access(es)
      0 unique,     0 total invalid heap argument(s)
      0 unique,     0 total GDI usage error(s)
      0 unique,     0 total handle leak(s)
      0 unique,     0 total warning(s)
      0 unique,     0 total,      0 byte(s) of leak(s)
      0 unique,     0 total,      0 byte(s) of possible leak(s)
ERRORS IGNORED:
     17 potential error(s) (suspected false positives)
         (details: C:\Users\Mani\Downloads\DrMemory-Windows-2.5.0\DrMemory-Windows-2.5.0\drmemory\logs\DrMemory-Project.exe.20236.000\potential_errors.txt)
     30 unique,    30 total,   7144 byte(s) of still-reachable allocation(s)
         (re-run with "-show_reachable" for details)
Details: C:\Users\Mani\Downloads\DrMemory-Windows-2.5.0\DrMemory-Windows-2.5.0\drmemory\logs\DrMemory-Project.exe.20236.000\results.txt
```

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.**

Overall, the collaborative software development experience was very convenient and went smoothly. Our group did a great job of setting defined roles and responsibilities for each person. We also planned out our time to have iterations done by certain dates to stay on track and comfortably meet the deadline. Furthermore, github made the experience very convenient as we were able to use the repository to constantly update our code and share our progress with each other. We had some issues with the merging requirements after git push and pull, however, we were able to troubleshoot these problems. This collaborative experience was invaluable as it taught me that when two individuals put their mind together to solve a project, deliverables could be completed faster.