COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Isha Patel + Maryam Ismail

Team Members Evaluated Alina L + Hanna N

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, the possible behaviour of the objects can be interpreted from the inclusion of different section within the classes such as private and public following encapsulation principles. Within the header files, they have successfully included the files the program is interacting with. An example of this is within ObjPosArrayList.h they have included ObjPos.h proving to be a well-structured program and allowing programmers to understand how each file interacts with one another. One thing to improve on is being more descriptive with their variable names such as 'fPos' could be named 'FoodPos' in the gameMechs.h file to help understand the purpose of each variable and enhance readability.

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.**

   In the main program loop, the code is structured in a modular way where each function has a specific purpose and it is distinctly separated from one another. An example of this is the GetInput(), where the program receives input from the user, and Drawscreen() where the program prints out the drawings to the screen. The objects interact with one another, which is seen where the Player and gameMechs object interact and the logic is updated through the player object for functions such as movePlayer or updatePlayerDir. Furthermore, the main game loop is structured so that it follows a pattern of getting input, run logic, draw screen, cleanup etc. A suggestion for improvement is getting rid of unused variables such as 'Inputmain' which is declared but never used throughout the main logic. Removing such variables will declutter the code and prevent potential confusion regarding their purpose.

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

- Modularity: it is easier to encapsulates data through objects rather than having it all in one file
    - Having separate files for the object position, player and the main program loop helps keep the program more organized
- makes the code more organized and easier to follow/understand
    - Trying to implement it in a procedural way would have been complicated and made it harder to develop the code and handle errors
- Reusability: the classes can be reused
- Easier to maintain: changing a certain part of code will not affect the others
    - It was easier to debug in the project than the PPA3 because we do not have a lot of code in one place and it is structured and organized, which allowed us to pinpoint errors within our code

Cons

- It was difficult trying to manage all of the file and know where each function went. So there was a learning curve to doing this project in C++
- Compiling: it took longer to compile the code because of the amount features used within the project
- Working with memory in C++ and managing the memory allocation was more challenging than in C procedural because of the creation of destructors for deallocating memory

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 program uses sufficient comments in all files used. In particular, they have used detailed comments in almost all method declarations, giving a further explanation as to what their functionality is. This enhances the programs readability and helps guide the developers in understanding the methods. Before every block of code, they give a detailed explanation as to what that code is intended to do provided a clear and distinctive roadmap. This gives any newcomers who are reading the code for the first time adequate information to understand the code.

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 good indentation and naming convention. The variables have descriptive names in marts of the project such as 'BOARD_Height' or 'tempbody'. The code has very descriptive comments outline the purpose of each line and method and how they interact with other parts of the program. An example of this is within 'Runlogic()', there are comments outlining the sections of code where collision detection occurs and if food has been consumed. This allows us to understand the logic behind there code better and make it easier to follow. Within there Drawscreen(), they have implemented newline formatting to help create a clean display for the game, displaying the score and the position of the player and its body in the terminal.

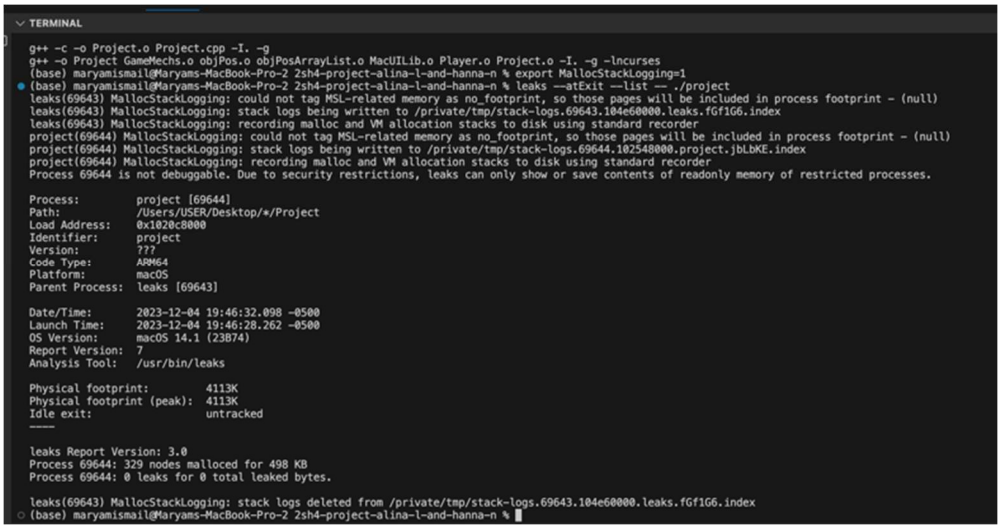
Part III: Quick Functional Evaluation

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)**

   Overall, the snake game offers a smooth, seamless, and enjoyable playing experience. The game is largely free of bugs when it comes to the food position and randomization though the game does become noticeably slower the more food you catch and the bigger the player becomes. This could be due to many factors including the intricacies of editing lists and processing data within the game's logic. Although not covered in the scope of 2SH4, as the developers advance in future coding, they should consider other algorithms better suitable for advanced data structures to enhance game responsiveness leading to a smoother playing experience.

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 code provided demonstrated an efficient approach to memory management confirmed by running Dr Memory. The analysis showed a result of 0 leaks for 0 total leaked bytes reported showing high levels of efficiency and precision when it comes to Dynamic Memory Allocation. The allocation of memory for the 'GameMech' and 'Player' objects during the initialization phase, accompanied with their deletion in the Cleanup function in the same order it was created, reflects a well-thought-out strategy.



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.**

   Having a collaborator on a project like this was very beneficial as it was easy to brainstorm ideas together which reduced the amounts of mistakes made. One difficulty we encountered was the tediousness of sharing code. Although it was recommended that each developer split the work, if someone wanted to make an edit, the other would have to match the exact same for the code to flow well. Collaborating In a team made the process more smooth, but there were some challenges such as trying to make changes to code and having to constantly be updating the code depending on the other partners changes. We had to become creative at finding avenues to share the code and being able to communicate with one another on our progress of the iterations.