COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Jeniishaa Bhagudeva, Aliyah Qayum

Team Members Evaluated Harsh Duba, Adam Smith

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.

   The code is written with OOD principles in mind. Each class has been encapsulated to have specific functions, thus improving code readability and behaviours of the objects. The function names suggest clear purposes, helping readers identify the possible interactions of the objects with each other throughout the program. A few negative features that are worth mentioning is that some global variables in the program 'GameMechs.h' such as 'foodPos' and 'foodList' could potentially be encapsulated in the 'GameMechs' class to reduce global scope. Also, some variables could be a bit more descriptive. For example, in 'Food.h', there are two variables 'bx' and 'by.' This could cause confusion to readers as there would be no indication of what these variables mean. To reduce this confusion and interaction of variables in latter parts of the program, it can be named to something more explicit like 'BoardSizeX' and 'BoardSizeY.'

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 name of the objects are clear and easy to interpret on what they are used for. Additional variables are created to check for collisions and food consumption that make the code easy to read to an outsider who is not familiar with the project. The functions in the main program are also kept short and concise while a majority of the code lies in its respective classes. This makes the main program extremely easy to read. One possible negative feature is that many of the objects defined are initialized to NULL at the beginning of the program. This may cause issues later down the line into the program if the objects are not redefined due to an error or bug. In the future, it is best to either not initialize them to NULL or even better add more checks to see whether the objects are NULL or not.

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.

    - C++ has polymorphism, inheritance, abstraction, and encapsulation due to OOD while C does not
    - C++ contains the uses of classes and objects and while C is only broken up into functions
    - C++ supports function overloading in classes while C does not have this feature

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.

    There are numerous comments to help readers understand the code functionality more efficiently. Every class and object has a concise line of code to summarize the purpose and helps readers build the connection between each object. The commenting for this project was done extremely well. One thing that could possibly be done a bit better was that some of the loops, specifically the nested loops, were hard to follow along and complex. Adding comments within these loops to outline structure and purpose would be beneficial for readability. Also, while comments exist for many variables, there were some variables that were not entirely clear such as 'printed' in Project.cpp. This caused confusion since there were many things being printed and it was difficult to keep track of what the variable was representing.

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 has good indentation with the code in the if statements, loops, and functions being indented to signify its association with the mentioned statements. There are a significant amount of white spaces but to improve readability, there can be more spaces between the if statements and the code that comes before and after it.

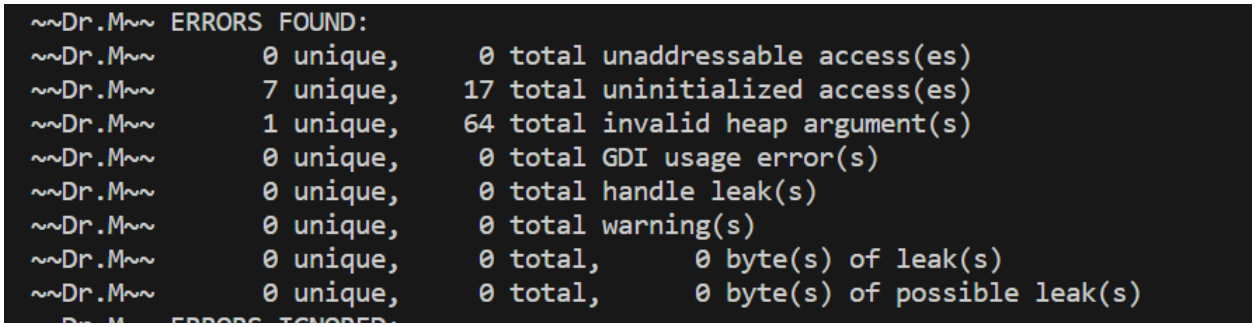
Part III: Quick Functional Evaluation

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 offers a smooth, bug-free playing experience. This game was executed flawlessly and no bugs were caught. There were numerous conditions for each of the 5 symbols, and all of them altered the score differently. On top, everytime any symbol was caught, the body of the snake increased by one. As mentioned, there was no memory leakage seen through the attached image below. During our development, challenges arose with wraparound conditions and player direction control, causing unexpected movements. Our initial code utilized numerous conditional statements and ++ logic for left + right, up + down movement. Recognizing the enumerated states at the code's outset, we leveraged them to enhance efficiency and resolve bugs. These refinements resulted in a flawless execution. Interestingly, a similar approach was observed in the implementation of the other team's code, proving that must have been what was causing bugs on our end.

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 are no memory leaks with this program. This is due to the team adding a deconstructor function for the classes that require them. All the objects placed on the heap were all deallocated resulting in 0 bytes of 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.

   Working on the Snake Game was enjoyable and challenging, highlighting the significance of Object-Oriented Programming. While PPAs focused on writing functions, the complexity of this game emphasized the efficiency gained by organizing functions into classes. Our first collaborative development was successful; we worked together to debug and address challenges. In the intricacies of the game, having a second set of eyes proved invaluable to catch potential errors. This experience underscored the practical application of programming concepts and teamwork in complex software development.