COMPENG 2SH4 Project – Peer Evaluation

Your Team Members: Nuha Fahad and Shanza Riaz

Team Members Evaluated : Yuxiang Chen and Nirmaan Patel

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.
   Positive features:
   - The header files are easy to read. It is clear that the design of the program follows an OOD approach.
   - The use of header files efficiently implements the main features in an OOP Design. For example, encapsulation is seen in the 'ObjPosArrayList.h' file. Private data members such as 'aList' and 'sizeList' are encapsulate within the class, this keeps the internal details of the class private from external users.
   - Also, the methods in the header files serve as a well-defined interface that interact with objects in the class. For example, in the Player class, the method increasePLayerLength(), increases the number of items in the 'objPosArrayList playerposList,' this method is cohesive and demonstrates how the function focuses on a single responsibility related to the player position length.

   Negative features:
   - The header files lack basic comments. This would make it difficult for other developers to understand and update the code.
   - It would also have been better if there were a separate food class. This would make GameMechs a simpler class and adhere to the Sigle responsibility Principle, where classes have a single responsibility and purpose. Overall, updating the design and introducing a new class (Food) would allow high cohesion and a more organized program.

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 main logic in the main program loop is easy to understand.
   Positive features:
   - The code is modular. It has different functions that handle different parts of the game. Every function in the project.cpp file controls a different aspect of the game. For example, 'DrawScreen' draws the game board as well as all player objects, while 'Initialize' sets up the game variables. This modularity makes the code easier to read, maintain and update.

- The objects also interact together smoothly, ‘myGM’ handles the game mechanics and ‘myPlayer’ handles player related game functions.

- The comments in the code file explain the purpose of functions and loops. This is helpful for other developers to understand different aspects of the program.

Negative features:
- The ‘GetInput’ function in the project.cpp file should be deleted entirely since it is redundant and performs no function.
- Also, since the ‘GetInput’ function is integrated into ‘myPlayer’ functions there should be an explanation on why this is better for the program.

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
    - Since each mechanism of the code is separated into individual files, one can easily determine exactly where to find a specific function or mechanism
    - Easier to collaborate as each programmer can work on different parts of the code without interference
    - Different files for each mechanism also allows for easier debugging, as issues can be isolated and identified
- Organization
    - Having multiple files for each mechanism prevents having one large, unorganized main file, which makes the code easier to interpret and understand
- Reusability
    - Easy to reuse a function or variable and incorporate into different mechanisms, avoiding code repetition and thus saving time
    - For example, board size was only created once in “project.cpp”, but other functions in different files were able to access the same number for different contexts

Cons

- Multiple files
    - Having multiple files open at once can be overwhelming and hard to keep track of each file location
- Complexity
    - It can be difficult to learn how to link and connect each file together

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 comments make the code very easy to understand and interpret as they are concise and clear. Additionally, there are a sufficient number of comments, and they are placed strategically at blocks of code that may be confusing or complex. For example, the comments were extremely helpful in the "genarateFood" function, although there was a lot of code and several for loops, the comments made it easy to determine which parts were for generating normal food and which were for special foods. Furthermore, for an effective self documenting coding style, the variables are named appropriately for their specific usage.

   One element they could improve is adding comments to some variable names, explaining their function in the program. More specifically, comments could be added to variables such as "objPos tempBody" and "objPos tempFood", to further clarify that they store current player and food information.

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, it is readable and well formatted. Additionally, newline formatting is appropriately placed, making if, while, and for loops easy to follow. The consistent use of whitespaces in for loop declaration, ie. for (int t = 0; t < 5; t++),
   makes each character very clear and straightforward to follow. Overall, the code is very clean and correctly formatted, leading to exceptional readability and organization.

   One aspect they could improve on is adding less space between the actual code and comments. In some areas, the comments were indented very far from the code, making the program look incohesive and scattered.

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)
   Yes, they snake game offers a smooth and bug free experience.

The player moves and changes direction upon command and 5 foods effectively generate randomly on board. The game did not glitch or crash while playing, and successfully terminated by either losing or pressing the exit key.

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 causes 0 bytes of memory leak.

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.

   Through this project, we learned the importance of communication in collaborative software development. By outlining what mechanisms each partner should work on, and setting informal deadlines, we were able to complete the project efficiently and in a timely manner. One thing that our group was struggling with was using GitHub, as we were not too familiar with the collaborative features of the software. Towards the end of the project we were getting errors and conflicts if we were working on the same file. Thus, to combat this issue only one person would work on a specific file at a time.