COMPENG 2SH4 Project – Peer Evaluation

Your Team Members ___Brooke__ ____Samantha_____

Team Members Evaluated ___Abaan____ ____Somesh_____

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.

Positives:

- Upon looking at the Player.h file, it is easy to understand the main behaviours of this class. Functions such as getPlayerPos(), updatePlayerDir(), and movePlayer() are intuitively named and their purpose in the game is clear.
- The GameMechs.h file also had intuitive naming, it is clear what getScore(), incrementScore() and getFoodPos() do upon being called.

Negatives:

- In the comments within Player.h, operations that check for food consumption and self collision are alluded to. However, this could be made clearer by declaring them as individual functions such as checkFoodConsumptio() and checkSelfCollision(). Otherwise, it suggests that procedural programming is used within the movePlayer() function.

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.

Positives:

- The logic for printing the snake body was easy to follow and concise. The use of 3 for loops ensured correct printing of the border, food and snake body.
- We liked how the current head position was being printed throughout the game. It is a good debugging technique to ensure that the snake is moving properly. Similarly, we also liked how they printed the direction of the snake movement.

Negatives:

- There were some inconsistencies in the DrawScreen function regarding printing characters. While printing the food and snake body, MacUILib was used, however when printing the '#' for the borders, 'cout' as used. We suggest remaining consistent throughout the code.

- There was some redundant code in the GetInput() and RunLogic() functions. In the GetInput(), the getInput function from GameMechs was called, however it did not need to be since it is already being called in the Player class, in the updatePlayerDir() function. Additionally, in the RunLogic() function, the exitFlag was being checked when it should be checked when the input is being collected in the updatePlayerDir() switch statement.

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:

- It is easier to organise and implement different behaviours of different objects in your code.
- As projects grow in complexity, procedural code may become harder to manage and maintain, especially in terms of debugging.
- In OOD, you have the ability to make variables private/protected in different classes, which is a benefit when coding large scale projects.
- OOD can help avoid redundancy within features because of inheritance.

Cons:

- Using the C++ OOD approach creates more files to keep track of such as class declarations and header files. These files must be correctly included in each file in which it is implemented.
- There's a larger learning curve when working with C++ OOD as it requires these extra files, and you need to call functions using the dot or -> operator.

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.

Comments are very limited through the code. In the logic running functions/sections of the code, there were no comments to explain the functionality of the code. The majority of comments were found in the header files; however, they were more so to-do lists rather than code explanation. This is not very helpful to someone trying to the read and understand the code. An area of improvement would be to add more comments in the bulk areas of the code, specifically function calls, switch cases, loops etc.

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 adequate indentation that makes it easy to read and examine. In select regions, extra white space was included that spaces the code out too much and makes the code more confusing to follow. This can be observed around lines 136-170 in the DrawScreen() function within Project.cpp. For example, 6 lines of code was used for a 1 line function, which is a excessive use of space.

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)

While the game is running, it runs smoothly and is nice to play however, we have experienced a glitch that causes the game to freeze and stop running. We believe this is a sporadic bug because it does not appear to follow a predictable pattern. Because it happens at random times, it is hard to pinpoint a specific root cause however a common occurrence of this bug is when multiple threads or processes are accessing shared resources concurrently, which results in a race condition. This is when the outcome of the program relies on the order in which two different operations are executed. This can cause sporadic and unpredictable behaviour.

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.

No, the game does not cause memory leak. Upon calling drmemory, we saw there are 0 bytes of memory leakage.

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 the project made the workload much easier and more enjoyable. It was also very helpful in terms of debugging since we had each other to bounce ideas off and talk through coding problems.

Something we did have some difficulty with was the git push/git pull processes. At times we had merging conflicts when we were both working on the file, and we tried to git pull/push. We ended up mostly working on one screen together to resolve this issue.