COMPENG 2SH4 Project – Peer Evaluation [30 Marks]

Your Team Members Hee6

Team Members Evaluated Tanga55 Hassaney

Provide your genuine and engineeringly verifiable feedback. Ungrounded claims will lead to deductions. Completing the peer code evaluation on time will earn your team a total of **30 marks**. Do not exceed 2 paragraphs per question.

Peer Code Review: OOD Quality

1. **[3 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 logic in the main program loop is easy to read. All variable names are representative of their function, comments are clear, and the program is organized which improves the overall program's readability. The contents of all functions are representative of the function name. For example, RunLogic does contain the game logic, and DrawScreen contains code for the console screen. This is a positive because it enables easy accessibility to edit or change features of the game. This can be seen in the Initialize function where you can change the board size easily by altering the parameters inside "GameMechs();. Finaly, the game provides clear instructions on how to play the game in the console screen.

   The program does come with a few minor issues. For example, the console screen is not fully aligned properly. The "SCORE" text on the console screen is not aligned properly to the left but starts with a space character. In addition, the program has a input debugging feature where the console screen shows the current input of the user. This part of the program was left on the console screen and makes the console screen a little cluttered. The input flashes for a small amount of time and disappears which takes away from the immersion of the game but not the functionality. I would have like to see a keyboard key that enables visibility of the debug function instead.

2. **[3 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 pros of the OOD approach includes program readability, program maintenance, code reusability. Instead of creating multiple functions throughout the program, each piece of relevant logic is kept together in one place. For example, the player object houses its coordinates and icon rather than keeping separate variables for each. It is simpler to debug faulty code because the user can identify which object is faulty and edit the file rather than scrolling throughout a single file. In addition, variables can be accessed from anywhere in procedural design which increases the difficulty of finding bugs.

Finally, objects can also be reused for different functions if needed. For example, the objPos class is used in player and food(GameMechs in the case of this group) objects rather than having a separate functions that handle each.

The cons of the OOD approach include the steeper learning curve, less efficient for small projects, and dependency on code. The concepts of OOD are much harder to grasp over procedural design it is harder to understand what a class does over a function. In addition, classes have multiple requirements for them to run efficiently. This however is a drawback for smaller projects because it involves more time to set up the program. Finally, the dependencies are an issue because it may be harder to trace a bug. For example, the player object uses the objPos object which means if the player object is faulty, the issue could not lie within the player class but rather the objPos class.

Peer Code Review: Code Quality

1. **[3 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 code does offer sufficient comments and has a self-documenting coding style. Comments are well placed in locations that aren't distracting. For example, in initialize, only lines of code that may seem confusing at first are commented rather than having a comment for every variable. In this case myGM = new GameMechs(26, 13)'s function is confusing at first glance. The comment of "Makes a board that's 26x13" is clear and descriptive of what the line of code does. In addition, long comments are not placed on a single line, but multiple, eliminating the need to scroll horizontally. As mentioned above, variable and class names are descriptive of their function, meaning it is easy for the user to understand their function.

   A minor drawback of the comments that in some header files like objPos.h, there is a comment on every line, slightly reducing readability. Some of the functions do not need comments as they are already descriptive. This includes getsymbol() and isPosEqual() functions.

2. **[3 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 does follow good indentation, sensible white spaces, and deploys newline formatting for readability. All If, for, while etc. statements include indentation which clearly indicates the code body that belongs to it. All equal to and assignment operators include whitespaces before and after use. Something the program does well is separate groups of code with newline formatting. For example, in initialize, the MacUI functions are separated with a single newline and the variable initializations are separated with a single newline. The MacUI functions and variable initialization on the other hand are separated with two newlines, indicating they perform different functions.

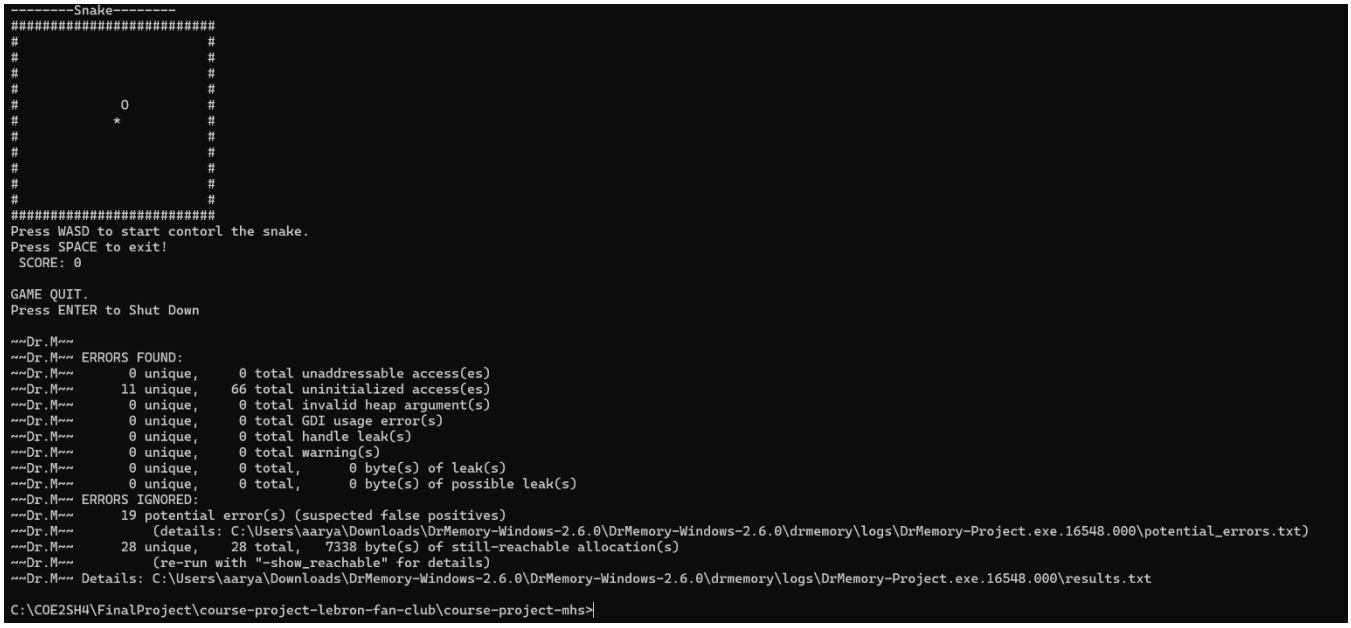
Peer Code Review: Quick Functional Evaluation

1. **[3 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 technical user feedback)

Running the snake game was a bug free experience, as the game ran in a fluid manner without any faulty behaviour. The game achieved all the assigned tasks such as spawning food in random locations, mimicking snake movement as well as object collisions without having any bugs. The team can work on changing the delay time to ensure smooth gameplay across all devices as sometimes the gameplay can be buggy and cause lag on the screen depending on a device's hardware limitations.

2. **[3 marks]** Does the Snake Game cause memory leak? If yes, provide a digest of the memory profiling report and identify the possible root cause(s) of the memory leakage.
There was no memory leakage when the Project.exe was ran under DrMemory.



Project Reflection

Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team's implementation in addition to yours, reflect on the following questions:

1. **[3 marks]** Do you think the compound object design of objPos class is sensible? Why or why not?

Since a class and a struct are both complex datatypes, we do not believe that the objPos class's compound object design makes sense. Making the x and y positions components of the class itself would be easier and more efficient because it would reduce the heap and the eliminate the need for using structs.

2. **[4 marks]** If yes, discuss about an alternative objPos class design that you believe is relatively counterintuitive than the one in this project. If not, explain how you'd improve the object design. You are expected to facilitate the discussion with UML diagram.

To improve the object design, one can move the x and y pos struct into the class. This would reduce the heap and the eliminate the need for using structs.