COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Tasmiah Gani & Tajmim Maisha

Team Members Evaluated Jana Nofal & Lujain Nofal

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 file, it is possible to interpret the possible behaviours of the objects involved in the program and how they would interact with each other in the program. The method names are descriptive and similar methods are grouped together in the header files. For example, in the GameMechs header, the two methods to get the board dimensions are grouped together [getBoardSizeX() and getBoardSizeY()], the same for the two methods involving food generation.

There are also a few comments describing the functionality of the method, however, more could have been added for better understanding of the methods.

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.**

Looking at the main logic program loop, it can easily be interpreted how the objects will interact with each other in the program logic. The loop condition is clearly stated, the program will run until the exit flag is set. The four functions inside the loop, GetInput(), RunLogic(), DrawScreen(), and LoopDelay() have descriptive names that help with the readability and understanding of each step of the program. However, since no comments were made, it can make it difficult to read and understand for future programmers looking at this 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.**

C++ OOD approach

PRO:	CON:
• Improved code readability and organization • Code reusability • Reduced code repitition • Increased data protection • Improved organization and efficiency when working with multiple people on the same project • Efficiency in coding the main logic of the program • Short main program file	• Longer code compared to when using the procedural approach • More advanced coding than procedural • Complex (could be difficult to maintain and remember functionality of each class and method. • Many files (can be difficult to keep track of them all)

C procedural design

PRO:	CON:
• Less line of code • Simple (not working with multiple files) • Less coding skills required compared to the OOD approach • Less time consuming	• Limited Code reusability • Limited data protection • Decreased code readability and organization (since all code are in the same file) • Relies on global variables (might run into problems with scoping issue) • Long main program file • Repetitive code

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.**

As learned in class, commenting in code is practised as an etiquette that acts as documentation and to improve readability of the program. The code provided to us included a good number of comments that at times would act to enhance understanding to the functions or variables used throughout the code. It was also noticed that the code contained helpful comments prior to introducing loops, such comments explained the purpose of the loop, and the conditions that came with them. This was functional as it would be very helpful for anyone regardless of their coding background to understand the purpose of the plenty loops used throughout the entire code. One shortcoming noticed is that the code sometimes seems to lack comprehensive comments which explain the flow of logic. For example, in the Project file, we noticed that the loops and conditionals were explained within a function, however the functions lacked commentary. We thought it would be a bit more helpful for a coder looking at this group's code, if the code contained comments before each function, briefly explaining the purpose of the function, in contrast to having a coder having to look through each function to understand the purpose of each function.

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.**

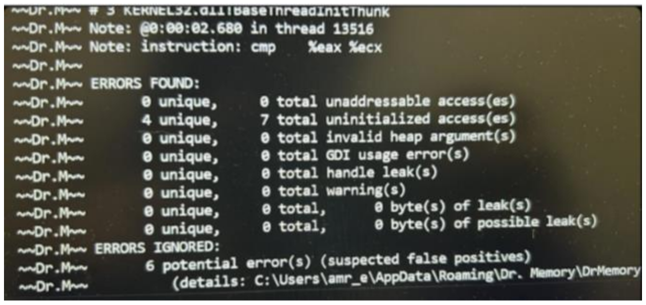
The code generally follows good indentation and uses sensible white spaces, making it readable. The code demonstrates consistent indentation, and there are no apparent issues with indentation inconsistencies, resulting in improved readability and proper reflection of the structure of the code. Through proper indentation etiquettes, we were able to visually identify the occurrence of different loops, functions, and conditionals in the code. The coders also were able to properly deploy newline formatting contributing with overall readability. No significant shortcomings were found as everything in the code seemed to be neatly organized through the following of the coding conventions.

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 ran smoothly, and it was a bug-free playing experience. However, changes could be made for a more fun playing experience. However, we noticed that the group decreased the speed of the snake in their code, this made it a bit too easy to play their game, which made it a bit less eventful and more time consuming to play the game. Additionally, when the game ends when we lose, the score board display disappears, so it is hard for the player to tell what score they had each time after they have lost. For an improved player experience, it would be better if score was displayed the entire time. We noticed when we played the game, the group forgot to delete their debugging messages, thus there was too many print statements displayed during the game, which impacts the professionalism of the software development. Lastly, when playing the game, we noticed there were no instructions of which keys to use or what the exit key was, this decreased the overall player experience and quality, as for new players, it would be difficult to get the game started and playing.

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.**



```
~~Dr.M~~ # 3 KERNEL32.dll!BaseThreadInitThunk
~~Dr.M~~ Note: @0:00:02.680 in thread 13516
~~Dr.M~~ Note: instruction: cmp    %eax %ecx
~~Dr.M~~
~~Dr.M~~ ERRORS FOUND:
~~Dr.M~~       0 unique,     0 total unaddressable access(es)
~~Dr.M~~       4 unique,     7 total uninitialized access(es)
~~Dr.M~~       0 unique,     0 total invalid heap argument(s)
~~Dr.M~~       0 unique,     0 total GDI usage error(s)
~~Dr.M~~       0 unique,     0 total handle leak(s)
~~Dr.M~~       0 unique,     0 total warning(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of leak(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of possible leak(s)
~~Dr.M~~ ERRORS IGNORED:
~~Dr.M~~       6 potential error(s) (suspected false positives)
~~Dr.M~~         (details: C:\Users\amr_e\AppData\Roaming\Dr. Memory\DrMemory
~~Dr.M~~
```

Using DRMemory, it was confirmed that the Snake Game does not leak any memory, ‘0 byte(s) of leak(s)’. A picture of the DRMemory report found is provided. Dynamic memory allocation of the gameMech was done using new. At the end of program, the GM object from heap, gameMech was deleted under the CleanUp() function to release unnecessary stored memory. Through our visual inspection of the code, we can be assured that there is also no memory leakage in the code.

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.**

In our first collaborated software development project, the experience was both enlightening and challenging. One aspect that worked exceptionally well was the merging of our unique ideas and skills. By being able to collaborate with one another, we both were able to leverage one another's strengths, resulting in an extensive and robust solution. Additionally, by having two people working on one software development project, our group was able to split up the work amongst ourselves. For example, one team member worked on one parts of the iteration (ie: iteration 1A), while the other team members worked on the other part of the iteration (ie: iteration 1B). By splitting up the iterations, we were able to efficiently get through the project in a shorter amount of time. Overall, both of our experience working on our first collaborated software development was a successful experience.

However, one thing that could be improve on is better scheduling of our meetings. We both had other deadlines and a midterm to study for, so it was difficult to schedule meetings early in the day and in person. We usually had our meetings late at night online. For some parts, especially iteration 3 it would've been nice to collaboratively work on it face to face. However, we both understand that our schedules were quite packed and busy, thus we were understanding of it and mutually agreed to meet on very late nights.

Additionally, collaboratively working on VSCode was also quite difficult, as each time one of us made changes on the software, we would have to git push and pull on each of our ends to merge the codes. This was inconvenient at times as we sometimes did not work on it at the same time, so we would have to let the other know to git push and pull during the next time they begun working on it. To add, we end up running into frequent errors and issues when merging our works after a while, which would result to us having to spend time to troubleshoot the merging of our codes. Although we understand this is just how VSCode works, and there is nothing much we can do about it, we would like to just highlight how this was one of the negative sides of working on a collaborative software development project.