COMPENG 2SH4 Project – Peer Evaluation

Your Team Members: Anna Norris, Angelique Stahl

Team Members Evaluated: Darren Temporale, Konstantin Delemen

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 new header file for the snake food is clearly named and has comments describing what each function does. The generation of an initial piece of food is separated from the rest of the food generation, with a comment explaining that there is only the position of the snake head being passed in to block off. In addition, they chose to modularize the creation of the game board, adding a printBoard function inside of GameMechs. This results in a very clean-looking project.cpp file.

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 very clean and easy to understand. Their program initializes then iterates through 4 different functions until the programs exit flag is true. Then a cleanup function is called before the program terminates. In each function it is easy to know what each object is doing and how it is interacting with the code as the names of objects aswell as functions are descriptive. Something that could be added to increase clarity would be additional comments in functions with larger code blocks, such as the draw function, as they can get overwhelming without additional clarity.

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:
   - Keeps main function looking clean and, if sensible function names are used, readable to understand the functioning of the code
   - Reduces the need to copy and paste code, since functions can be reused multiple times
   - Allows for more complicated programs and for multiple programmers to work in parallel on different parts of the same program

   Cons:
   - If functions are poorly named, the code can quickly become a list of incomprehensible calls, which must then be hunted out through their own files.
   - Every private scope variable in a class needs to be equipped with the methods to access and change it, whereas a procedural programming style will have all variables immediately accessible inside their own scope.
   - Modularization will lead to a large number of separate files, which must all be correctly linked to

each other.

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.

   In general, the code offers adequate comments. The one area where commenting could have been improved is the modularization of the game board drawing; it is useful to have the drawscreen function simply call a gameboard function, but since that style is a departure from the style of game board used in the PPAs from which that code was supposed to be inhereted, it would have been useful to have the change flagged in a comment. Also, in some cases comments in the code are redundant and do not help improve the reader's understanding of the logic. To improve the code, comments should describe code that is not human readable in natural language.

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.

Yes, the code follows good identation and the format is easily readable. Newlines are used properly to clump similar logic and seperate different actions. No code is unproperly indented including in loops, functions, and if statements. No large improvements need to be done to improve the format. If any improvement needed to be made it would be to simply ensure a whitespace is included between variables declaration and logic as it was missed in the printboard funtion.

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 does provide a bug-free playing experience; the food pieces generate randomly on the board and, so far as our team was able to tell, never on top of the snake segments. Although the team did deply above-and-beyond features, with multiple pieces of food and a "special food" displayed on the board, it is not clear to the player what those features are and what result eating the special food will have; since you would have to look at the code to figure this out, it would be a better user experience if the instructions stated what symbol represented the snake, regular food, and special food, and what effect the special food has on the snake.

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 has 0 bytes of leakage and possible leakage. For every object created on the heap a destructor is made and called at the end of the program.

```
~~Dr.M~~
~~Dr.M~~ ERRORS FOUND:
~~Dr.M~~       0 unique,     0 total unaddressable access(es)
~~Dr.M~~      13 unique,   242 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)
```

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 with a collaborator was made easy mostly by the detailed instructions in the manual describing sections of the code that could be written independently of each other; even so, we did have some instances of needing to manually merge changes, which was not covered in class. In the future, we would need to think carefully about what each partner's assigned workload should affect in the code and make decisions about workload accordingly.