COMPENG 2SH4 Project – Peer Evaluation

Your Team Members _Jason du_____
_____Cale Ying_____

Team Members Evaluated _____ Yaohui _____
_____ Samuel _____

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: First, a well-designed header file can provide clear and concise information about an object's structure and behavior. For example, from the header files, we can know what GameMechs.h", "Player.h" and "Food.h" represent. Clear variable names can increase our understanding of the entire project. Secondly, each header file can be understood and edited as an independent module. For example, the "Player.h" header file represents the player. We can find all the code about "Player.h" from "Player.cpp" and edit it. The player passes w,a,s,d move and interacts with the files in "Food.cpp". When the player eats different food, it will produce different effects. In other words, each header file can interact with each other.

Negative Features: The shortcomings of header files are also obvious. If the comments in the header file are unclear or insufficient, their use and understanding will become more difficult. If there are some important files, it will make later collaboration and maintenance difficult. And too many header files will cause a certain module to perform too many operations, making the entire project confusing and difficult to execute.

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.

In general, much of the core logic isn't executed in the main loop, limiting my complete understanding of the properties. The initialization function appears straightforward and logical, similar to the input handling. The logic for running the program is concise and clear. However, the screen drawing process is somewhat complex due to the numerous loops involved. Adding more comments or better organizing these loops could enhance their clarity and intuitiveness.

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.

In summary, the choice between C++ OOD and C procedural design depends on the size, complexity, team expertise, and specific requirements of the project. C++ OOD has advantages in modularity, encapsulation, and extensibility, but it requires a deep understanding of object-oriented principles. C-procedural design may be suitable for simpler projects or where a more direct approach is preferred, but as the project grows it may face maintainability and scalability challenges.

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 programming code primarily utilizes a self-explanatory style through its choice of names for variables, methods, and objects. However, there are sections where clearer variable names could have been beneficial for better understanding. Regarding comments, they are present to some extent throughout the program. Including more detailed comments to describe the purpose and behavior of the methods in each file would significantly enhance the ease of understanding the code.

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.

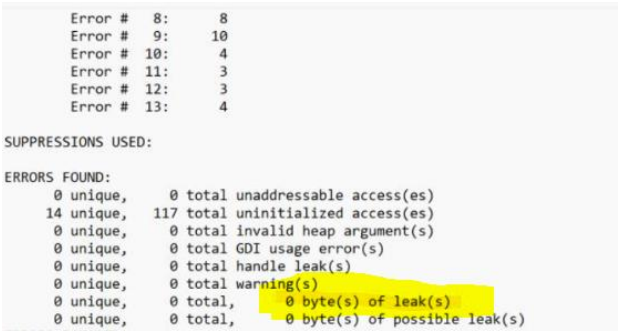
The code is well-indented and effectively organized, with clear separations between different sections. While there could be slightly more spacing in some parts of the C++ files, this is a minor issue and does not significantly impact the readability of the code.

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 code operates quite seamlessly, and it is free of bugs. The snake moves fluidly, and the food placement is random but never overlaps with the snake. Additionally, the collision detection functions flawlessly.

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.



```
       Error #   8:      8
       Error #   9:     10
       Error #  10:      4
       Error #  11:      3
       Error #  12:      3
       Error #  13:      4

SUPPRESSIONS USED:

ERRORS FOUND:
      0 unique,     0 total unaddressable access(es)
     14 unique,   117 total uninitialized access(es)
      0 unique,     0 total invalid heap argument(s)
      0 unique,     0 total GDI usage error(s)
      0 unique,     0 total handle leak(s)
      0 unique,     0 total warning(s)
      0 unique,     0 total,      0 byte(s) of leak(s)
      0 unique,     0 total,      0 byte(s) of possible leak(s)
```

Their snake has no memory leakage, this is because they added delete () statements in their destructors.

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.