COMPENG 2SH4 Project – Peer Evaluation

Your Team Members ___Zerui Chen_____ _____Shijun Gao_____

Team Members Evaluated _____John_____ _____Stan_____

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.

We can easily interpret the possible behaviors and understand the interaction between the objects when reading the header files. The code is written in a clear and professional way, so it is easy for others to understand and follow the code. However, there is a concern that the collision with the food function is not very perfect, when the snake head collides with the food, the food will be pushed to the next space,and there are some delay when the player's behavior is updated, which may cause that the snake can collide with many food before the new food are generated.

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 program loop is easily to interpret the interaction with each other for other users. The code is in a high quality and the readability is excellent. However, the only small negative feature is that the in-game message is not very detailed,

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 Pros:

1. **Code can be reused easily**
2. **Make the code more readable.**
3. **Helps to make the code more flexible to handle**

C++ OOD Cons:

1. **May be much more complex compared to procedural code.**
2. **May have a steeper learning curve**

C Pros:

1. **More straightforward and easier to understand and write for some small projects.**
2. **Easier to learn compare to OOD.**
3. **Procedural code may have lower overhead**

C Cons:

1. **Most of the code cannot be reused.**
2. **The code can be difficult to read.**

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 code offers very precise and detailed comments, and it is easily to see that the code is written by themselves because the comments in the code explains their thinking process in the Player.cpp file. However, the comments in the Project.cpp file is not so detailed than the comments in the Player.cpp, it may hard to understand some logic of 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 readability is good, the indentation are used reasonable, it is easy for others to read the code in Player.cpp. However, it can be better if some white spaces are added in the Project.cpp file, especially in the DrawScreen function, the code and comments looks like squeezing together, the following screenshot shows the DrawScreen function.

```
// Draw the player at its position on the board
for (int y = 0; y < mechs->getBoardSizeY(); y++) {
    for (int x = 0; x < mechs->getBoardSizeX(); x++) {
        if (playerPtr->isPlayerPos(x,y)) {
            // Draw the player symbol
            cout << "*";
        } else if(mechs->isFoodPos(x,y)){
            // Only print the first food symbol found at this position
            for(int i = 0; i < foodList.getSize(); i++){
                foodList.getElement(food,i);
                if(food.x == x && food.y == y){
                    cout<< food.getSymbol();
                    break;  // Exit the loop after printing the first food symbol
                }
            }
```

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 Game offers a smooth, bug-free gaming experience, the only concern is that the snake can eat many food before the new food are generated, which was already mentioned in Part1.1.

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.

There is no memory leakage, the management is perfectly done.

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 this project, one of the most important thing is communication, a high quality of communication helps to improve the efficiency of working process, for example, my partner was suffering a error, and I didn't know which part caused the error, he was thinking for a long time, but he suddenly realized what's wrong with it while talking, and sometimes, I was stuck, I told him my concern, he helped me to finish the part, it is much faster than thinking by myself. However, there are something requires to improve, we didn't have a clear role of working, so we had to send the file again and again, sometimes, we messed up the version of the file we sent, it slows down our efficiency of working