COMPENG 2SH4 Project – Peer Evaluation

Your Team Members _Leopold Blew – 400454641 Jake Vu-400462600

Team Members Evaluated _ Zhenghan _ _ Yifan _

Provide your genuine and engineeringly verifiable feedback. Ungrounded claims will lead to deductions.

Part I: OOD Quality

1. **[5 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 only the header files, there seems to be a lack of comments describing the functionality of variables and methods etc. Looking deeper, most comments seem to be instead written in the .cpp file of their respective class. This is not technically the correct implementation as the general descriptions should be in the header, but it is acceptable to instead have them in the .cpp file. With this said, the chosen variable/method names are descriptive and well chosen. There are left over debugging implementations and commented out code which sometimes make it hard to find descriptions for the lines of code actually in use, such as in the Player.h file. Overall, we believe that the interplay and choice of variable and method names are well done. However, unnecessary code and comments should be removed, and more general method/variable descriptions should be included in the header files.

2. **[5 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 program loop has a good amount of commenting to describe most of the processes. However, some variables, such as "initMap" are not well documented, and it is unclear what they do. As well, we believe that too much of the code is procedurally executed in the draw method. There should more encapsulation done to separate the code into more readable chunks so that the flow of the main loop is more coherent. As well, there is leftover debugging code and comments which add confusion to the rest of the meaningful comments. Overall, the code is mostly described well, but could be easier to decipher if it were encapsulated and documented more.

3. **[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.

Pros:

- Allows for variable/method privatization, increasing security
- More future proof and more easily improvable without rework
- Increases encapsulation and allows code to be more bite-sized

Cons:

- More computational overhead associated with shell/manager objects
- Can be more difficult to access wanted data and can decrease ease of code implementation

Part II: Code Quality

1. **[4 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 previously discussed, the code offers almost sufficient amounts of commenting, and just falls short for some variables such as the "initMap" variable in project.cpp. Other than these few shortcomings, the code offers a great amount of documentation that thoroughly explains most of the content. To improve, we would perhaps change some of the variable names of the undescribed variables to make have a more implied meaning. As well, we would write more detailed explanations as to why a certain approach was chose over another to explain to the reader the logic behind the choices made.

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 could use more blank lines to separate methods/variables into packets that are used together. This could be implemented in many of the header files, such as FoodBucket.h, where there are no blank lines used to group definitions together. This could be improved by identifying what methods/variables are used together, and grouping them by separating them with a blank line on either side, perhaps accompanied by a comment for clarification.

Part III: Quick Functional Evaluation

1. **[6 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)

It seems no food/trap objects are drawn at the leftmost column of the playing field. This is likely due to the range used to find new random coordinates. The other team should inspect how the modulus operate works with a given random number, and adjust their code accordingly to allow for spawning on the left side. In addition, they could use a debugger to view the output currently being provided by their code. The code does run at a sufficient speed to be playable, however, sometimes inputs do not register and the user must input the same key multiple times just for one desired action. This could be fixed by decreasing the game speed just a fraction so that more inputs are caught, whilst also not increasing the difficulty of the game too much.

2. **[4 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 in this code. However, we believe that the “myGM”, “myPlayer”, “SnakeFoodBucket”, and “traps” objects in main should be explicitly deleted in the cleanup method instead of leaving this up to the c++ compiler to interpret.

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.

Issues with GitHub were a big concern and caused a lot of unnecessary stress. If one thing went unexpectedly, we were in the dark and had no experience to back us up. We believe there should be some teaching, or maybe a help document on avenue, that could aid us to really understand what GitHub is used for, and what all of the terms used by GitHub mean (such as merge, branch, head, etc) which may easily present themselves but for which we have no experience. Working in a one person team may be have less confusing logistics, but also this confusion may be desired as it is inevitable in the workplace, and the more we have safe opportunities to experience it the better. This could be improved by giving more prior instruction on topics such as GitHub, so that we can be better at identifying logistics issues, and solve them instead of aimlessly trying to fix a problem.