COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Yaohui Cai Samuel Abdizadeh

Team Members Evaluated ~~Junlong Ying~~ ~~Xiao Chen Du~~

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

Note: The group that was assigned to us did LITERALLY NOTHING before the deadline and only pushed very little work that cannot be really evaluated AFTER the deadline- their program does not even fulfill the basic requirements of the project (it's only a cpp version of PPA2). Moreover, there's a potential plagiarism issue in their code: they did not test the arrayList class in the test suite (the arrayList class in the testing folder is empty) and did not use anything from the arrayList class, yet they have a functional arrayList class in their main project, which is very strange.

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.

   They didn't work on the project at all before the deadline so we'll be evaluating our own code.

   Yes, first you can see what they interact with by looking at what is included. Then you can see the public members that show the way the object works. Lastly, you can see the functions that the object can do at the bottom of the public section. These are all separated by enough space to allow an easy interpretation. There could be comments in food.h to better understand exactly what is done, though.

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.

   They didn't work on the project at all before the deadline so we'll be evaluating our own code.

   Overall, yes, you can see at a high level what is happening. Most of the behind-the-scenes logic isn't happening in the main loop, so you also don't have the best view of the properties overall. Looking at the initialize function, everything is straightforward and makes sense, get Input as well. Run logic is very short and simple and makes a lot of sense because the naming convention is very well done. The draw screen is the most confusing just because there are so many loops. Maybe more comments or separations could be included to make each loop stand out more and be more intuitive.

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.

   They didn't work on the project at all before the deadline, so we'll be evaluating our own code.

   Pros
   -Modularity through the classes and objects that allow a lot of customization to make each aspect of the code
   -Easier to reuse code

   Cons
   -More complicated
   -Longer to develop
   -May be less efficient

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.

   They didn't work on the project at all before the deadline so we'll be evaluating our own code.

   Yes, overall, the code had enough comments to understand. It is easy to understand the overall flow of code and what is happening. Some parts are missing, but if the reader has any experience with cpp, they should be able to understand or fill in the small gaps.

2. **[4 marks]** Does the code follow good indentation, add sensible white spaces, and deploy newline formatting for better readability? If any shortcoming is observed, discuss how you would improve it.

   They didn't work on the project at all before the deadline so we'll be evaluating our own code.

   The code does follow good indentation. There are good separations between the code to make each part different. Some parts could have more space in the .cpp files but is minimal and doesn't affect readability.

Part III: Quick Functional Evaluation

1. **[8 marks]** Does the Snake Game offer a 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 technical user feedback)

They didn't work on the project at all before the deadline, so we'll be evaluating our own code.

The code runs smoothly without any bugs. The snake works great, and the movement is that of a snake. The food is distributed randomly, not on the snake, and collision detection is perfect. As the contributor to the random generation section, I sincerely invite you to read the random coordination generation part of the code. It's interesting.

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.

   They didn't work on the project at all before the deadline so we'll be evaluating our own code.

   It does not leak any memory. Running the DrMemory report shows that. All memories allocated on the heap are deallocated using destructors at the end.

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.

   We had good communication throughout the project, so we were able to talk about our goals and how we wanted to achieve them. We followed an agile methodology that allowed us to work on individual components separately based on a rediscussed plan. There were some challenges in making everything work when we put it together just because of variable naming conventions and GitHub merging issues. A notable part of the collaboration was the makefile- because we had different operating systems, we needed different settings in makefiles. Other than that, we finished early due to good programming practices.