COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Makarios Samwail Marwan Ali

Team Members Evaluated Li-Xintong Xikai Xu

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.

Player.h:

Very reasonable member naming, it would have been better to stick to a single naming convention like camel case which is already provided in the skeleton codes. Some of the comments could be more detailed. For example, line 19 has just the comment "FSM" which doesn't really describe what the function does and how it interacts in the main program. There is also an unnecessary comment on line 29 of some unused code which could be deleted.

objPosArrayList.h:

The comments describing the class members were a lot better for this class. The blocks of member functions grouped together underneath their respective comment descriptions makes it very easy to interpret their functionality. One thing to change would be to be more descriptive with the print_pos variable and the methods that use it like detect_to_print and detect_get_char.

objPos.h:

Great class file header. The newly added member functions are named descriptively following the same naming convention. There is also a comment that lets us know that the 2 new member functions are used for comparison. Well done!

Food.h:

This class had good commenting. One thing we were confused about was the need to create a temporary objPos (tempFood) object as a class attribute. Normally you could just create a temporary objPos object in the body of the helper function that requires it. This is more memory efficient because the tempFood object would be freed at the end of whatever function call that uses it instead of having to wait till the food object is freed.

GameMechs.h:

Good comment grouping. Functions all have descriptive names which explains their purpose. Once again, we like the commenting over the group of functions which explain what the functions are used for.

However, we think that the “stock” comments are unnecessary since they don’t provide meaningful information to the programmer.

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.

Going through the project source code, we noticed how easy it was to interpret how the objects of the program interact with each other. This is the result of using descriptive function names and running the write program logic under the write program logic function (for example, setting the game mechanics input under the GetInput program function).

Something we would like to see changed is in the DrawScreen function, the use of numbers in the for loops could have been replaced with preprocessor constants that describe what their value represents. This also could have been done in the condition blocks that check the snake award value to a number.

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:

- Better organization compared to the maintenance challenges for C procedural programming.
- Better understanding of the functions with respect to their class
- Ability to model any object in a programmatical sense.
- Being able to protect data members from developers using the objects.

Cons:

- Steep learning curve compared to the simplicity of C.
- Very easy for a simple project to become overly complexed based on how you design your classes.

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.

Very minimal commenting in the main project source code. It would have helped us in understanding how objects interacted in the main program if their were comments documenting their thought process. Especially in the run logic program function, a lot of comments would have been beneficial in understanding how they implemented their code and why they chose to ,

implement something the way they did. Also commenting the purpose of the temporary objects like on line 66, 57, 78, and 79.

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.

In our opinion, all their source files follow good indentation. No comments are cluttered together, and the indentation helps show what code is within a loop or conditional block. Doing this allows us to read and understand their code logic a lot quicker since we do not need to spend time trying to figure out what code is executed in which block of code. There is no real shortcomings within their 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 snake game ran perfectly fine. They implemented the bonus which worked perfectly fine. The only issue we had with their game was not understanding what the different powerups did. The only way we could figure out what each powerup did was by looking at their source code. We recommend adding a little legend at the bottom of the game UI that explains the rules for the powerup. Another thing we would recommend is getting rid of their debugging keys so that players do not modify the game.

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.

Their game does not leak memory. In the initialization phase, they dynamically allocate memory on the heap for the game mechanics, food, and player objects. In the clean up phase, they make sure to deallocate the memory they allocated for each object in the initialization phase. As well, they made sure to create a destructor for classes for the player, food, and objposarraylist which all allocate memory on the heap internally within the class.

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.

   The project was fun to work on since it was our first time experiencing what coding in industry is like. It was frustrating coming across a bug but learning how to be patient and calmly go through the logic of your code is an important skill to have. Like professor Scott says, the compiler does exactly what you tell it to do. We only wish that there was a brief introduction to using git in the beginning of the course, especially for having multiple collaborators. Normally using git as a solo developer is not that complex since you do not need to worry about merge conflicts and using the conflict editor.