COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Haocheng Li Joonseo Lee

Team Members Evaluated Rameez Nerjes

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.
    a. **The members of each class are clearly named. GameMechs clearly contains all the required data members with appropriate accessors and mutators, although there is a duplicate for the loseFlag boolean. Player, objPos, objPosArrayList have appropriate member names that reflect their functionality.**

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.
    a. **checkPlayerCollision() in Player is a method that return type bool. However, in the implementation, there is no return statement, and instead it just sets the loseFlag through mainGameMechsRef.**
    b. **The global variables declared at the top of the code are problematic**
        i. **tempPos is never used**
        ii. **temp_food_pos, PlayerBody, tempBody, can all be declared within the local scope of DrawScreen() instead of in the global scope.**
    c. **The draw routine has a block of poorly implemented if statements. The conditions for drawing each element on the board is unclear, and the code is very hard to read.**

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.
    a. **The OOD approach allows for faster development and easier debugging. It also allows for easier tracking of what feature is stored under which object, which allows for cleaner code. However, the procedural approach is faster to get started, since each object does not need to be prototyped before developing.**

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.

a. **Code is mostly well documented and explained. Some places have confusing and outdated comments. The draw routine needs some more documentation, and so does generateFood() in GameMechs. In line 159 of Player.cpp, the comment points out an issue, but the fact that the comment remained to the final submission, suggests that either sloppy documentation or that is the issue hasn't been fixed. It seems that the partners are communicating with each other through temporary comments in the code, which is normally fine, except they haven't removed those comments. At line 102 in GameMechs.cpp, one team member points out to the other that they have created a isPosEqual() method for comparing the coordinates of two objPos objects. However, the method is never used, suggesting that the teammates had inadequate communication.**

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.
    a. **The code does NOT follow good indentation. The indentation spacing is inconsistent (sometimes 3 spaces, sometimes 4 spaces, sometimes the tab character, etc.).**
        i. **To improve, choose common indentation (ie. tab char) and use it whenever a statement becomes nested within another (ie. a conditional within a loop) and be consistent with it.**
    b. **Curly brackets for code blocks are also very inconsistent. Sometimes, the brackets are right under the associated statement, sometimes an indentation behind, and sometimes an indentation ahead.**
        i. **To improve, choose common curly bracket placement style, and stick to it. We chose to put brackets on the same line as the statement (ie. statement {}), but could also use the style of** statement \n {} **instead.**
    c. **Spacing is inconsistent. After conditional statements, sometimes a space is placed, while during other times it isn't.**
        i. **To improve, choose a spacing style (ie. x = y or x=y) and stick to it throughout.**

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)
    a. **Sometimes, the input did not register, and the player did not move.**
        i. **To test, spam two keys (tried S, then D)**
        ii. **The root cause appears to be the input getter function of the GameMechs class, where they get the next available char if no available chars were available.**
        iii. **To find the root cause, use the GDB debugger on the input getter of the GameMechs class.**

    b. **There is no lose message. The game just terminates, with no indication whatsoever on why the game ended. This is the same behavior as pressing the exit character, which increases the confusion even more.**

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.
    a. **Drmemory indicates no memory leak.**

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 manual clearly laid out the responsibilities of each member of the team, which made it easy for each member to develop the project in a modularized manner. The two developers are able to work independently for the most part.