COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Ayush & Kirit

Team Members Evaluated Sofia Perme & Natalia Jara

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.
    - Looking at the header files, the program follows the Object-Oriented Programming model with classes representing different entities and their behaviors. There's a clear separation of concerns among the classes, promoting modularity.
    - I can easily interpret the behaviours of the objects. If we hadn't coded the same program, we could easily tell what the objects do simply based on their names. For example, the "bool checkSnakeSuicide" function in the "Player" object. It is obvious that the function is there to check if the snake ate itself and probably returns true if it did.
    - However, it was hard to tell what the "getByPos" function does in "objPosArrayList" object. There were no comments to explain it either (even in the source code).
    - For the most part, the behaviour can be easily interpreted based only on the header files.

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.
    - Even with the lack of comments, it is easy to tell how each of the objects interact with each other through intersecting outputs in addition to good code formatting, each action is spaced when a new class comes into play (similar to how paragraphs in an essay are structured).
    - For example, in the "Initialize" function, the code: "myFood->generateFoodBucket(myPlayer->getPlayerPos());", can be easily interpreted. It generates food and passes the player position as an argument (to most likely ensure the food isn't generated at the player's coordinates).
    - Overall, the main program code is very easily interpretable, and no negative features were observed.

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 approach allows the developer to have a clean and organized main file and also allows multiple people to work on the project at once and test their own code, without affecting each others work.
    - If this was done procedurally like in PPA3, our main code would be much harder to read with the increase in content this project brings and working with a partner on the main program would prove to be a hassle especially if the other person keeps running into bugs and doesn't allow for compilation.
    - OOD makes it easier to understand the code, change the code, extend the code (for example, doing the above and beyond feature), and reuse the code (using "objPosArrayList" to also generate food).
    - Some cons would be the learning curve to understand everything (by coming from previously only doing procedural design), more use of memory (for example, having to make multiple temporary objects), and might 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.
    - The code uses comments where needed, as areas of the code that would take time for an outside developer to figure out now become easy to follow through the concise and well-placed comments.
    - Other areas such as in objPosArray.cpp did not use as many comments but due to proper structuring, were easy to follow.
    - Minor improvements such as adding comments explaining some the algorithms the objects use (such as the "getByPos" function in "objPosArrayList" as mentioned before) would make understanding the code even easier.

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.
    - Yes the code effectively employs the use of indentation, allowing the viewer to easily follow along with which block of code is connected to where.
    - There are proper indents right after for loops, if statements, case statements, etc.
    - There is also good separation between lines of code using newlines to better show the next block of code separately from the previous block.

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)
    - Overall, the game is smooth without any bugs. The movement works as expected, with the food generation being on point. The wrap around and all the other features are also implemented correctly.
    - However, there is one small "bug". The program asks for input twice. In the "GetInput" function in the main file, the "getInput" function in the "GameMechs" class is called once. It is called again in the "RunLogic" function in the line "myPlayer->updatePlayerDir();". Looking at the "updatePlayerDir" function, it calls the "getInput" function in "GameMechs" in the line "char input = mainGameMechsRef->getInput();". This would be fine if the "getInput" method only returned the input. However, the function does the input processing (using "MacUILib_hasChar" and "MacUILib_getChar"), then returns the input. In conclusion, the input is processed twice. This only becomes an issue when pressing the keys to move the snake quickly (especially to do a 180^{o} turn since that requires 2 key presses). It doesn't register the keys properly causing the program to most of the time ignore the input or only do one of the inputs.
    - One solution can be to do the input processing in the main file once and have the "getInput" method only return the input.
    - Another bug can also be observed by looking at the code but not necessarily by the game since reaching the end of the game is difficult. The bug is that the program doesn't know how to handle the case of if the maximum array length is achieved. The only end game cases are if the snake eats itself or the game is exited.
    - One solution can be to implement this case the same way the suicide case was implemented (by adding a function in the "Player" class that checks this). Adding a separate exit message would also be a good feature.

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.
    - The game does not have any memory leaks.

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.
    - Kirit: Working in a collaborative environment for this project allowed us to learn how we can work in parallel to overall achieve the same task in an effective manner. I found that I wasn't really "bumping into" my partner as much as I thought I would with the code and felt that the work got done much faster than if I'd embarked on a task like this alone.
    - Ayush: The usage of OOD was definitely helpful. This made the experience much smoother as it was easy to split the work. The project manual made it easy to understand what my partner was doing and how to use their code in the main program or in my designated classes I was working on. Overall, the experience was great, and everything went smoothly.