COMPENG 2SH4 Project – Peer Evaluation

Your Team Members: Sarah Ahmed, Amna Yousafzai

Team Members Evaluated: Ali, Borna

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.**

The header files are designed to easily interpret the behaviour of the objects. For example, the player header file has a private data member named "shouldGrowBy" which is aptly named for holding the number of segments the snake body should grow by.

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.

They set the lose flag in the main logic which is a suitable place for it. However, the main logic has some redundant code. They made a function call for getting headPos, however they don't use the head element in the logic of this function. The logic for increasing the player length function is confusing due to the lack of comments and scattered use of the variable shouldGrowBy. The structure of the conditionals could be improved because they did if-else-if when it can be structured as if- else if-else.

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

- C++ enforces OOD properties such as encapsulation and composition through it's syntax.
- Allows us to build more modular code because we can create manageable components (functions) and call multiple instances of a class (objects) and it supports enhanced features such function overloading.
- Allows code to be organized in a hierarchical manner.
- Easier access of variables and data members in C++ because they are related through classes.

Cons

- Difficult to debug due to complex complex hierarchies
- Rigidity in design (For example, implementing the bonus feature of FoodBucket list caused issues in multiple files and required us to make changes in multiple classes.)

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.

Most of the code is well-commented, however, some functions such as increasePlayerLength(), has no descriptive comment. The generateFoodFunction has sufficient comments yet it is unclear how the food positions are checked for possible overlap against other generated food positions. There are redundant comments such as "create normal food" when creating a new food object. They should focus on commenting the less apparent logic to help with understanding.

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.

   It's readable and deploys good identation and whitespaces. It is structured well and is easy to follow.

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)

   The game offers a smooth gaming experience. The food collision, wrap around, and random food generation work as expected. One thing that can be improved is displaying the exit key information to users.

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.

The snake game has no memory leak because they deleted all objects created on heap in by implementing the destructor correctly.

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.

   Dividing the code between two developers is a good idea. Each person person was able to focus and learn independently however, collaborating for some parts was also beneficial as we were able to learn from each other. Also, having another person to help with the debugging and problem solving made the program development process faster and more efficient.