COMPENG 2SH4 Project – Peer Evaluation

Your Team Members _____Amadou_Wane____ ____Abdul Wahab____

Team Members Evaluated __Yasandu Gunawardana__ _____Abdul Moeez Bhatti__

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 behaviors 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.

   Positive Features:
   - Each header file is easy to read and understand what the responsibility of it is
   - member functions are implemented in a way where it's obvious what they do
   - Constructors and destructors used where needed with appropriate getter and setter methods
   - Their Player class used proper enumerations for directions. This enhances code readability and makes understanding and managing player movements easier.

   Negatives:
   - Implementation of comments is a good practice for documenting code, so more comments could have been added to help

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.

   Positive Features:
   - The main program loop is very minimal and clean
   - Easy to read and parse through the code to see how the objects interact with each other
   - The dynamic creation and deletion of GameMechs and Player instances (new and delete) show proper memory management for these key objects.

   Negative:
   - Nothing wrong with the main program but again adding comments enhances the documentation of the code for developers who may not be familiar with the project

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:

   - Abstraction and encapsulation enable better data management.
   - Enhanced reusability of code through objects and classes.
   - Improved maintainability and scalability.
   - OOD allows more natural mapping of real-world problems.

   Cons:

   - Higher initial complexity and learning curve.
   - Potential for increased memory and processing overhead.
   - OOD can introduce unnecessary complexity for simple problems.

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.

   Observations:

   - The code lacked comments. Which are crucial for explaining the purpose and functionality of code segments, especially for complex logic.

   Improvements:

   - Add comments to explain the purpose of functions, the logic within complex code blocks, and the role of key variables.

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.

   Observations:

   - The code follows good indentation and formatting in general. Overall it's well structured and organized. There are no issues in this area other than adding extra comments to have better documentation

   Improvements:

   - Make sure you use consistent indentation levels throughout the code. To avoid confusion, especially in nested structures.

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)

   Observations:
   - The game runs smoothly and the draw screen refreshes properly, with the snake growing properly with a reasonable delay
   - The snake collision detection works well
   - The game-over flag works properly when the snake collides with itself
   - The border wraparound works properly

   The game runs smoothly and as expected overall. There's no technical feedback.

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.

   There is no memory leak. The cleanup function correctly deletes all members created on the heap. Destructors are also implemented correctly in the necessary class implementations. This ensures that there's no memory language

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 collaborative work experience was good as it helped with managing the workload and finishing the code efficiently. Downsides would include having to meet up in person to work on certain aspects of the code to ensure the smooth integration of all parts of the code. One way to overcome this could be to use git effectively. Such as working in branches and merging the code at the end so that partners can work independently if possible.