**COMPENG 2SH4 Project – Peer Evaluation [25 Marks]**

Your Team Members floraa1(Anjali Flora) \_\_\_\_\_srikas15 (Sarangi Srikantha)\_

Team Members Evaluated Silver Owl: shil41 pan20

Provide your genuine and engineeringly verifiable feedback. Ungrounded claims will lead to deductions. Completing the peer code evaluation on time will earn your team a total of **25 marks**. Do not exceed 2 paragraphs per question.

**Peer Code Review: OOD Quality**

1. **[3 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.

It is moderately clear how the objects interact with each other, however there are some areas in the design that could be modified to be easier to understand. The initialization process is straightforward as myGM is initialized and dynamically allocated first to set the board, while myPlayer is initialized and dynamically allocated afterwards, with a reference to myGM, clearly showing that that Player depends on GameMechs for context. Another aspect that’s easy to understand is the food generation through myGm->generateFood(myPlayer->getPlayerPos()). This part is clear on how the objects depend on each other, as the design has it so that it takes in the player’s current position and ensures that the new food that gets generated isn’t at the same position as the player. However, a critique I have is about the myGM->setPlayerRef(myPlayer) feature as the code will normally have Player depend on GameMechs for the game board context, which makes the reciprocal dependency less intuitive. It suggests that GameMechs actively manipulates or observes the Player, but the exact interaction between them isn’t immediately clear. This bidirectional dependency reduces the flexibility of the code and makes it harder to extend.

1. **[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** | **Cons** |
| **C++ OOD approach** | * Each class (Player, GameMechs, objPos, and objPosArrayList) encapsulates an important aspect of the program’s functionality. * Clear separation of responsibilities between each class allows each to be tested and developed independently of one another (changes to one of the classes won’t affect the others). * Using clear class definitions with descriptive member functions make the code easier to read and understand. | * The tight coupling between the classes (specifically Player and GameMechs) make the code rigid and difficult to test in isolation. * Overusing raw pointers can increase the risk of memory leaks, dangling pointers, or double deletions. * Some classes handle multiple responsibilities, such as GameMechs manages game rules, board state, player references, and input. The Player manages position and movement but indirectly interacts with the game rules. This makes the classes harder to understand. |
| **C procedural design approach** | * Separated functions such as Initialize, GetInput, RunLogic, DrawScreen, and CleanUp divide the program into logical parts that makes the code easier to understand and debug. * Certain game components and input handling are encapsulated in separate functions which reduce redundancy and easier to debug. * With preprocessor constants being defined the game’s dimensions/parameters can easily be adjusted. | * Code relies heavily on global variables which increases coupling between functions which can make debugging more complex as state changes can occur unpredictably. * Functions are designed with specific use cases tied to global variables, making them less reusable. * enum dir and RunLogic used in PPA 3 implements a simple finite state machine but the design is unextendible. Meaning adding new movement would cause there to be changes in multiple places, increasing the complexity. |

**Peer Code Review: Code Quality**

1. **[3 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.

Yes, the code provides sufficient comments in areas of the design that makes the code a bit harder to interpret, and it also provides sufficient self-documenting coding style throughout the code. For example, in the draw screen function of the Project.cpp file, they included nested for loops to print out the contents of the game board and to check the player’s position. Without the comments, this would’ve made the code a bit difficult to follow along, however the sufficient amount of comments they implemented to break down each part and their use of spacing makes it much easier to follow along. They display this to all the other cpp files as well making their code very easy to understand.

1. **[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.

Yes, the code is very good in terms of readability as it provides a good amount of spacing between each section of the code and the right amount of indentation, with lines under for loops or if statements. This formatting makes the code very easy to follow along and understand. For example, in their checkSelfCollision() method they included an adequate amount of spacing between the for loop and if statement, as well as indentation to the if statement under the for loop, and the return true under the if statements. This makes it very distinct and clear that the if statement is checked under each iteration, and the return true only occurs if the conditions are met.

**Peer Code Review: Quick Functional Evaluation**

1. **[3 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 technical user feedback)

The Snake Game provides a generally smooth and enjoyable experience. However, there are a few areas that could benefit from improvement. One minor issue is the absence of an exit key, which would allow players to end the game early if they choose, without waiting for a loss condition. This feature could be added to the switch case within the updatePlayerDir function in the player.cpp file to handle a key press for exiting the game.

1. **[3 marks]** Does the Snake Game cause memory leak? If yes, provide a digest of the memory profiling report and identify the possible root cause(s) of the memory leakage.

The Snake Game does not cause any memory leak.

**Project Reflection**

Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team’s implementation in addition to your own, reflect on the following questions:

1. **[3 marks]** Do you think the compound object design of objPos class is sensible? Why or why not?

In our objPos.h file we included the move constructor and move assignment operator to adhere to rule of six. We noticed that the other team did not do this and adhered to the minimum 4 rule. We think the unusual objPos class design with the additional Pos struct is sensible because it holds the x and y coordinates which represent a specific coordinate on the 2D plot. This is helpful because it can be used in various classes that require the x-y coordinates.

1. **[4 marks]** If yes, discuss about an alternative objPos class design that you believe is relatively counterintuitive than the one in this project. If not, explain how you’d improve the object design. You are expected to facilitate the discussion with UML diagram(s).

An alternative objPos class design, which may seem counterintuitive, involves introducing a separate Pos class. The objPos class would then inherit from the Pos class. The Pos class would handle all aspects of the 2D grid layout, while the objPos class would focus on managing the symbol or object placed on that grid.

A blue screen with black text

Description automatically generatedA blue screen with black text

Description automatically generated