COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members Nabiha Sartaj Insiya Khan

Team Members Evaluated Null Pointers

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.

   Yes, the main code has a very organized structure that is easy to follow through. You can easily tell the role of each class and how they interact with each other. The run logic in the main is also very clear and concise. Furthermore, while the DrawScreen(); was lengthy, the use of comments helped make it easier to read and understand.

2. **[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:
   - It is more organized since certain tasks are split up into classes
   - It has the option to make certain variables and functions private which ensures that they can only be changed through controlled interfaces
   - Once the code is written, it is much easier to update since everything is separated into classes. Only one class would need to be updated since everything is not liked together.

   Cons:
   - C does not require the addition of setter and getters, making it easier to finish the code. There are also no private variables that you have to work around.
   - There are more files, this could make debugging more difficult.

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.

   The comments provide clear context for the updated/added functions. The naming of functions also provides good context for the person reading the code. However, there are some places, such as in

player::snakeMovement, where additional comments could help improve the readers understanding of the code.

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.

   The indentations and spacing between functions is very consistent throughout the code, making user readability very easy. The team also added newlines where necessary.

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 code runs very smoothly without any bugs and works like how a usual snake game should behave. The game is very responsive, once I hit food the snake grows right away, and the player follows every move I input.

2. **[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.

```
0 unique,     0 total,      0 byte(s) of leak(s)
0 unique,     0 total,      0 byte(s) of possible leak(s)
```

   0 bytes of 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?

   Yes the objPos class is sensible because it can determine information about the position of the player (x and y coordinates) without the symbol. This makes it easier to use the struct Pos without affecting the objPos class. However, the implementation of objPos class heightened the difficulty of the project, increasing the risk of memory leaks.

2. **[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).

To make the objPos class design relatively counterintuitive than the one in this project, we can make obj pos have more responsibilities than just finding the position of the player. For example, we can make objPos draw its own contents on the screen, and indicate its own collisions with items on the board.

ObjPos
-X: int -Y: int -Symbol: char
+objPos(); +objPos(int xPos, int yPos, char sym); +setObjPos(objPos o); +setObjPos(int xPos, int yPos, char sym); +getObjPos() const; +getSymbol() const; +isPosEqual(const objPos* refPos) const; +getSymbolIfPosEqual(const objPos* refPos) const +objPosDrawContents(char sym); +objPosIndicateCollision(const objPos &other);