COMPENG 2SH4 Project – Peer Evaluation [30 Marks]

Your Team Members Thevenin (shinj16/ shinj16-fall2024, roht/roht-2024f)

Team Members Evaluated cool beans (mouzakik/mouzakik, rivera8/rivera8-mcmaster)

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 **30 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, we can easily interpret how the objects interact with each other in the program logic through the code. For example, they used different objects to implement the correct drawScreen() and runLogic() functions. They used correct methods of each class to generate correct results.

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.

   The pros of the C++ OOD approach in the project is that we can reuse the code and the objects across different parts of the project without having to implement/create variables everytime. We can also debug and test our code more easily, as we can modify/fix certain methods without having to modify the whole project. One of the cons of the C++ OOD approach is that it is challenging to learn/adapt at first. For example, our team had a little bit of difficulty transforming some PPA3 features into our project. We also learned that the C procedural is much straightforward and takes much less time to develop things.

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 offers sufficient comments to help users understand the code functionality more efficiently. Each logic/code block has a comment that explains the overall functionality of it.

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.

   Yes, the code follows a good indentation and sensible white spaces for better readability. The format is clear and is in style of what we learned/did throughout the course.

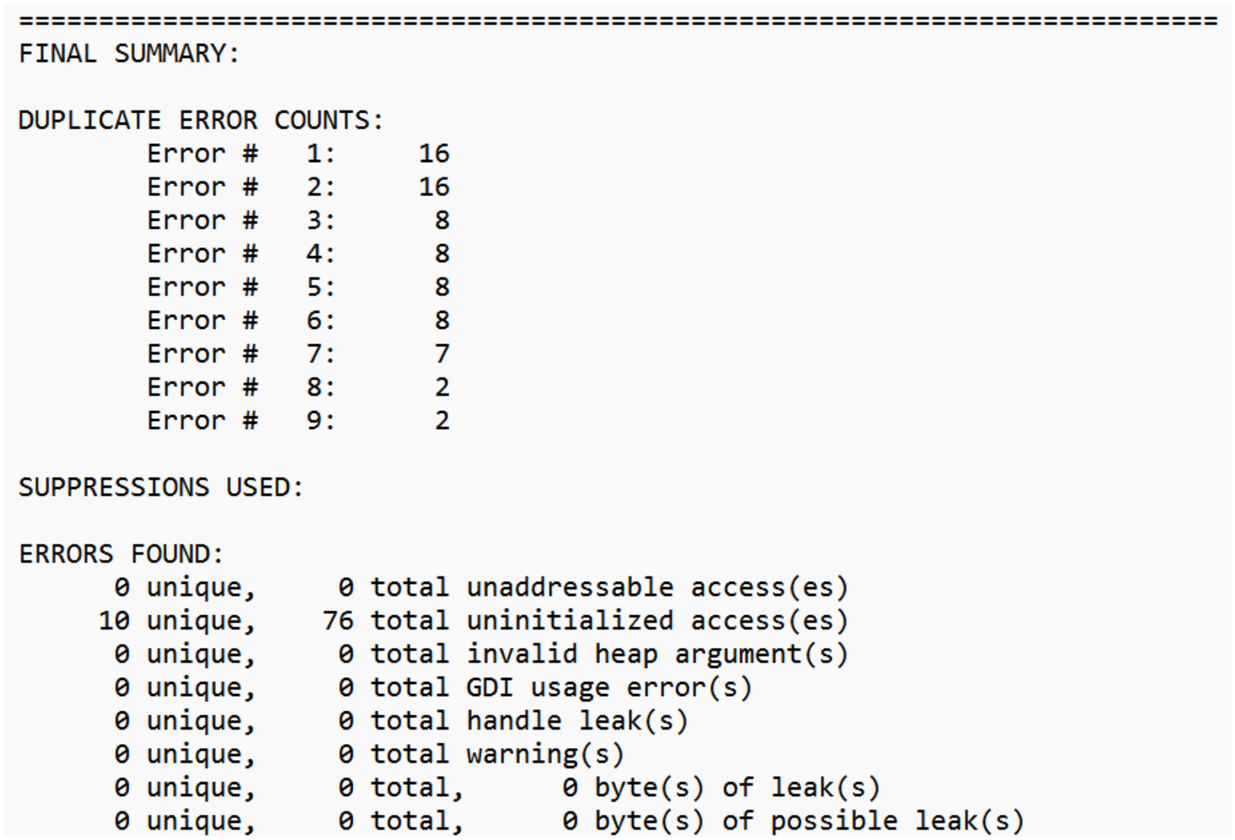
Peer Code Review: Quick Functional Evaluation

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)

   Yes, the snake game offers smooth, bug-free playing experience. There are no buggy features and the game runs and ends well with expected results.

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.

   No, the snake game does not cause many memory leak.



```
============================================================================
FINAL SUMMARY:

DUPLICATE ERROR COUNTS:
        Error #   1:     16
        Error #   2:     16
        Error #   3:      8
        Error #   4:      8
        Error #   5:      8
        Error #   6:      8
        Error #   7:      7
        Error #   8:      2
        Error #   9:      2

SUPPRESSIONS USED:

ERRORS FOUND:
      0 unique,     0 total unaddressable access(es)
     10 unique,    76 total uninitialized access(es)
      0 unique,     0 total invalid heap argument(s)
      0 unique,     0 total GDI usage error(s)
      0 unique,     0 total handle leak(s)
      0 unique,     0 total warning(s)
      0 unique,     0 total,      0 byte(s) of leak(s)
      0 unique,     0 total,      0 byte(s) of possible leak(s)
```

Project Reflection

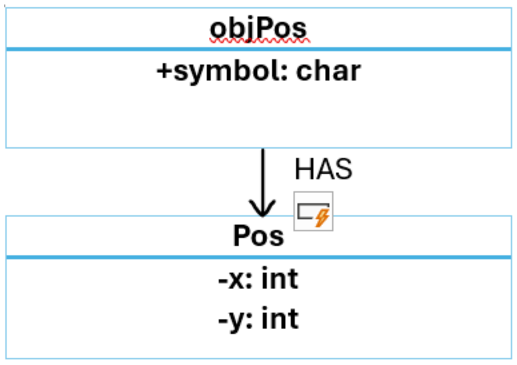
Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team's implementation in addition to yours, reflect on the following questions:

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

   No, we think we should have Pos as a separate class so that we can implement various specific methods for Pos. Having a separate class that manages Pos would allow us to implement more advanced/specific features to Pos. For example, in the future we could make Pos/coordinate calculations or transformations in the separate class.

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.

We should consider separating the Pos struct to a separate class and having the objPos class have that Pos object as its data member. Separating this into its own class improves the object design as it promotes single modularity. This also would reduce objPos's complexity and make it easier to maintain/debug.


objPos
+symbol: char
HAS
Pos
-x: int
-y: int