COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members woldeyed oreild6

Team Members Evaluated engs3 weij64

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 code implements a strong OOD of the snake game. We can easily interpret the main logic in the program and understand how objects interact with each other. The code is very organized and is efficiently written.

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 C++ Object-Oriented Design approach is particularly useful for larger programs, like in this project, because it organizes code into modular classes. This modularity makes it easier to break down tasks, manage different actions, and maintain or upgrade the code throughout iterations. Adding new features or making changes is more efficient since you can avoid duplicating blocks of code, which also demonstrates its strong reusability. Additionally, OOD simplifies collaboration in large projects by allowing team members to focus on different components independently.

   However, debugging in OOD can be challenging because errors might be buried deep within classes and only appear in specific functions. We experienced this issue and realized how crucial constant testing is to catch bugs early. Also, OOD in C++ requires more effort to plan and set up in contrast to the C procedural design we used in PPA3 which is much easier to understand and implement. It requires less planning and offers better readability, making it a more straightforward option for smaller, less complex tasks. There is also increased risk of memory leakage if not careful.

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 code is very well commented, providing clear explanations that make it easy for the reader to understand the purpose of each function and its implementation. For example, their use of comments to summarize and explain how player was defined was helpful in providing clarity.

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 code is well-indented, making it easy to follow loops, conditionals, and functions. It also uses whitespace and newlines effectively, making the code clear and well-organized.

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 game runs smoothly with clear instructions for movement and the scoring system. There are no noticeable bugs, and the game is well-implemented.

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, there is no memory leakage.

```
ERRORS FOUND:
     0 unique,     0 total unaddressable access(es)
     7 unique,   102 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)
ERRORS IGNORED:
    19 potential error(s) (suspected false positives)
        (details: C:\DrMemory-Windows-2.6.0\drmemory\logs\DrMemory-Project.exe.28320.000\potential_errors.txt
    28 unique,    28 total,   7454 byte(s) of still-reachable allocation(s)
        (re-run with "-show_reachable" for details)
Details: C:\DrMemory-Windows-2.6.0\drmemory\logs\DrMemory-Project.exe.28320.000\results.txt
```

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?

We believe the compound object design of objPos was a smart choice because it enhances the class's modularity, making the code more flexible and easier to maintain. This design allows for the seamless addition of new components, improving code maintenance in the project. The separation of dimensions also helps keep the code organized. There are a lot of position related aspects of the project too so it ensures that programmers can easily access dimensions as intended.

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).

   By placing the x and y coordinates within the private scope of the class, we can access these values through other member functions. This direct access would be an easier design and allowing simpler approach.