COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members ___ Jiangr42, li3081

Team Members Evaluated ____ahmem121, yangj351

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.

In terms of positives, the initialize and run logic functions look very straightforward. Objects are initialized in the main function on the heap and deleted at the end in the cleanup function. The player object and food objects are well encapsulated so when they appear in project.cpp it is usually inside concise and straightforward ways. The member functions and variables are intuitively named. We especially like how the copy constructor and copy assignment constructors were included in the objects even if they are not used, this adheres to OOD conventions.

As for areas of improvement, the design of the gameMechs object can be made to follow OOD principles more. For example, the exit flag for the main loop is still kept as a global variable in main. This should have been encapsulated inside GameMechs. The gameMechs object also lacked ways to interact with the player object. Which lead to there being a huge chunk of code in the drawscreen function in main. All the logic there should have been encapsulated in GameMechs but could not because gameMechs did not have a reference to the player object. This design choice also lead to redundancy in the code. Every cycle there is a nested loop which has to go through and check if the empty spaces are on top of the snake, a loop to check for if the snake has collided with itself, and a loop to check for if the snake has eaten food. This leads to slow code execution and there being (at least on our laptops) constant flickering on the screen due to the drawscreen function executing too slowly.

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.

OOD pros:

- Descriptive names helps readers and future writers understand what each code does. E.x. playerObject->movePlayer();
- Easier to collaborate with other people on
- More scalable and modular
- Less pollution of name space

Cons:

- Long development times

- Requires more files and lines of code
- More complex to read especially when code is not complex enough to warrant such a complex system
- When implemented incorrectly the whole code becomes a jumbled mess of references across different files, making it harder to debug in that case

Procedural pros:

- Fast to develop
- More concise in many cases
- Less files required
- Better for small scale programs

Cons:

- Hard to collaborate on since editing the same file can lead to merge conflicts, and each programmer must understand the purpose of every function inside the program
- Hard to scale
- Potential redundancies when multiple instances of similar data needs to be processed
- Pollution of namespace when there are too many functions

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. There is an abundance of comments from this group. Comments and descriptive function names and variables make it very easy to understand the procedures of the code. One area to improve is if possible, use a block of comments, as the differently placed comments at the end of code lines makes it hard to read.

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 maintains great readability. A small criticism is that there are extra spaces where it doesn't seem necessary, such as the mass of codes in drawscreen in Project.cpp. But this is a small issue.

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 is extremely laggy. There are no bugs, but, at least for us, the game flashes often, and can make playing the game very uncomfortable. We suspect this is because the code runs a nested loop for every "pixel" to check if the snake/food is there or not. We recommend a different approach to the drawscreen logic, where the board is stored as a attribute inside gameMechs, maybe a 2D Array. Then, the entire board can be "prerendered" before being displayed. This way when drawing the screen, all that is required is to print the character at [i][j] of the board. Running Logic between displaying each "pixel" of the game slows it down and introduces uncomfortable lag. Otherwise, there are no Bugs to be seen.

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 memory leaks. The program has 2 bytes of leak stemming from MacUILib.

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?

I do not think Pos inside objPos is a sensible idea. Using Pos struct increases code length when accessing the data and is confusing when writing and reading code. It is especially nightmarish when debugging, and since it cannot be named a sensible name when using the Pos feature, it makes debugging and writing slower and harder. Instead of getX or getY, it is object.pos->x and this can be confused for, for example, heap objects and arrays of objects.

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.

We think having x and y be private member variables will achieve the same result. The struct in this case is redundant and provides no advantages other than increased compatibility with our PPA3 code in C. This way, sensible getter names can be used to describe the code more effectively, without confusing arrow operators.