COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Sanya Sharma and Anushka Menon

Team Members Evaluated Yiheng Hu and Steven Zhang

Provide your genuine and engineeringly verifiable feedback. Ungrounded claims will lead to deductions.

Part I: OOD Quality

1. **[6 marks]** OOD is about sensible code modularization. Looking at the header files of each object, can you easily interpret the possible behaviours of the objects involved in the program, and how they would interact with each other in the program? Comment on what you have observed, both positive and negative features.

Analyzing the potential behaviors and interactions of the objects based solely on the provided header file names offers some insights into the structure and functionality of the program. However, without access to the actual content within these headers, the analysis remains limited to assumptions and logical inferences based on standard conventions and naming conventions. The proper declaration of the header files made it easy to identify the objects included and due to our prior knowledge of the code we were able to understand the reasoning behind including it, but it would be hard for a person who sees the code without knowledge. A description of each header file and the purpose would have enhanced the understanding of the code.

2. **[6 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.

By reading the main program loop we see it manages the interactions between different objects (GameMechs, Player, Food, objPos, objPosArrayList). It initializes the game, updates player movements, manages screen drawing, and controls game logic. The use of global pointers facilitates communication between these objects, but there's a risk of tight coupling. While the code demonstrates modularity and encapsulation, reducing global state reliance and enhancing abstraction could improve its flexibility and maintainability.

3. **[5 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.

In the C++ OOD approach, utilizing classes like GameMechs and Player helps keep things coordinated and straightforward. You are able to easily add in new updates at a later time thanks to the game's dynamic updates to the game state during the main loop, which make the game flexible and adaptable. However, there are challenges, such as using global variables, which might make the code too connected and cause confusion between the drawing and game rules in one function, which can get a little messy.

On the other hand, PPA3's C procedural design has a straightforward control flow and is simple to follow. However, making adjustments might be difficult, and you might end up writing the same code over and over, making it difficult to maintain and enhance the program in the long run.

Part II: Code Quality

1. **[5 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 provided code lacks comments or self-documenting code to explain the functionality and logic of various sections adequately. While the function names offer some indication of their purpose, detailed comments within the code are crucial to understand the intricate interactions between different objects and the specific logic implemented in each function.

2. **[4 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 provided code demonstrates reasonable indentation and uses white spaces to some extent for readability. However, it could benefit from consistent formatting and improved newline usage to enhance readability further. Improvements to enhance code readability, Consistent Indentation, Whitespace Usage, Newline Formatting, Grouping Related Code, Standard Formatting Guidelines, Improving whitespace usage, strategically inserting newlines, and maintaining consistent indentation levels throughout the codebase can significantly enhance readability. These practices can make the code more accessible and easier to understand for both the current developers and potential contributors in the future.

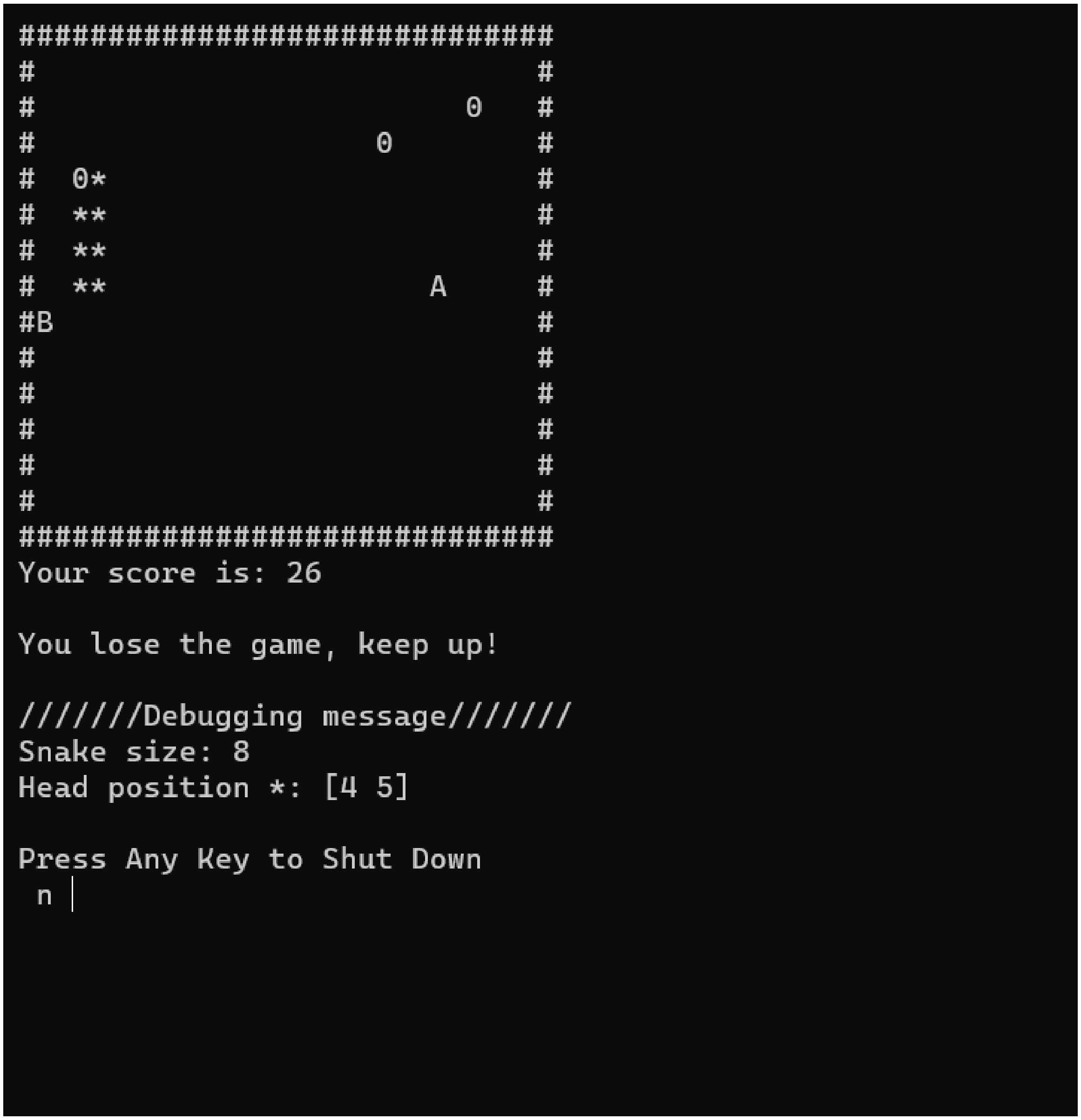
Part III: Quick Functional Evaluation

1. **[8 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 a technical user feedback)

The game provided a bug free experience and ran as expected. The team also add the above and beyond feature with 2 additional letters with 2 different scores when ate by the snake. 2 suggestions we would like to give are:

1. It would be great if the team added a display to show the gamer the stats of the game. Having the score plus the points earned with a certain letter would be great as when we were playing the game it took us a long time to determine which letter consisted how many scores.

2. A game instruction on the display would have enhanced the experience and provide the gamer which more information on how to win the game.



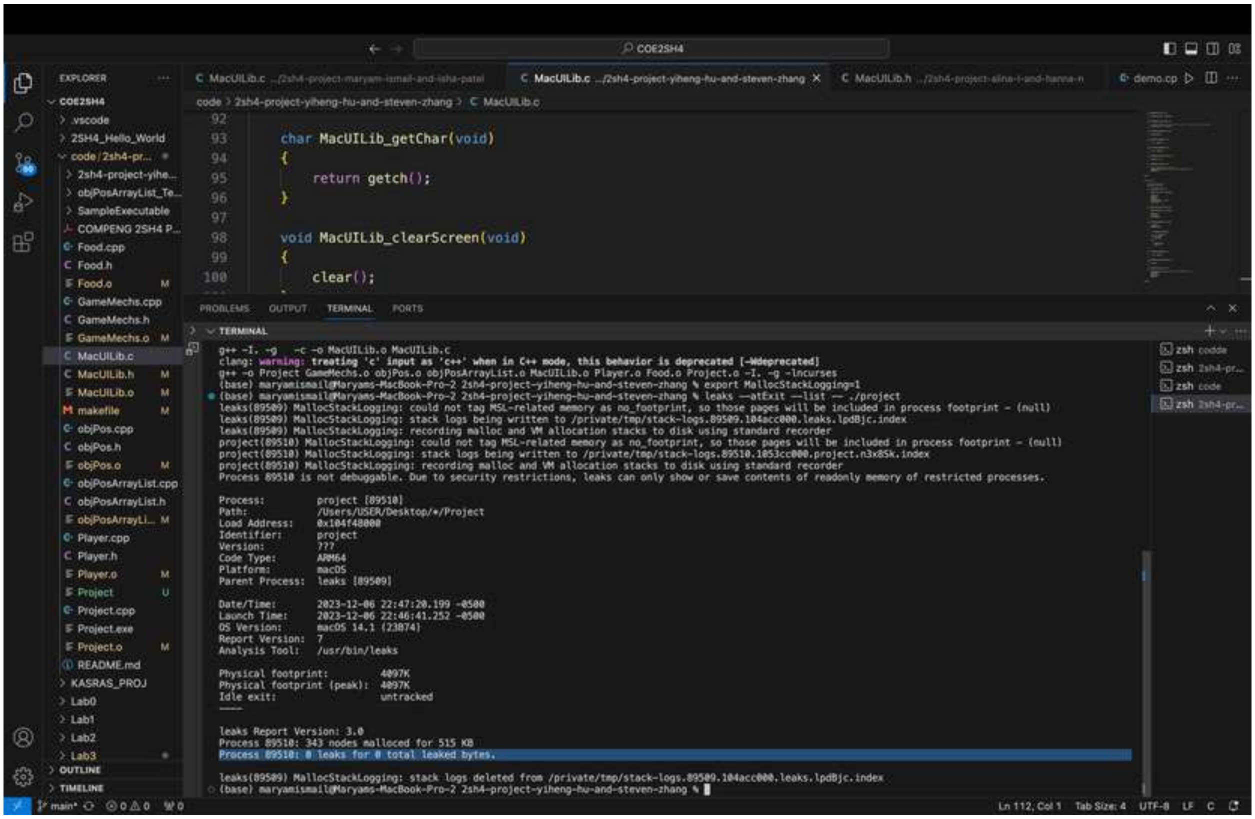
```
##############################
#                            #
#                        0   #
#                   0        #
#  0*                        #
#  **                        #
#  **                        #
#  **                  A     #
#B                           #
#                            #
#                            #
#                            #
#                            #
#                            #
##############################
Your score is: 26

You lose the game, keep up!

///////Debugging message///////
Snake size: 8
Head position *: [4 5]

Press Any Key to Shut Down
 n
```

2. **[6 marks]** Does the Snake Game cause memory leak? If yes, provide a digest of the memory profiling report and identify the possible root cause of the memory leakage.



```
char MacUILib_getChar(void)
{
    return getch();
}

void MacUILib_clearScreen(void)
{
    clear();
```

```
g++ -I. -g   -c -o MacUILib.o MacUILib.c
clang: warning: treating 'c' input as 'c++' when in C++ mode, this behavior is deprecated [-Wdeprecated]
g++ -o Project GameMechs.o objPos.o objPosArrayList.o MacUILib.o Player.o Food.o Project.o -I. -g -lncurses
(base) maryamismail@Maryams-MacBook-Pro-2 2sh4-project-yiheng-hu-and-steven-zhang % export MallocStackLogging=1
(base) maryamismail@Maryams-MacBook-Pro-2 2sh4-project-yiheng-hu-and-steven-zhang % leaks --atExit --list -- ./project
leaks(89509) MallocStackLogging: could not tag MSL-related memory as no_footprint, so those pages will be included in process footprint - (null)
leaks(89509) MallocStackLogging: stack logs being written to /private/tmp/stack-logs.89509.104acc000.leaks.lpdBjc.index
leaks(89509) MallocStackLogging: recording malloc and VM allocation stacks to disk using standard recorder
project(89510) MallocStackLogging: could not tag MSL-related memory as no_footprint, so those pages will be included in process footprint - (null)
project(89510) MallocStackLogging: stack logs being written to /private/tmp/stack-logs.89510.1053cc000.project.n3x8Sk.index
project(89510) MallocStackLogging: recording malloc and VM allocation stacks to disk using standard recorder
Process 89510 is not debuggable. Due to security restrictions, leaks can only show or save contents of readonly memory of restricted processes.

Process:         project [89510]
Path:            /Users/USER/Desktop/*/Project
Load Address:    0x104f48000
Identifier:      project
Version:         ???
Code Type:       ARM64
Platform:        macOS
Parent Process:  leaks [89509]

Date/Time:       2023-12-06 22:47:20.199 -0500
Launch Time:     2023-12-06 22:46:41.252 -0500
OS Version:      macOS 14.1 (23B74)
Report Version:  7
Analysis Tool:   /usr/bin/leaks

Physical footprint:         4097K
Physical footprint (peak):  4097K
Idle exit:                  untracked
----

leaks Report Version: 3.0
Process 89510: 343 nodes malloced for 515 KB
Process 89510: 0 leaks for 0 total leaked bytes.

leaks(89509) MallocStackLogging: stack logs deleted from /private/tmp/stack-logs.89509.104acc000.leaks.lpdBjc.index
(base) maryamismail@Maryams-MacBook-Pro-2 2sh4-project-yiheng-hu-and-steven-zhang % 
```

0 Memory leak detected

Part IV: Your Own Collaboration Experience (Ungraded)

1. Tell us about your experience in your first collaborated software development through this project – what was working and what wasn't. If you are a one-person team, tell us what you think may work better if you had a second collaborator working with you.

   In our first collaborated software development project, several aspects worked well while some presented challenges. The division of labor among the 2 members was effective, allowing us to focus on tasks based on our strengths and skills. This approach enhanced productivity and ensured efficient progress. Challenges arose in communication. Inadequate or ineffective communication among us led to misunderstandings, delays, and conflicting implementations. Misaligned objectives and differences in understanding project goals also impacted the project's direction. Overall, it was a great experience as this is a standard working environment in the industry that we have to get used to.