COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Nathan and Hunter

Team Members Evaluated David and Talha

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.**

   The possible behaviours of the objects involved in the program are easily interpreted by looking at the header files. Descriptive member names like foodBucket and generateFood clearly indicate that the two members interact with each other and generateFood creates food for the foodBucket. Also, it's clear how the GameMechs and Food classes interact with the Player class because a pointer to each class is needed for the Player constructor. One piece of criticism I have for the header file is the name for the FoodReference member in Player. It would be easier to identify it's purpose if it had a similar name to the mainGameMechsRef member, such as mainFoodRef. This would indicate that it serves essentially the same purpose as mainGameMechsRef, to point to a specific object of GameMechs/Food and allow for a personalized game.

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.**

   For the most part, the main project loop in the project.cpp file is easily interpreted, and you can see how the myGM, myPlayer, and myFood objects interact with each other easily. Due to the descriptive member functions of each class that are used in the main loop, it is easy to interpret the logic of the game and how the program runs. One critique I have is for the initialization routine that runs several for loops to initialize the game border because it is closer to procedural programming than OOD and is less intuitive for understanding the program. As a suggestion, instead of having close to 20 lines dedicated to creating the border, a GameMechs member function could be called that initializes the border and replaces the 20 lines of code.

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.**

   C++ OOD:

   Pros:

- Greater organization, less code to scroll through
- Reusability, inheritance, decreased redundancy in programs, easier to update features
- Restricted access with private data members
- The ability to create more complex projects in a more organized and sensible way

Cons:

- Sometimes isn't needed for simpler projects
- Challenging to learn, results in more files and lots of underlying code
- Can perform slower than C code
- Due to the complexity, can be harder to debug

C Procedural:

Pros:

- Simpler than OOD
- Can perform quicker than OOD
- Can be more useful for smaller projects or processors to use procedural C code instead of OOD

Cons:

- Less reusability, more redundancy, can be annoying to maintain and update small features
- Harder to use and maintain for complex projects that require a lot of code
- Many global variables which use memory

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.**

After inspecting the class files as well as the main project file, though the code does provide comments, as a team we feel that overall, the code lacks commenting to provide further clarity and understanding. Furthermore, there are sections of code that seem to have comments from the development stages; questions, and reminder statements. On class files, the typical commenting style was expressed as one or two lines at the beginning of each function. In order to improve self-documenting coding style and understanding of code functionality, we would consider commenting before the start of each loop explaining the loop functionality and end goal.

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 overall code provides good indentation, sensible white spaces, and newline formatting for better understanding and readability. The only shortcoming, we observed is in the food class, where the

programmers initialized temp positions and food bucket items, we think that this might be better to separate them with a blank line between each line of code for better clarity as the cluster of code might be confusing to the naked eye.

```
Food::Food(){

    srand(time(NULL));

    objPos tempPos, tempPos_1, tempPos_2, tempPos_3, tempPos_4;
    tempPos.setObjPos(-1, -1, 'o');
    tempPos_1.setObjPos(-2, -2, 'o');
    tempPos_2.setObjPos(-3, -3, 'o');
    tempPos_3.setObjPos(-4, -4, 'o');
    tempPos_4.setObjPos(-5, -5, '$');
    foodBucket = new objPosArrayList();
    foodBucket->insertHead(tempPos);
    foodBucket->insertHead(tempPos_1);
    foodBucket->insertHead(tempPos_2);
    foodBucket->insertHead(tempPos_3);
    foodBucket->insertHead(tempPos_4);

}
```

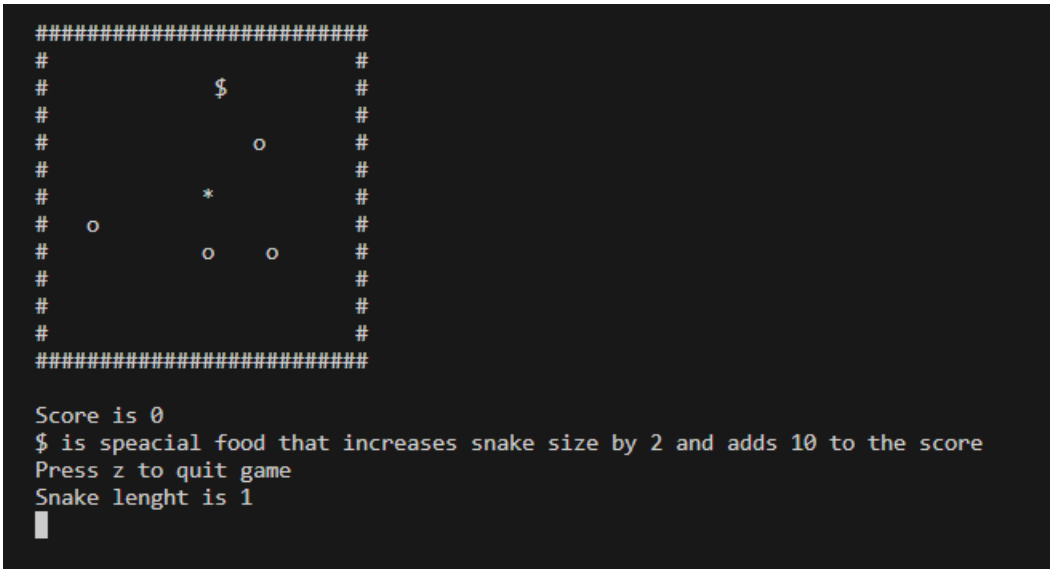
Part III: Quick Functional Evaluation

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 snake game offers a smooth and bug free playing experience, however the food is only generated on columns 1-18 and rows 1-10. This is because in their generateFood function, they don't call the boardSizeX and boardsizeY getter functions, so the places that food can be generated is limited to only column size 18 and row size 8.

```
45          x1 = (rand() %18)+1;
46          y1 =(rand() %8) +1;
```

One way to debug this is to allow for a pointer to a GameMechs class inside of the Food class, that way you can call the getter methods for boardSizeX and boardSizeY, then replace 18 with GetBoardSizeX - 2 and 8 with GetBoardSizeY – 2.



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.**

   The snake game does not provide any leak as shown below, all allocated members were deallocated properly within the class files.

```
Process:         Project [81129]
Path:            /Users/USER/Documents/*/Project
Load Address:    0x1041a0000
Identifier:      Project
Version:         0
Code Type:       ARM64
Platform:        macOS
Parent Process:  leaks [81128]

Date/Time:       2023-12-05 12:02:36.095 -0500
Launch Time:     2023-12-05 12:02:31.372 -0500
OS Version:      macOS 13.3.1 (22E261)
Report Version:  7
Analysis Tool:   /Applications/Xcode.app/Contents/Developer/usr/bin/leaks
Analysis Tool Version:  Xcode 14.3.1 (14E300c)

Physical footprint:         3217K
Physical footprint (peak):  3217K
Idle exit:                  untracked
----

leaks Report Version: 3.0
Process 81129: 311 nodes malloced for 233 KB
Process 81129: 0 leaks for 0 total leaked bytes.
```

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.**

As a team of two, we found our first collaborated software development experience to be knowledgeable and advancing. Together we created an open-ended communication system, where we split up our roles as different developers while making sure one another understood the code of the other developer. Furthermore, our very flexible communication method allowed us to help each other with a different perspective through different programming difficulties we were stuck on. However, some things we feel we could have improved as a team is perhaps meeting up in person and conducting more calls to allow for greater communication and setting further clarity for goals we had at different stages of our project.