COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members	Elisabeth Mark (marke3)	Sapna Suthar (suthas4)
Team Members Evaluated	Christina Bridges (bridgc1)	Sonia Parekh(pareks5)

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.**

When examining the main logic in the main program loop, we found it easy to understand how the objects interacted with each other through the code. However, if we had no prior knowledge of the assignments, more codes would be needed to understand their logic and how it is being implemented. This group added comments before big chucks of code, but that didn't always explain their logic as it was quite simple comments for a large amount of code. As well, the exit and lose print statements are in cleanup, which is a function reserved for deallocating memory. Instead these statements should be moved to drawScreen() to maintain the logic of the functions.

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.

Pros:

- Less repetition in making objects
- More intuitive (removes global variables)

Cons:

- Harder to understand
- More files to work on (file per class)
- More use of pointers
- Can get complicated especially if classes are built through other classes of objects

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 does not offer sufficient comments, making it hard to understand. When looking at the code, the main file, Project.cpp has comments, but they are often simple one line comments for large blocks of code. In addition, all the class functions have little to no comments explaining what the methods are for or what is occurring inside them.

The easiest way to fix this would to add more comments, as well as make the existing comments more in depth. This would make it easier for any onlooker to understand the logic of the code, which is incredibly necessary in all class files.

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 properly indented, uses white spaces and deploys newline formatting for better readability. There are no changes we can suggest to improve in this area.

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)

For the most part, the game runs as intended, however we spotted a couple bugs.

For one, the random food generation x and y coordinates when starting up the game is the same. In the Food class in generate food(), we suggest srand(time(NULL)) should be placed in the for loop right before the generate of x and y coordinates. This is so that the for loop repeatedly calls srand(time(NULL)).

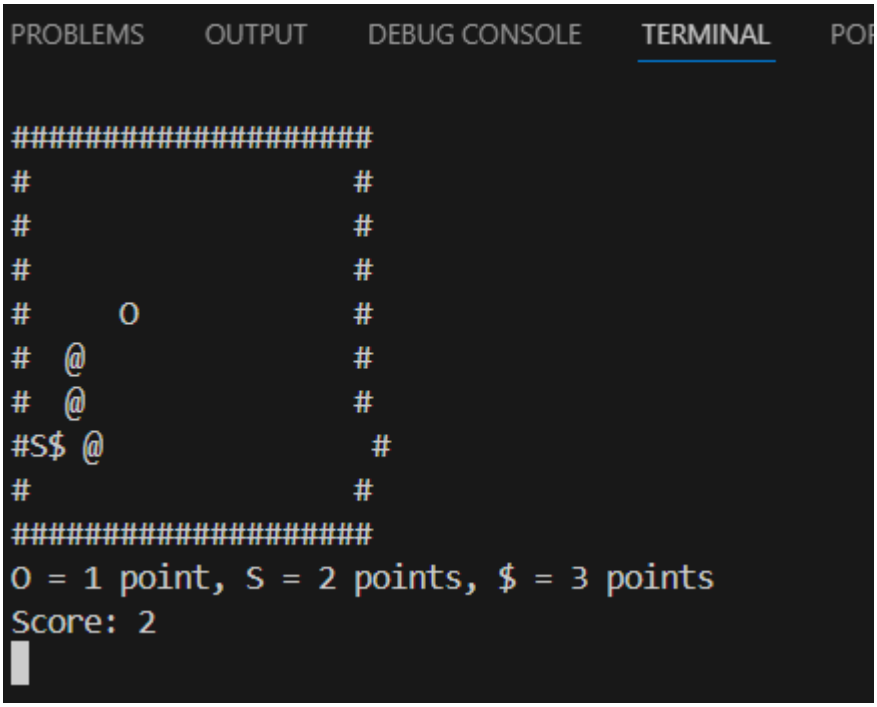
```
void Food :: generateFood(objPosArrayList& list, int x_size, int y_size)
{
    srand(time(NULL));

    objPos playerNow;
    objPos canFood; //candidate food
    int canX, canY; //candidate x & y coordinates for food location

    for(int i=0; i<foodBucket->getSize(); i++)
    {
        while (true){
            bool overlap = false;
            canX = rand() % x_size +1;
            canY = rand() % y_size +1;
```

i.e., move srand(time(NULL)) in the while loop before rand() is called.

In addition, the boarder is shifted when the snake tries to eat a food item that is next to another food item (i.e., when the snake is about to “eat” the food, it then “dodges” the food and then goes back to the path it was in prior) (see screenshot below).



```
// if the current iteration matches the border location print '#'
// otherwise print a space
if (used == false)
{
    if (j == 0 || j == (boardX - 1) || i == 0 || i == (boardY - 1)){
       MacUILib_printf("%c", '#');
       }
    else{
          MacUILib_printf(" ");
       }
}
```

We think the root cause of this error is in drawScreen. This if statement is incorrect. Because the boolean used is the indicator of if the for loop has printed a character for the space in the boarder, it will mess up with the spaces in the else block. This is because the if else should be separated to two if statements and make used true in each segment. Below is the code we propose.

```
// if the current iteration matches the border location print '#'
// otherwise print a space
if (used == false)
{
    if (j == 0 || j == (boardX - 1) || i == 0 || i == (boardY - 1)){
       MacUILib_printf("%c", '#');
       used = true;
       }

}

if (used == false)
{
    MacUILib_printf(" ");
}
```

In this way, spaces and hashtags are checked independently, so a space is only put when needed and not just if a hashtag isn't being placed. Moreover, since the final if statement is the last line in the j for loop, there is no need for boolean used to equal true .

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, this game does not have any memory leaks.

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

DUPLICATE ERROR COUNTS:
        Error #   2:      2
        Error #   3:      2
        Error #   4:     61

SUPPRESSIONS USED:

ERRORS FOUND:
      0 unique,     0 total unaddressable access(es)
      4 unique,     6 total uninitialized access(es)
      1 unique,    61 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:
      7 potential error(s) (suspected false positives)
         (details: C:\DrMemory-Windows-2.6.0\drmemory\logs\DrMemory-cmd.exe.22480.000\potential_errors.txt)
      4 potential leak(s) (suspected false positives)
         (details: C:\DrMemory-Windows-2.6.0\drmemory\logs\DrMemory-cmd.exe.22480.000\potential_errors.txt)
     73 unique,   150 total,  28714 byte(s) of still-reachable allocation(s)
         (re-run with "-show_reachable" for details)
Details: C:\DrMemory-Windows-2.6.0\drmemory\logs\DrMemory-cmd.exe.22480.000\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?**

The compound object design of objPos class is not sensible. For the x and y coordinates, it is in struct. To use the objPos class, a struct pointer would need to be used to access the x and y coordinates, and a separate char symbol is declared as a variable. This is not intuitive as the variables are separated.

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).**

To improve the object design, we believe the struct should be annihilated and instead the x and y coordinates should be declared as variables like the symbol variable (as shown in the annotated UML diagram below). In turn, the user can code in a way where accessing the x and y variables are like accessing the symbol. One does not need to remember if the variable they wanted to access was through the struct or not.

Recommended Basic UML Design

For your reference, here is the more detailed UML design containing our recommended features. Your implementation can deviate from this recommended design, as long as the **RED** mandatory features are identical.