COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Kevin Adieb & Hemant Arora

Team Members Evaluated Saad & Shounak

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

Part I: OOD Quality

1. ***[5 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 methods in each header file have descriptive names and thus are easy to understand what they do. Each method also has descriptive parameters which explains what the input should be for that method. Possibly separating the food functionality of the game from the game mechanics class might have made the code more organized. Additionally, in the "player.h" file the function to return the list of player positions is called "getPlayerPos()", which may be misleading because it returns a list of positions instead of a singular position. A more appropriate name for that function is "getPlayerPosList()".

2. ***[5 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.*

Overall, the main logic is easy to understand, the functions and variable names are descriptive. For example, in the draw routine the use the variable "tempBody" to hold the body segment that is currently being printed. The variable name describes what the variable holds making it easier to understand what the for loop does. However, an "objPos" called "playerPosition" is instantiated in the "Initialize" function which is then never used in the program. Unused variables make the code slightly more confusing to understand.

3. ***[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.*

One benefit of OOD is that, through encapsulation A well written class allows other programmers to identify things that they should not touch (private data and methods), and identify things they should touch (public methods). For example, in the project I did not write the "objPos" class, however I needed to use it when developing the "Player" class. Just by looking at the header file I can see what methods I should be using (public methods). Additionally, the OOD approach makes it easer to reuse code, for example the "objPos" class was used again in the "Food" class. One disadvantage of the OOD approach is that it can also add unnecessary complexity. For example, in the “Player” class in the “updatePlayerDir” method instead of taking the input as a parameter, there is a reference to the “GameMech” class and then the “getInput” function is used to get the input. By adding dependencies, it makes it harder to understand what the code does as the result of “updatePlayerDir” is no longer dependent on just the inputs, its dependent on some external factor. Compared to the procedural design approach, where just by looking at the parameters you know what the function does.

Part II: Code Quality

1. ***[4 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.*

   Many of the files barely have any comments at all, and if they do, they are very minimal or are not relevant to the actual code, such as the comment on lines 4 and 5 in the “objPosArrayList.cpp”. There should have been more comments in the “Project.cpp” to explain how the general structure of the game works, especially in the DrawScreen function. The lack of comments makes big parts of the code hard to follow along, yet the variable and method names do help in showing what is involved or used in each line of code. Overall, more comments are needed to help read the code throughout the whole project. Given the fact that we too have created a similar program, it is easier for us to understand the code. If this was a random code given to us, it would have been a lot more challenging to interpret the code without the comments.

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.*

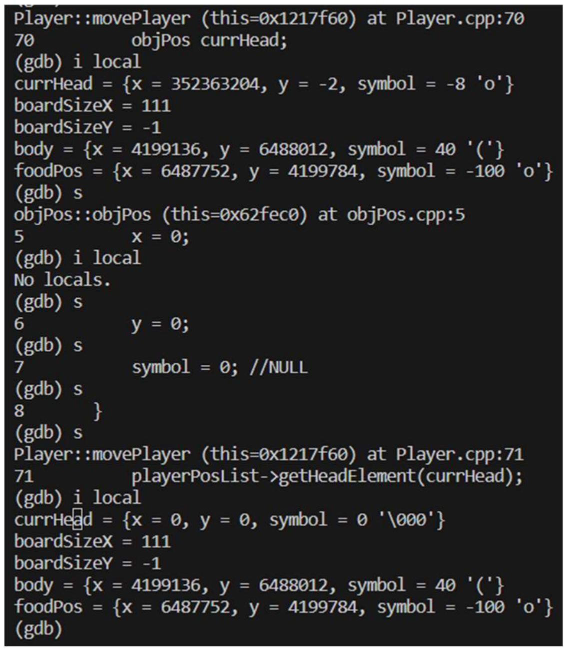
   Generally, the indentation and whitespaces are good, however the formatting is a little inconsistent in some places. For example, in “Player.cpp” at the end of a function before the final curly bracket there are a couple of empty lines. The number of empty lines seem to be random, and other files do not have them. Given that multiple people worked on this program it is natural for them to have different formatting preferences, one way around this is to decide on a formatting style in the beginning and then stick with it. Although this is a small error and does not affect the readability and functionality that much.

Part III: Quick Functional Evaluation

1. ***[6 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 program appears to be non-functional, when starting the game, the player is never printed. The root cause appears to be a break statement on line 103 in "project.cpp". Removing this restores functionality. We recommend the team to examine the player printing portion of the code and determine whether the break statement is necessary. After implementing this fix the program appears to work adequately. The snake moves fine, it eats the food and grows, the suicide condition works, and the exit button works. One additional bug was found after the user exits the program (suicide or button) after the "press any key to exit" screen is displayed, and the user presses any key (other than the power button) the game board is printed for a split second. Ultimately this does not really affect the playing experience. We would recommend the team examine their exit printing function.

   Note: When we tested the code on Monday at the Thode Makerspace the program worked (after we got rid of the break). However, when we tried again it doesn't work? And we get some crazy numbers. It seems to be some sort of memory issues or initialization issue.



```
Player::movePlayer (this=0x1217f60) at Player.cpp:70
70          objPos currHead;
(gdb) i local
currHead = {x = 352363204, y = -2, symbol = -8 'o'}
boardSizeX = 111
boardSizeY = -1
body = {x = 4199136, y = 6488012, symbol = 40 '('}
foodPos = {x = 6487752, y = 4199784, symbol = -100 'o'}
(gdb) s
objPos::objPos (this=0x62fec0) at objPos.cpp:5
5           x = 0;
(gdb) i local
No locals.
(gdb) s
6           y = 0;
(gdb) s
7           symbol = 0; //NULL
(gdb) s
8       }
(gdb) s
Player::movePlayer (this=0x1217f60) at Player.cpp:71
71          playerPosList->getHeadElement(currHead);
(gdb) i local
currHead = {x = 0, y = 0, symbol = 0 '\000'}
boardSizeX = 111
boardSizeY = -1
body = {x = 4199136, y = 6488012, symbol = 40 '('}
foodPos = {x = 6487752, y = 4199784, symbol = -100 'o'}
(gdb)
```

2. ***[4 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 have any memory leak when we run Dr. Memory on its exe file. This seems to be the case because in each class and the main file ("Project.cpp") there are the appropriate deallocation calls and "delete" 's taking place. The use of "~ClassName" is also very nice and allows their program to run without any memory leak.

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.*

   Kevin: I have enjoyed working with a partner and it has made it fun and yet accountable for the parts I do and getting them done on time. The way that the project was structured in that it is separated into iterations for each member (A and B) was very nicely planned out and thus not causing a lot of interference issues with our code working together. Overall, it has been very good, and the project went well and it was all really smooth thanks to the structure and teaching done throughout the course.

   Hemant: Overall, I agree with Kevin. But there were a lot of issues with git merges.