COMPENG 2SH4 Project – Peer Evaluation

Your Team Members	Liam Sullivan and Nihal Inel
Team Members Evaluated	Brayden Roberts and Dyia Bhal

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.

Looking at the Game Mechanics file, we observed that objPos and objPosArrayList header files were included at the top, as they are both used within gameMechanics. The team also declared the variables in private, and member functions in public scope. One thing we observed that could have been done differently, is declaring a double pointer game board character variable in the public scope – this could have been declared in the private instead. In objPosSArrayList header file, objPos.h was included and the rest of the file also looks acceptable. No notable issues were noticed. Finally in the player.h file, including the other header files, declaring the variables in private scope, and writing the member function prototypes in the public scope was done nicely. To conclude, from the inclusion of the header files at the top in each of the desired files, we can easily get an idea of how each class interact with each other to achieve a common goal. From the member function prototypes, we can tell the possible behaviours and attributes of each class

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.

Looking at the main program loop, we can get a thorough understand at how each object interacts with each other using their member functions. A few things we noticed about the code is there is unnecessary variable declaration and initialization in the global scope. The declaration of the variables in the global could have been reduced using a better OOD approach. Other than small excess variable usage, the main file looks clean and organized and easy to interpret.

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.

- Uses Modularity, which allows the opportunity to reuse code using classes and objects. For something like the project where code is often reused over again, object-oriented programming is very helpful
- Uses Encapsulation, allows the prevention of possibly messing up someone else's code while working in a team. It prevents it buy not effecting the pre-existing code by using public and private scopes

- Easier with more complex code because as the PPA's got lengthier and more difficult, the code became harder to manage. OOD Programming is easier to keep track of as a lot of the code is reused.

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 code inside the object classes (Player, GameMechanics and objPosArrayList) is very well documented and almost every single member function's behaviour is described with good comments. However, in the main Project.cpp file, there is minimal documentation; but enough to get a general understanding of where each functionality is done.

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.

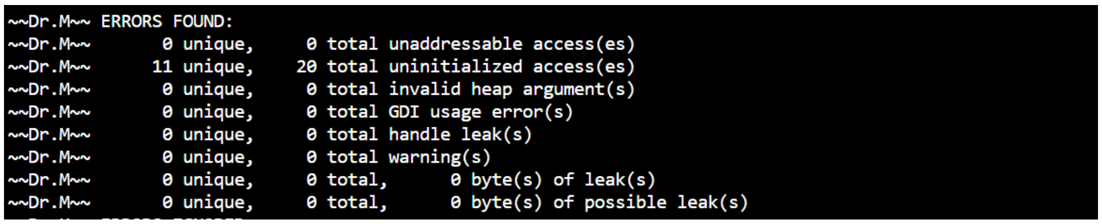
The code follows good indentation style and has enough white spaces where needed. The overall formatting was done carefully, contributing to the readability of the code. When I first looked at the code, I did not have a hard time interpreting the purpose the lines, which proves that the code is generally clean and understandable. While reviewing the Project.cpp file, particularly within the initialization section of the game loop, I noticed a minor instance where certain blocks of code lacked whitespace. This could have been improved by inserting an empty line between specific lines to enhance clarity and categorize the code for better comprehension.

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)

Upon compiling the code, the playing experience was very smooth and ran without any bugs. The program first welcomes the user with a well written introductory message, creating a clean entrance. It later offers the option to play with different board game sizes, which was a very well thought feature of the game that enhances its user-friendliness. The choice of the capital letter 'A' as the food object is excellent – it is easily noticeable during regeneration. Clear instructions displayed at the bottom of the game console help users understand the game mechanics, especially those unfamiliar with the snake game concept. The end-game features operate flawlessly; the game correctly exits when the snake eats itself, confirming the accurate implementation of the self-collision feature. Overall, it was a bug free and highly enjoyable playing experience, and we appreciate the team for their hard work!

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.



```
~~Dr.M~~ ERRORS FOUND:
~~Dr.M~~       0 unique,     0 total unaddressable access(es)
~~Dr.M~~      11 unique,    20 total uninitialized access(es)
~~Dr.M~~       0 unique,     0 total invalid heap argument(s)
~~Dr.M~~       0 unique,     0 total GDI usage error(s)
~~Dr.M~~       0 unique,     0 total handle leak(s)
~~Dr.M~~       0 unique,     0 total warning(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of leak(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of possible leak(s)
```

There are no memory leaks in the game as shown in the above drmemory report.

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.

This project was a very good learning experience as it really helped to expose us to the learning possibilities of working with a collaborator. Most projects in the past offer more creativity, which is why groups are effective, however this assignment is more goal driven. We know what we have to create using the skills we learned in the previous PPAs and have to perform it correctly and effectively. Some issues we faced during the project was the fact we had different software. One of us was using windows and the other one was using a MacBook, which posed a problem when pushing and pulling code into the repository. There were constantly things we needed to change for our software purposes which was annoying and caused fluidity issues. There is not much of a solution for this problem other than limiting the amount of times code is transferred. Other than that, the project experience was a great opportunity as we got to learn from our partners how they tackle problems and how other people achieve similar goals.