COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Matthew Rakic James Cameron

Team Members Evaluated Charanjit Rikhi Mathew Gobeil-Chianchai

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.

   When regarding the OOD code modularization of the peer code, the code modularization is generally well-defined. The program's code is well defined in the sense that header files were created for almost all scenarios where they needed to be made. One difference observed revolves around the use of a **Food.h** header file. It was provided as a choice whether or not this file was to be included, however not including the file as done in the peer code can cause slight confusion when looking for the **generatefood** and **getfoodpos** functions. This however does not affect the main program structure and functionality, it just causes slight difficulty when looking for the listed functions, (located in **GameMechs.h**). All other parts of the program were modularized very well, including all the correct functions in the correct parts of the code, allowing for easy interpretation of the possible behaviours for each object associated with the program.

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.

   The program follows an object-oriented design approach, utilizing classes such as GameMechs, myPlayer, and objPosArrayList. This makes the code modular, promotes encapsulation, and allows for better organization. The internal state of objects is encapsulated in the member functions of classes (GameMechs and Player), and well-defined interfaces control external access. This encapsulation prevents accidental interference and helps with complexity management. Dynamic memory is used to create instances of GameMechs and Player on the heap. The cleanup process (CleanUp function) correctly releases the allocated memory, preventing memory leaks.

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.
   Pros:
   - Program functions defined according to each object allow for simpler retrieval

- Defined header files for each object allow for simpler debugging as it is easier to determine where issues are arising
- Code is able to be reused for inheritance, reuse functions for different purposes

Cons:

- **OOD** design can take longer to design than a procedural design due to various files being created
- **OOD** may lead to increased file size or memory usage

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.

   All parts of the code were commented on very well, including the parts of the code which were directly implemented from PPA2/PPA3. This was a great addition, as, throughout the program, these parts which were completed using PPA2/3 were written differently from what was previously seen, allowing for easier understanding of the functionality of the code. Specifically for the GameMechs.cpp and Player.cpp files the commenting was great, explaining every detail allowing us to understand how most of the code worked very quickly.

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 in the main program file (**project.cpp**) follows good indentation and presents a well-organized look. When looking at the other main files, (**gamemechs.cpp and player.cpp**), the code is generally organized well with proper spacing and indentation, however, there is a slight unorganized aspect involving the commenting. It seems unnecessary to comment on every part of the code as many parts were predesigned by the professor which were quite self-explanatory and did not require commenting to be done. This just added to the length of each file, causing them to be double the size as they would be without the commenting. This however does not mean that comments on the added and more intricate features were not required, as these were done very well. The newline, (\n) was implemented well in the drawn screen, clearly printing the endgame message and score.

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 program designed by our peers was very well designed. Firstly, when starting up the game, the board was well designed and our peers selected good characters for the player, '@', and the food, 'o'. The printed message telling the user to collect the food is well written, however, it may be a good addition to include a message on how to exit the game, (found to be the ESC key). The functionality regarding the consumption of food and increase of player length has a very smooth look as the peer code is well designed for the **addHead and addTail** functions in **objPosArrayList.cpp.** One inefficiency in the code however is present in the **player.cpp** file. The code in **player.cpp** doesn't create functions **for checkSelfCollision, checkFoodConsumption, and increasePlayerLength.** These functions were not explicitly "required" in the project manual; however, these functions were shown in the class image for iteration 3, hinting at the inclusion of three functions to be created and not just all code written in the **movePlayer** function. This would just allow new users to better understand how these 3 pieces of the code interact with the player object. Aside from this, all other functions of the code were well-written and efficient, however, we found one error in the **generateFood** function. While playing the game before reading this function's code, it was observed that food was spawning underneath the snake's tail when a larger score was reached. When diving deeper into this function, we found an error in line 137 of generateFood, gameMechs.cpp, where they are attempting to set samePos to true. They used ==, which is incorrect. To set the value of the Boolean variable to true, only 1 = must be used, as it is not an equality. This error was found and fixed within seconds, and after testing the **generateFood** was running correctly. Also, the condition of the while loop could be simplified as the random generation accounts for the food being in the border and it doesn't need to be checked, instead a variable named count for example could be set to 0 and the while loop could be written as while (count ==0), and count could be set to 1 after a valid food position is located. In the future, they could test the game more to ensure this food generation error is not occurring.

```
samePos = false;
for(int i = 0;i<blockOff->getSize();i++)
{
    blockOff->getElement(tempBody, i);
    if(randomCoords.isPosEqual(&tempBody))
    {
        // Exiting the loop if the positions are the same
        samePos == true;
        continue;
    }
}
```

(It was extremely hard for us to get a screenshot of the snake tail being on top of the food, however, it happened repeatedly).

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.

   No, after testing for memory leaks there are 0 leaks. The code of our peers was properly designed in means that they accounted for the deletion of **myPlayer** and **myGM** memory allocation, using the C++ delete function.

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.

James Cameron:

Matthew and I have collaborated together in many classes on many projects and assignments, and we continued to collaborate extremely well and do a very good job at producing a well-developed program. I thoroughly enjoyed the project as it allowed me to apply everything I have been learning from lectures, and I believe that learning about OODs and how to use them will benefit me for my future as a programmer. The project design approach was also great, as we were able to split up iterations 1 and 2 to allow for optimal time efficiency for the simpler parts, however, we were still able to team up for the more intricate later iterations, giving us the chance to further develop our group programming skills. Matt did a great job at looking over the parts of the code I wrote, and we collaborated to determine issues from both our parts of the code.

Matthew Rakic:

During the project me and James collaborated very well and did not have many issues working together. The work was split evenly, and we mostly worked on the code together to save time pushing and pulling and having 2 perspectives while creating the code helped our efficiency a lot. James was an excellent partner, and his coding understanding helped our success a lot as he was able to fix many of the problems we encountered over the course of the project. Overall, I think we collaborated very well and produced an end result that we can both be proud of and know that we met the expectations of the project.