COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Sarujan Baheerathan and Salini Sivalingam

Team Members Evaluated Dennis Cao and Leo Calogero

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.

After observing the header files, we found that it was easy to interpret the possible behaviour of the objects involved in the program. Variable names in the parameter lists were easy to follow which allowed us to follow the other team's logic. Some positive features were that they kept their header files organized and added comments for readability. The team realized gameMechs did not require a destructor since nothing was being created on the heap in the constructors. The addition of checkFoodConsumption in Player.cpp seems to be very helpful in their design as it allows for an easy way to check if the snake consumed food in the movePlayer function.

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 code in the main program loop was easily interpretable because of their strong OOD implementation. Their code was well implemented in terms of encapsulation, making sure that private members stayed private, and changes to data are made only when needed. The code was also proficient in scalability, as sizes in the code were always implemented as variables. To implement a change, the code would only need to be modified at one point. This change will then reflect across the code, due to their efficient scalability. Another positive feature was the inclusion of a separate check food consumption function. This separate function in the player file allowed for a clean and concise move player function, where its only goal was to move the player. Not having to implement a food check routine here increased the codes readability, especially for those new to the 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.

PROS	CONS
• The C++ OOD allowed for function overloading which helped tremendously when creating constructors. • Another benefit with this is the ability to encapsulate information from the public interface. This proved to be useful with our data members and made sure that the values could not be changed easily. • C++ also makes managing memory leakage easier using destructors. C++ automatically deletes surface level instances, so that we only need to worry about deleting the sub-level heap instances.	• Implementing classes might be a concept that is hard to grasp and, in some cases, might be over complicating things • The addition of classes increases the size of our code which can generally make the program slower. This can result in testing times being longer making the development more inefficient.

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.

This team made sure that the code was thoroughly commented, offering explanations to actions that were not as straightforward or obvious. However, one concern with the commenting was that some of the comments could have included more of a detailed explanation. While it was easy for us to understand, we have already completed this project, and therefore have a general understanding of the layout and the functionality of the code. For someone with no context, comments with more description would be useful. An example of a place where this could be implemented is found on line 139 of the Project.cpp. Rather than just saying “checking for food”, adding something like “printing food symbol if food coordinate and current board coordinate match.” This addition would help those reading understand what exactly this if statement is checking for.

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.

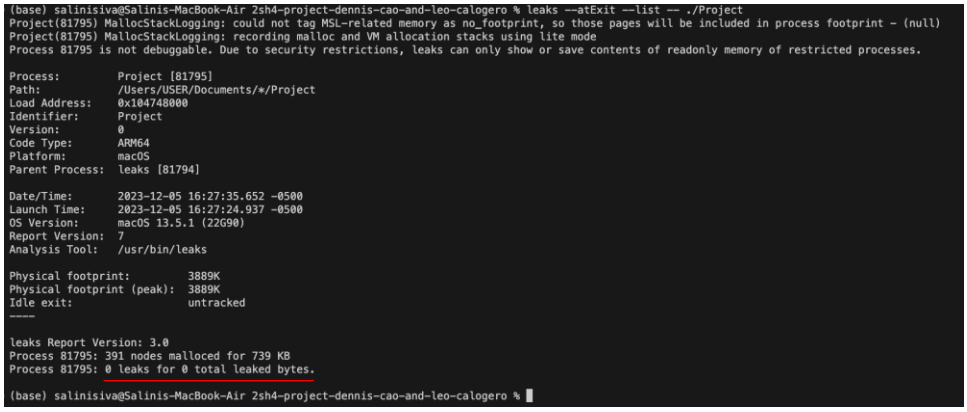
After observing the code, it is evident that the other team followed good indentation throughout their function implementations, control structures, and loop blocks. Visually the code looks neat and organized which the other team achieved through the addition of sensible white spaces and indentation. Print statements had newline characters which contributed to the overall readability and organization of the code, which provides the user with a smooth experience when running the game. An example of this can be found in the Project.cpp

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)

After running the other team's game multiple times, we found that it offered a smooth, bug free experience. Features such as the generation of food did not have any issues when randomly generating new food objects. Snake movement functions well and does not appear to move within border suggesting the wraparound feature is working correctly. Snake is growing at tail end when food is consumed followed by the generation of new food. Score also updates correctly after food is consumed without any calculation errors, it follows an increment of 1. User forced shut down key also functions correctly, however there is no message indicating that the user shutdown the game.

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.



As seen in the image above, their snake game did not cause 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.

Overall, the project ran very well. Having two separate goals for each team member was very useful as tasks were accomplished efficiently. It allowed each team member to produce something that was important for the game and allowed them to see progress in their code. Sharing our work to one another was not a difficult task as well.