COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members Euan Bohm, Jack McMillan

Team Members Evaluated Mustafa & Rameez

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.

   Yes, it is relatively easy to see how the Player interacts with the GameMechanics. The player handles its movement and collision, while the GameMechanics class handles score implementation and food generation. This is a pretty intuitive way of designing the program.

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:

   - Easier to work in parallel with partner as can focus on specific classes
   - Easier to read main program logic
   - Easier to test and slowly build up the program by focusing on one class at a time
   - Inheritance allows for code to be reused and reduces redundancy by allowing new classes to inherit functionality from things that already exists.

   Cons:

   - More files mean it is a bit harder to scan through for complete understanding
   - May be slightly harder to trace bugs if it involves multiple classes
   - Abstraction and other OOP functions may reduce processing time and efficency

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.

   Commenting was very well done throughout, as they explained what certain sections of code were doing. An example of this is in Player.cpp, lines 87-96. The comments are not

overwhelming, however they still provide enough context to understand what the code is doing. The only "shortcoming" might be that sometimes there are a few too many comments, rather than too little. For example, in GameMechs.cpp lines 6-12 there are comments on every line, some of which do not provide any additional context. The result is minimal additional information with a tougher to read section.

To fix this, we would suggest trying to encapsulate what the section is doing rather than explaining every line. For instance, in the above example they could instead say something along the lines of "Initialize all necessary variables". Because their variable names are well picked and explain their purpose relatively well to the reader, this combination would provide more than enough context to ensure understanding.

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.

While good indentation and whitespace are followed, new lines are not implemented where they could be, which leads to unclear code segments. An example of this is in Project.cpp, lines 73-98. As there are no new lines or separation this code segment is a bit difficult to read through. While this is just one specific example, this same observation can be made in the Player.cpp and GameMechs.cpp files.

The fix to this would be to try to separate different sections with newlines. While this is relatively subjective, we think any separation would most lead to easier readability.

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)

There are no buggy features within the code. It runs very smoothly and has no noticeable issues.

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.

There were no recorded memory leaks in the report generated by Xcode leaks.

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?

   We think it's a bit weird to have an extra parameter "pos" in order to access x and y. It seems unintuitive, and also requires the programmer to go through an extra variable in order to access the x and y values of the objPos.

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).

   Instead of using the composite design, we would instead replace it with two integers (x and y), which correspond to its position. This essentially cuts out the extra variable and would make the design a bit more intuitive.

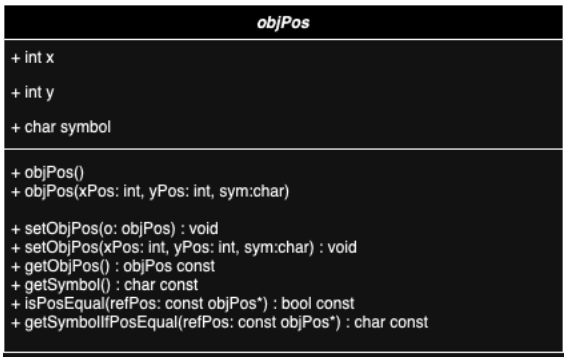

Figure 1: First UML Diagram

   Alternatively, if the position was required to be on the heap, these could also be made into pointers which held the address of the corresponding position, as outlined in the UML diagram below.

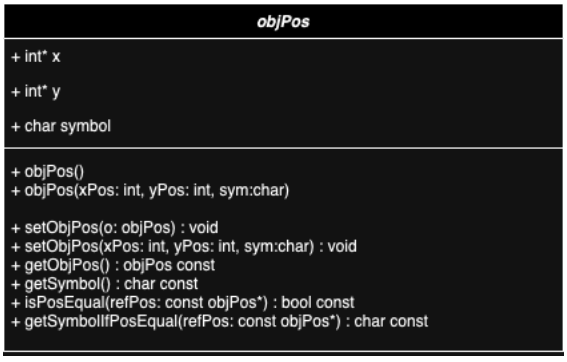

Figure 2: Second UML Diagram

We believe that this design would be easier to implement as well as more intuitive to understand.