COMPENG 2SH4 Project – Peer Evaluation [25 Marks]

Your Team Members Vani Maharaj Theepika Piramenthiratheesan

Team Members Evaluated khehrn2-sasits1

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.

   When reading the code, it is evident how the different classes are used within each other. The initialization of the class variables at the beginning of the code allows them to be used throughout the entire program without continuously calling them in each module, making the code less cluttered. The parameters used for the functions of different classes throughout the code included multiple classes needed, demonstrating the interconnection of the different classes to yield the snake game result.

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	Cons
C++ OOD Approach	Reusability – Can reuse code through inheritance Scalability – Can easily add on to and change design Abstraction – Hides the inner workings of the functions, which helps make the code easier to follow	More complex concepts - Requires a solid understating on concepts such as inheritance Requires more time to set up
C Procedural Design Approach	Simplicity – Easier to write code for Requires less time to set up	Harder to debug Harder to read Code repetition occurs (because there is no inheritance)

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.

   There were sufficient comments all throughout the code explaining the reasoning behind declaring certain variables, removing specific elements, and the reasoning behind the functions in the classes and the main project. Comments were made mainly in the Player class and the main Project file, the GameMechs did not have as many comments, however, they were not needed as majority of the functions were single line setters and getters. The comments that are present would help an outside programmer understand the reasoning behind the choice that this team made.

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.

   The code does follow good indentation rules and have sensible white space. This can be seen as their code looks very neat. The code also has newlines included in the print statement for better user 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)

   Yes, there are not observable bug while playing the game and it works just as expected. There is an exit statement when you crash into yourself and one when you exit without crashing. The only change I would make is to make the board bigger because the snake automatically kills itself when it reaches the length of the board and wraps around. Since the board is so small, the snake game ends after you score 8 points.

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.

   No, the code has no memory leak.

Project Reflection

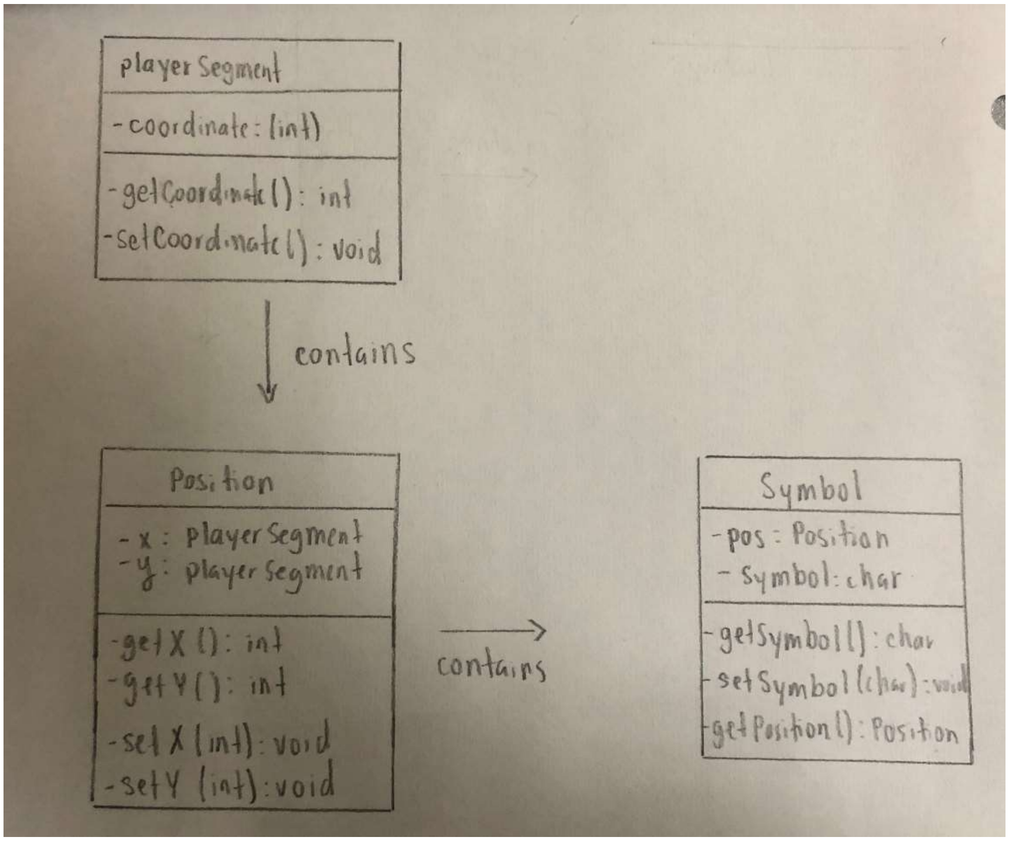
Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team's implementation in addition to your own, reflect on the following questions:

1. **[3 marks]** Do you think the compound object design of objPos class is sensible? Why or why not?

   I think the compound object design of objPos class is sensible. I think it is sensible because it shows the user how the x coordinate, y coordinate and symbol are all connected to each other. Another reason why I think this design makes sense is because it helps reduce code repetition. For example, objPos was used in defining the induvial snake body segments, and also the food items. For these reasons I believe that using objPos was sensible.

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. <u>You are expected to facilitate the discussion with UML diagram(s).</u>

   An alternative objPos class design is having a playerSegment class that has a variable of type int called coordinate. This class contains another class called Position (which has the x and y coordinates). Then the Position class contains another class called Symbol (which has the player symbol and position).


player Segment
- coordinate: (int)
- getCoordinate(): int
- setCoordinate(): void
contains
Position
- x : player Segment
- y : player Segment
- getX(): int
- getY(): int
- set X (int): void
- setY (int): void
contains
Symbol
- pos : Position
- Symbol: char
- getSymbol(): char
- set Symbol (char): void
- get Position(): Position