COMPENG 2SH4 Project – Peer Evaluation **[30 Marks]**

Your Team Members Benji Switzman (switzmab), Antoine Grenier (grenia4)

Team Members Evaluated shawb10, pyer2

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 **30 marks**. Do not exceed 2 paragraphs per question.

<u>Peer Code Review: OOD Quality</u>

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.

- **The main logic in Project.cpp is quite easy to follow and understand how things interact. The program is clear and concise. Variables are clear on what they represent and run logic is extremely simple. The only negative feature was that the draw screen was overcomplicated. There were a few times where object oriented program could have been further taken advantage of in order to simplify the code**

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

- C++ OOD allows for better organization of the main program loop as it categorizes and organizes functions and 'commands' into different classes that can be called with one line.
- Private sections of classes make it much easier to ensure that variables and functions are not modified are used unless explicitly called upon.
-

Cons

- Design in C is easy to setup and utilize all features rather than having to go through setter and getters to access and modify private variables
- Debugging is much more difficult as there are multiple files to go through and many inherit or use features from different classes.
-

<u>Peer Code Review: Code Quality</u>

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.

- The code has enough comments to understand the functionality efficiently with comments used only when confusion may arise
- There was sufficient commenting so not shortcomings. The only issue found was some commented out code that could have been removed.

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 a solid job with indentation, and ensuring white space is there as well as newline formatting for the draw screen. The code is easy to read through and the draw screen is organized.**

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)

- **The snake game offers a smooth, bug-free experience with food respawning immediately after consumption and the snake length growing immediately.**

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 is no memory leak in the snake game. Everything allocated on the heap is deallocated when no longer needed.

Project Reflection

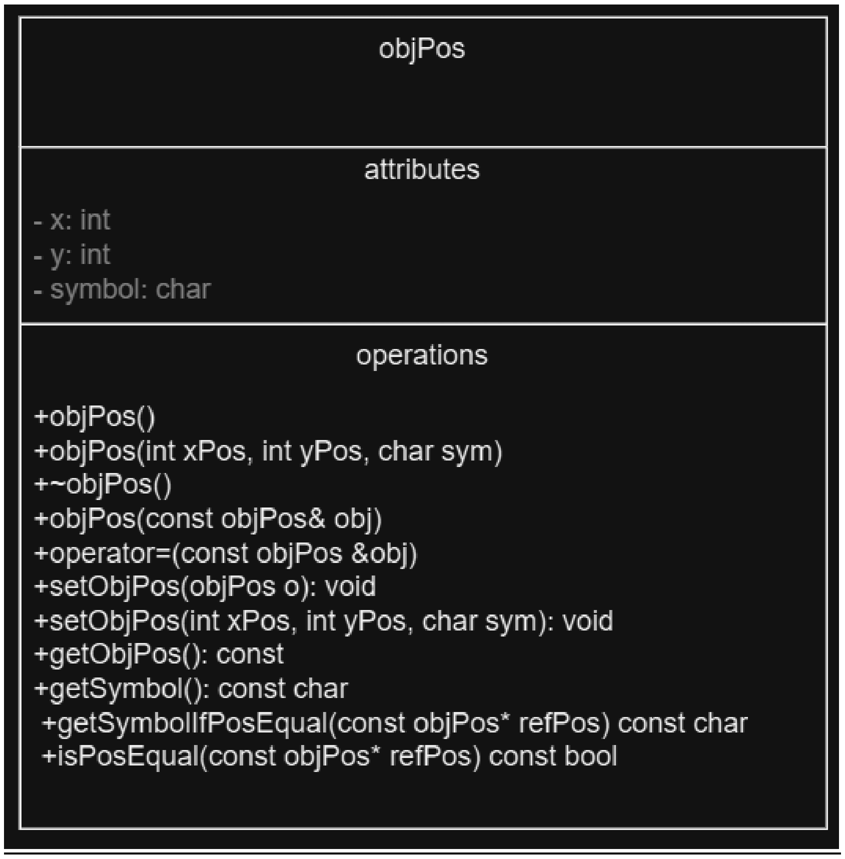
Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team's implementation in addition to yours, reflect on the following questions:

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

- The use of the objPos class is sensible as it allowed for the class to be used for multiple purposes. Both food and player objects use objPos so that the position can be tracked and updated the entire time. The Pos struct is also useful for encapsulation as it keeps the position values hidden from other classes so that it there is better control for accessing and modifying it.

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.</u>

- An alternative objPos class design could have been to eliminate the Pos struct and rather build it directly into the class via the constructor. Eliminating the Pos struct would make most of the programming experience the same however, it would eliminate the need to call the x or y value of the pos and instead just call the x and y variables directly using the getObjPos() method.



objPos

attributes

- x: int
- y: int
- symbol: char

operations

+objPos()
+objPos(int xPos, int yPos, char sym)
+~objPos()
+objPos(const objPos& obj)
+operator=(const objPos &obj)
+setObjPos(objPos o): void
+setObjPos(int xPos, int yPos, char sym): void
+getObjPos(): const
+getSymbol(): const char
+getSymbolIfPosEqual(const objPos* refPos) const char
+isPosEqual(const objPos* refPos) const bool