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. The project file has similar logic to ours, as it iterates through the game board, and prints a corresponding symbol depending on whether the position is clear or not. The bonus feature is implemented and works. As a nitpick, the number of MacUILib_printf() statements that print the game instructions can be reduced using different print logic.

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.

   - **C++ OOD PRO's**
     - **Easier to interpret**
     - **Modularity allows for easier debugging**
     - **Modularity allows for easier collaboration between two partners (can separate classes/methods)**
   - **C++ OOD CON's**
     - **Slower performance (must cross-reference, using more memory)**
     - **Easier to learn**
     - **More code required**

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.

   For the most part, this code is well commented. More self-explanatory classes and methods are tastefully commented, with a line at the start of each method explaining it. However, for more complicated game logic methods such as DrawScreen could benefit from a few more comments explaining computational lines,
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.

   In short, yes. The indentation is well-done (good use of alt+shift+f), and follows the expected formatting. The whitespaces and newlines are tastefully implemented, and improve the readability. Blocks of code that perform similar operations are grouped together, and lines that remain in the same class but perform different steps of computation are separated (such as the conditions in the Player and Project files).

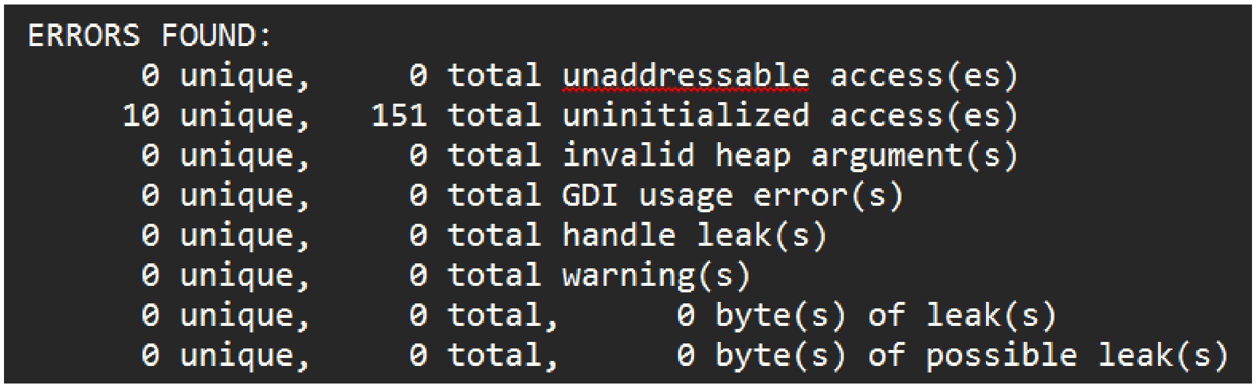
Peer Code Review: Quick Functional Evaluation

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 mainly a smooth, bug-free playing experience. However, sometimes, the game inputs start to stutter for a few seconds, then return back to normal. Upon further inspection of their code, all pointers seem to be deleted, and any memory issues do not seem evident directly through the code. This is likely a side effect of running a game like this within the terminal and not having a dedicated graphical user interface.

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, there are 0 bytes of memory leak.



```
ERRORS FOUND:
      0 unique,     0 total unaddressable access(es)
     10 unique,   151 total uninitialized access(es)
      0 unique,     0 total invalid heap argument(s)
      0 unique,     0 total GDI usage error(s)
      0 unique,     0 total handle leak(s)
      0 unique,     0 total warning(s)
      0 unique,     0 total,      0 byte(s) of leak(s)
      0 unique,     0 total,      0 byte(s) of possible leak(s)
```

This is the memory leakage report.

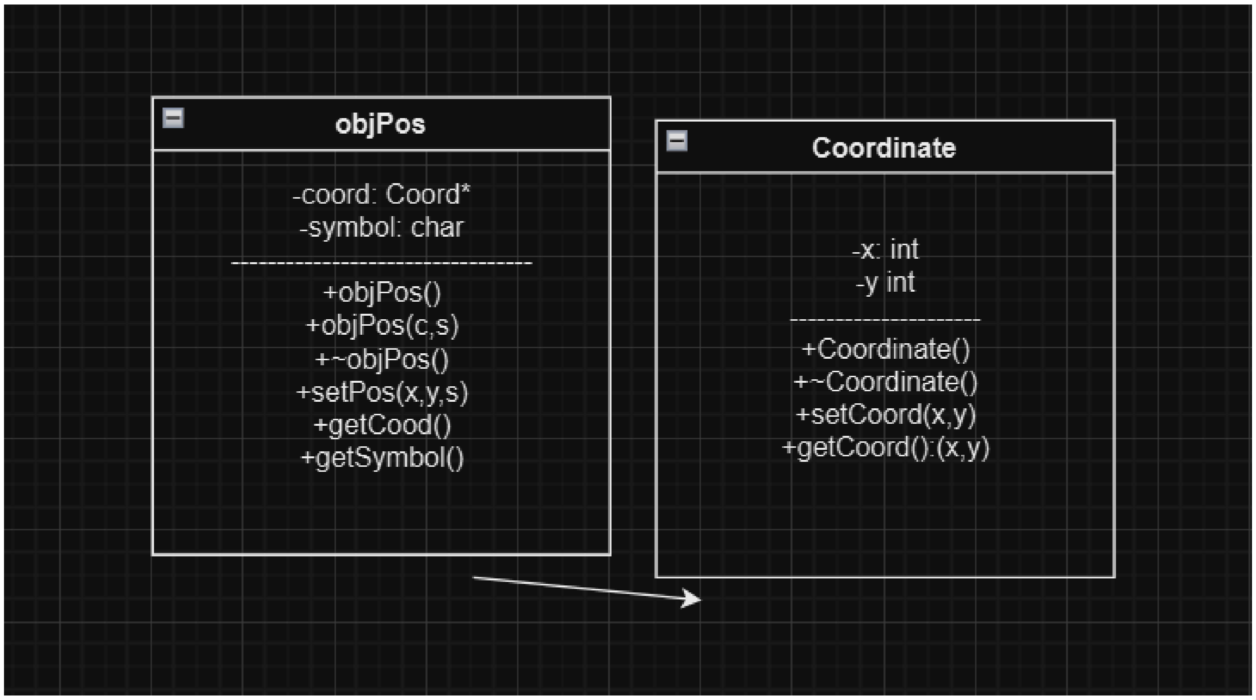
Project Reflection

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?
   The objPos class is sensible because it uses dynamic memory allocation which allows flexibility of how objects are used throughout the program. Additionally, any added attributes are much easier to handle if it's in a class setting and the destructor copy constructor and assignment operator allow clean and efficient memory allocation on the heap.

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>

A more counterintuitive design is if there was a separate class for coordinates(x,y) tied to the objPos class that handles all the other functions. It would have a getCoordinate and setCoordinate function under it while the rest of the functions including these two would be under the objPos class. The UML diagram below describes it.


objPos
-coord: Coord*
-symbol: char
+objPos()
+objPos(c,s)
+~objPos()
+setPos(x,y,s)
+getCood()
+getSymbol()
Coordinate
-x: int
-y int
+Coordinate()
+~Coordinate()
+setCoord(x,y)
+getCoord():(x,y)