COMPENG 2SH4 Project – Peer Evaluation [30 Marks]

Your Team Members Kristina Mouzakitis - mouzakik, Abigail Rivera - rivera8

Team Members Evaluated shinj16, roht

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.

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.

By examining the main program loop, it can easily be interpreted how the objects interact with each other. The Initialize function contains the creation of the heap members myGm, myFood, and myPlayer which interact through the Player class using myGM and myFood, and the Food class using myGM for their overlapping uses within each. As well, the food generation through the Food class has a parameter of the player position from the Player class, which easily ensures that the food is not generated where the player symbols are located. However, these are the only clear interactions within the main program loop itself. The DrawScreen function may be expected to contain more object use, but instead the game board is generated from within the GameMechs class. This is negative feature due to its unclear logic within the main program loop and overuse of connecting the classes to each other. By being intertwined with each other within the classes themselves, it prevents each class from individual use elsewhere, such as in other projects

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.

OOD Approach

Pros:

- Able to separate different components such as player movement or food generation into their own code sections
- Easier to work in parallel with another programmer through classes
- Classes offer reusability since they hold an entire subject of code separately
- OOD makes it far easier to change/update code

Cons:

- Object-oriented design is more complex than simple procedural logic due to object structures and overlapping
- Slower due to dynamic memory allocation
- Greater risk of memory leak from more heap allocation

C Procedural Design

Pros:

- Procedural code is more straightforward and easier to follow its logic
- Easier to debug without objects and classes to manage

Cons

- Contains repeated code from similar logic occurring throughout the project
- Harder to manage interacting components or larger projects

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.

The code offers a sufficient number of comments. Only the file objPosArrayList has very little comments even if the code can be straightforward. The file objPosArrayList.cpp only has two lines of comments that have been given from the initial project files. However, the number of comments in GameMechs and Player allowed us to fully understand the functionality of the code and program flow. Minor optimization details we observed is the implementation of displaying/drawing the board screen. Having the print statement in the Project.cpp file rather than in GameMechs.cpp would help the entire program avoid repeated for loops and if loops for the drawing statement. Along with having the print statement in the Project.cpp would help for player friendliness to see the board screen once the game ended. Since the drawing feature is in GameMechs, once the game has ended the entire board screen is cleared and you cannot see the final score or what the game looked like at the end. This does not affect the functionality of the code but rather the optimization and user-friendliness. Besides the drawing board feature, the program is fully functional with bonus features.

The only shortcoming of the project is the user-friendliness. There is no description on the print screen of the forced exit key or a special score legend. The print screen only has the board, and score displayed. This makes it harder for the player to know what special food items may do or how to forcefully exit the game if they do not want to play anymore. Although, not having a user-friendly print screen does not affect the functionality of the program. To improve this feature, we would add a print statement at the end of the DrawScreen function in Project.cpp. Having a line such as “MacUILib_printf("\nTo exit press ESC\n* = 1 Point $ = 10 Points”);” would improve the user friendliness of the program.

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 follows proper indentation, sensible white space and deploys newline formatting after each section of code. The code within the program never looks crowded and is always easy to read. There are no shortcomings observed or any recommendations to be made.

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 is a bug-free experience. There seems to be no issues with the food printing, character moving, game board and exit/end game features. We tested the game multiple times by testing edge cases such as printing and exiting the game to see any draw screen errors, testing how the snake moves (see if the snake is moving right, it will not change into left), and having the snake collide into itself in many different spots within the game board. After testing all these features, the program always ran as expected and never showed any bugs. The game seems to be fully functional, even after inspecting the code, the program seems to follow the expected code implementation of the project manual. There are no potential debugging approaches or coding fixes to implement or recommend, the program seems to not have any bugs and runs flawlessly.

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 leakage in their project. After running DrMemory on the project code the report said:

```
~~Dr.M~~
~~Dr.M~~ ERRORS FOUND:
~~Dr.M~~       0 unique,     0 total unaddressable access(es)
~~Dr.M~~       6 unique,    62 total uninitialized access(es)
~~Dr.M~~       0 unique,     0 total invalid heap argument(s)
~~Dr.M~~       0 unique,     0 total GDI usage error(s)
~~Dr.M~~       0 unique,     0 total handle leak(s)
~~Dr.M~~       0 unique,     0 total warning(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of leak(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of possible leak(s)
~~Dr.M~~ ERRORS IGNORED:
~~Dr.M~~      19 potential error(s) (suspected false positives)
~~Dr.M~~          (details: C:\Users\abbyr\Documents\Drmemory\DrMemory-Windows-2.6.0\drmemory\logs\DrMemory-Project.exe.13332.000\potential_errors.txt)
~~Dr.M~~      25 unique,    25 total,   7188 byte(s) of still-reachable allocation(s)
~~Dr.M~~          (re-run with "-show_reachable" for details)
~~Dr.M~~ Details: C:\Users\abbyr\Documents\Drmemory\DrMemory-Windows-2.6.0\drmemory\logs\DrMemory-Project.exe.13332.000\results.txt
PS C:\Users\abbyr\Documents\COE2SH4\PeerEval\course-project-thevenin>
```

It also seen that any new/features allocated on heap memory has been deleted in the destructors in GameMechs, Player and Food. Along with features allocated on heap memory being deleted at the end of the Project file in CleanUp.

Project Reflection

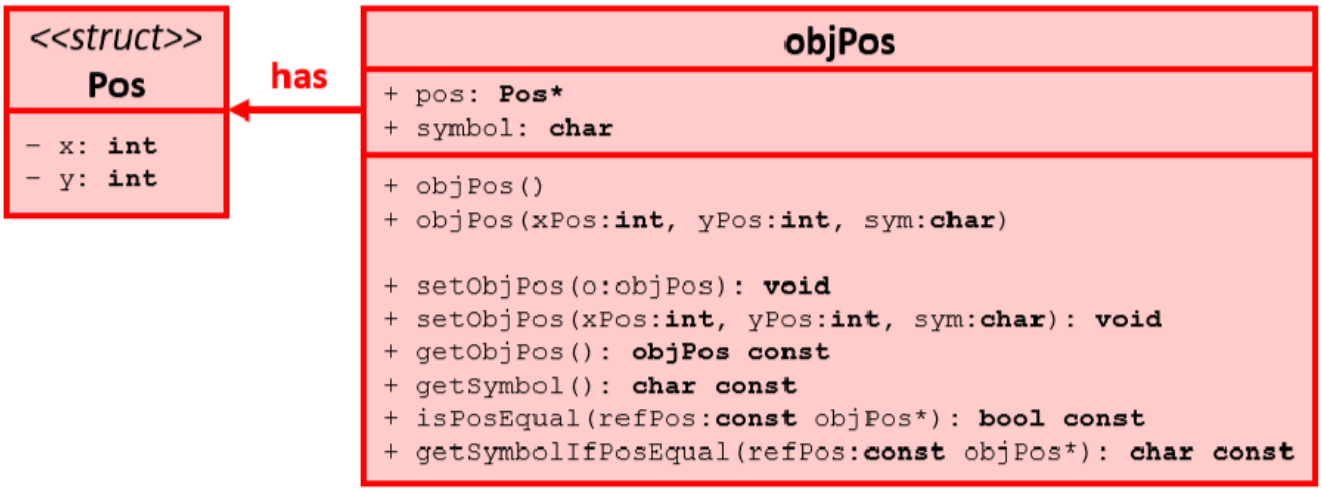
Recall the unusual objPos class design with the additional Pos struct. After reviewing the other team's implementation in addition to yours, reflect on the following questions:

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

The compound object design of the objPos class is not sensible, as it is more complex than it needs to be. The pos pointer makes it so that every time the struct members x and y are accessed, they need to be dereferenced. As well, since the struct is dynamically allocated, it adds further risk of memory leakage. In general, being required to access the struct is an unnecessary task in the objPos class.

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.

A way that the object design can be improved is by simply turning x and y into integers exactly how the symbol is just a character in the class. This would make the class less complex and easier to understand by having more straightforward constructors.



The UML diagram above shows the original objPos with its unnecessary struct and pointer, while it simply contains a symbol designated as a character. Comparing this to the new UML diagram below, it is clear that the direct integers will simplify the object design. The x and y coordinates are easily accessible and now operate in the same manner as the player symbol.