

COMPENG 2SH4 Project – Peer Evaluation

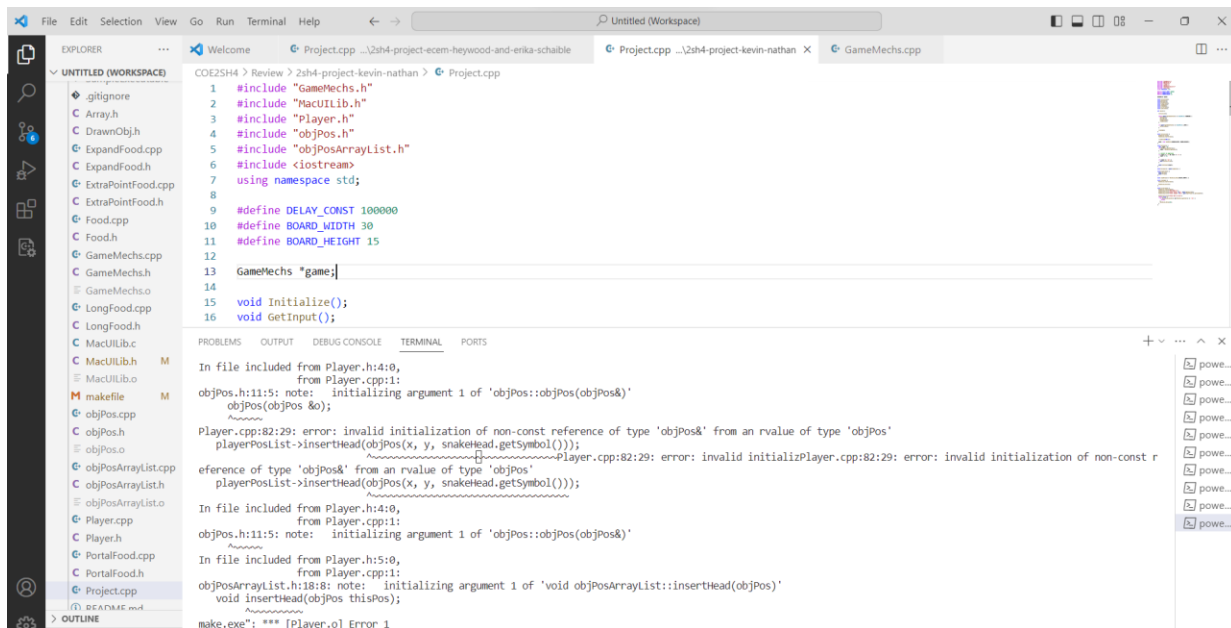
Your Team Members: Ecem Heywood and Erika Schaible

Team Members Evaluated: Kevin and Nathan

Provide your genuine and engineeringly verifiable feedback. Ungrounded claims will lead to deductions.

The program could not be fully compiled and run because there was no project executable created when 'make' was called. This was confirmed with Dr Scott Chen.

On the night this was due, we received an email from Kevin and Nathan. Apparently, this happened since they worked on Mac and it does not compile on Windows (even with the changes we made to the MacUI library and other necessary files to run on Windows), and they mentioned they talked to Dr. Chen. As this was given with extremely late notice, without confirmation from the professor other than their email (no professor was CC'd either), we did not revise our report.



```

1 #include "GameMechs.h"
2 #include "MacUI.h"
3 #include "Player.h"
4 #include "objPos.h"
5 #include "objPosArrayList.h"
6 #include <iostream>
7 using namespace std;
8
9 #define DELAY_CONST 100000
10 #define BOARD_WIDTH 30
11 #define BOARD_HEIGHT 15
12
13 GameMechs *game;
14
15 void Initialize();
16 void GetInput();
  
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

In file included from Player.h:4:0,
from Player.cpp:1:
objPos.h:11:5: note: initializing argument 1 of 'objPos:objPos(objPos&)'
objPos(objPos &o);

Player.cpp:82:29: error: invalid initialization of non-const reference of type 'objPos&' from an rvalue of type 'objPos'
playerPosList->insertHead(objPos(x, y, snakeHead.getSymbol()));

PlayerPosList->insertHead(objPos(x, y, snakeHead.getSymbol()));
Player.cpp:82:29: error: invalid initialization of non-const r
ference of type 'objPos&' from an rvalue of type 'objPos'

In file included from Player.h:4:0,
from Player.cpp:1:
objPos.h:11:5: note: initializing argument 1 of 'objPos:objPos(objPos&)'

In file included from Player.h:5:0,
from Player.cpp:1:
objPosArrayList.h:18:8: note: initializing argument 1 of 'void objPosArrayList::insertHead(objPos)'
void insertHead(objPos thisPos);

make.exe": *** [Player.o] Error 1

Part I: OOD Quality

1. **[6 marks]** OOD is about sensible code modularization. Looking at the header files of each object, can you easily interpret the possible behaviours of the objects involved in the program, and how they would interact with each other in the program? Comment on what you have observed, both positive and negative features.

Looking at the code by Kevin and Nathan I see that they were organized and had separate header files for each object. This includes the given ones: objPos, objPosArrayList, Player, and GameMechs but they also created their own such as ExtraPointFood, ExpandFood, LongFood, etc. By using a clear naming convention, it is easy to read and interpret what each object is and its intended behaviours. As an example, looking at the file objPosArrayList we can clearly tell that this object's behavior is more focused on

the movement of the array which in this game is the snake (increasing and decreasing size and position). Looking at the function calls we see clear naming conventions that state exactly what the functions will do: `insetHead`, `removeTail`, `getElement`, etc. While also providing key information on the type of the function's input and return value (if applicable). Looking at the `ExtraPointFood` header file the name describes that it might deal with the score allotment for different foods and the functions inside also point to that intended purpose. It would be great to have more comments for added clarity on what is happening. Also, while there are very clear headers for multiple objects it is quite overwhelming to see everything in a separate file. This could also cause errors if they are not all correctly linked.

2. **[6 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.

Looking at the main program loop there are calls to the created object classes, but they are not fully implemented. There are calls to implement a retrieval of the input as well as calls to objects to implement draw screen. The code in the main project loop is succinct and minimal which shows a heavy reliance on the object's design and interaction which makes the project file very clean. While it is very clean it is a bit confusing exactly what implementation is happening where and to what extent, especially since there is very little commenting in the main project file.

3. **[5 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.

Some pros and cons of the C++ OOD approach is that

Pros of OOD

- There are less global variables as we are now just referencing and calling classes (cleaner main file)
- Adding new data to the function is easy
- Code reusability is easier
- Good for more complicated programs
- You can have private, protected and public data which safeguards your variables and functions from outside interaction

Cons of OOD

- Linking of the files of the objects can be messy if there are many objects
- Can be confusing to know how to access/which files you can access from other files
- If the program is simple it might seem like more work than necessary

Pros of Procedural

- Better for small to medium size programs that are not as complex
- Linear flow which can make it easier to read in some cases
- Can sometimes be more efficient as it requires less overhead than OOD (small programs)
- Easier to debug

Cons of Procedural

- Adding new data and functions isn't as easy
- There can be more repeats in code if not designed carefully (don't really have code reusability)
- Not as secure as there is no private or public key terms to define variables and functions

Part II: Code Quality

1. **[5 marks]** Does the code offer sufficient comments, or deploy 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 comments in necessary and helpful areas, giving insight into what is happening for important parts of the code. The code is written with a mostly self-documenting style, with clear and understandable variable names that indicated functionality. Some comments could have been added to the ExpandFood.h and ExtraPointFood.h files for more clarity. It would be helpful to have more comments to explain what is going on, especially in the main program.

2. **[4 marks]** Does the code follow good indentation, add sensible white spaces, and deploy newline formatting for better readability? If any shortcoming is observed, discuss how you would improve it.

The newline formatting could be improved on. For single statement functions, new lines are not created, and the function is written on a single line. While it is only one statement, it makes it slightly more confusing to read and less consistent with the rest of the code formatting. As well, some of the if else if statements are written as

```
If(){  
  
} else if (){  
  
} else if() ....
```

which makes it slightly harder to see the individual statements as opposed to having each if, else if, or else statement starting on their own line. For improvement, more space could be added between statements and functional blocks of code, just to make it easier to read and more self-documenting.

Part III: Quick Functional Evaluation

0. **[8 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 a technical user feedback)

```

6 Player::Player(int x, int y)
7   myDir = STOP;
8   playerPosList = new objPos
9   extendAmt = 2;
10
11   playerPosList->insertHead(objPos(x, y, '@'));
12 }

```

class "objPos" has no suitable copy constructor C/C++(334)

```

objPos::objPos(int xPos, int yPos, char sym)
+2 overloads
View Problem (Alt+F8) Quick Fix... (Ctrl+.)

```

```

76   x = (x - boardSize + (boardSize - 2 * boardSize))
77   | % (boardSizeX - 2 * boardSize)
78   y = (y - boardSize + (boardSize - 2 * boardSize))
79   | % (boardSizeY - 2 * boardSize)
80
81   // insert new element at head
82   playerPosList->insertHead(objPos(x, y, snakeHead.getSymbol()));
83 }
84

```

class "objPos" has no suitable copy constructor C/C++(334)

```

objPos::objPos(int xPos, int yPos, char sym)
+2 overloads
View Problem (Alt+F8) Quick Fix... (Ctrl+.)

```

Unfortunately, we were not able to compile the code and run it. No executable was created when “make” was called. Not being able to comment on the functionality, simply looking at the code indicates that there are a lot of proper implementations and if I could compile it without errors, I am convinced a lot of features would work perfectly. One error I discovered while looking at the Player.cpp file was with the call of objPos. The errors have been screenshotted above. I believe this is due to an improper call of several of the functions and classes. Something like this could be implemented instead for the second error (and a similar format for the first error):

objPos currentHead; //creates a space to store the current head element

playerPosList->getHeadElement(currentHead); //retrieves the current head element from playerPosList and saves it into currentHead

playerPosList->insertHead(currentHead); //inserts a new current head

This or possibly another solution might be able to solve the issue, but again I cannot be sure since I cannot run the actual program to check.

1. **[6 marks]** Does the Snake Game cause memory leak? If yes, provide a digest of the memory profiling report and identify the possible root cause of the memory leakage.

The submitted code cannot be compiled fully therefore the program cannot be checked/run. This means that Dr Memory cannot be run to check if there are any memory leaks. In files objPosArrayList.cpp and its header file, Player.cpp and its header file, GameMechs.cpp and its header file, and Array.h all seem to have some written delete function and then its caller but again because the code cannot be fully compiled and therefore run it is not for certain that there is no memory leakage. In Project.cpp, game is initialized on the heap, but is never deleted in the cleanup routine, so just from this I might assume that there is some slight memory leakage.

Part IV: Your Own Collaboration Experience (Ungraded)

1. Tell us about your experience in your first collaborated software development through this project – what was working and what wasn't. If you are a one-person team, tell us what you think may work better if you had a second collaborator working with you.

This project was the first time in this course we worked collaboratively on a piece of code. While there were some challenges, like getting used to the git pulling and pushing process on GitHub, working in partnership meant we could work in parallel and in tandem on certain aspects of the code. By working in parallel it meant larger portions of the code could be finished at the same time. When working in tandem with other sections we could bring the experience we gained from our own parts to solve problems in the combining and building of the code. One of the best parts of working with a partner was being able to share problems we were having, and brainstorm solutions together with two different perspectives. Many times, it was helpful to have someone who understood the code but might not have been actively working on that specific part, look at the problem with “fresh eyes” and identify small mistakes that got overlooked. Overall, it was a great experience.