# COMPENG 2SH4 Project – Peer Evaluation

Your Team Members __Yuchen Tao__ __Bill Zhou____

Team Members Evaluated _Serena_Santos__ __Kasra Noyan__

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

## 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.

The objects in the header files were named appropriately and were easy interpret the possible behaviours of the objects involved in the program. They also included "#define, #ifndef, and #endif" statements in each header file, so that the related header files aren't copied more than once, which avoid the completion error. To sum up, all of the header file looks nice and they list all the functions methodically.

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.

Overall, the logic in the main program loop was intuitive. I clearly see that the Initialize function sets up the game by initializing objects for game mechanics, player, and food. It also generates the initial piece of food. However, the Initialization for board size could be replaced with the default constructor for better readability and maintainability. The GetInput function simply gets the input by calling the GameMechs class. The RunLogic function is concise and easy to read, with clear function calls indicating the steps involved in updating the game logic. The sequence of function calls (updatePlayerDir and movePlayer) within RunLogic follows a logical order, making it easy to understand the flow of the game logic. The DrawScreen function is well-structured and provides a clear representation of the game state on the console. It includes information such as the game board, snake positions, food positions, scores, and user instructions. The LoopDelay function clearly introduces a delay in the program loop. The CleanUp function properly handles memory cleanup by deleting the dynamically allocated objects (myGM, myFood, myPlayer). It also prints specific loss messages. In addition, comments explain the purpose of specific lines or sections of code, making it easy to understand. In summary, the logic of the main program loop is generally well-structured and follows a typical game loop pattern.

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.

**Pros:**

- Concepts like players, game mechanics, and food are modeled as objects, enhancing abstraction.
- The code is more readable and self-explanatory due to the use of classes and methods.
- Object-oriented design encourages reusable components, potentially leading to more maintainable code.

**Cons:**

- While C++ provides more advanced memory management features like constructors and destructors, this can also introduce challenges such as potential memory leaks and increased risk of undefined behavior.
- The procedural code in C makes it easier to understand and keep track of what happening, while C++ needs to check classes while understanding the code.
- Compilation times for C++ programs can be longer than those for C programs. This can impact development speed and iteration times.

## Part II: Code Quality

1. **[5 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.

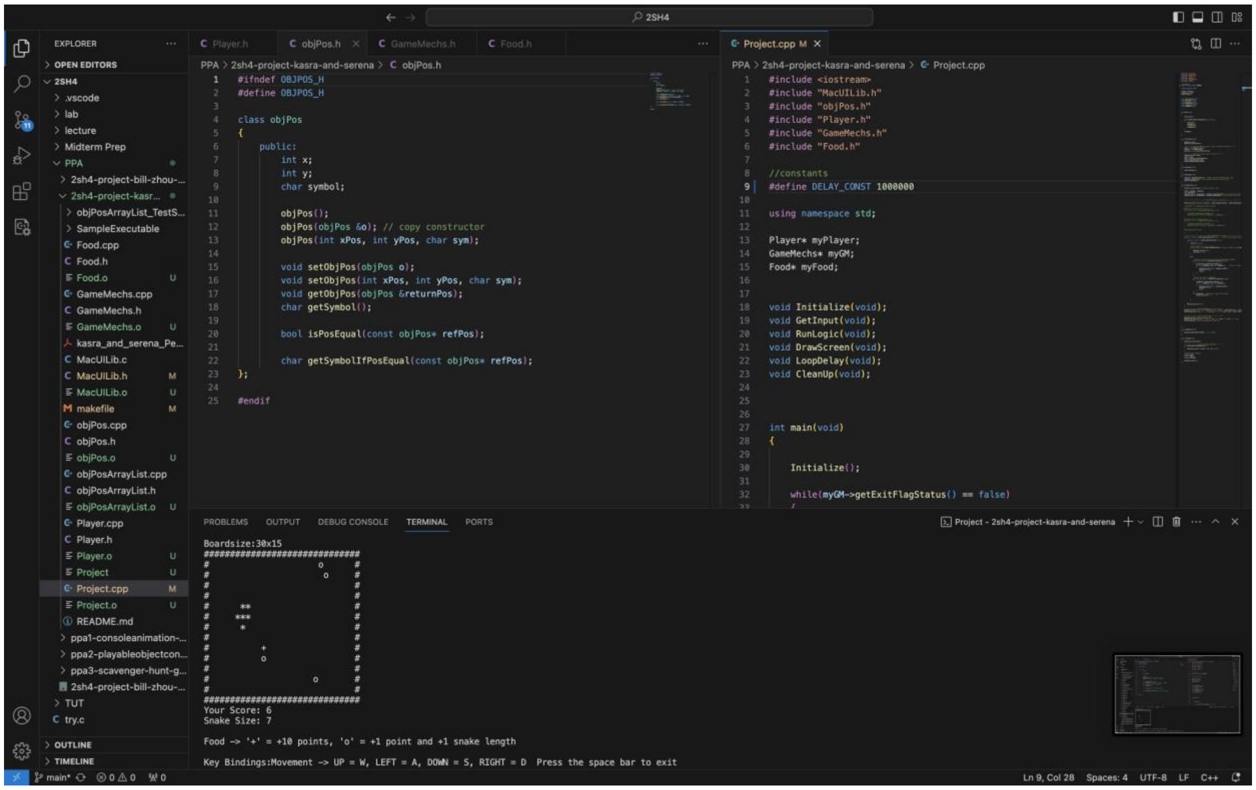
The provided code includes sufficient comments, which provide a basic level of understanding for certain parts of the code. However, there are areas where additional comments could enhance clarity. The global pointers (**myPlayer**, **myGM**, **myFood**) lack comments explaining their purpose and usage. Adding comments could provide insights into why these pointers are global and how they are used throughout the code.

2. **[4 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.

Yes, they move all the small pieces code to the class such as food.cpp, player.cpp and so on. Which make the main function is readable and easy to understand. And also, they write all the loops and functions with preparate indentations. Overall, they did great at this part.

## Part III: Quick Functional Evaluation

1. **[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)


2SH4
Player.h
objPos.h
GameMechs.h
Food.h
Project.cpp M
PPA > 2sh4-project-kasra-and-serena > objPos.h
PPA > 2sh4-project-kasra-and-serena > Project.cpp
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Project - 2sh4-project-kasra-and-serena
Ln 9, Col 28  Spaces: 4  UTF-8  LF  C++

```cpp
#ifndef OBJPOS_H
#define OBJPOS_H

class objPos
{
    public:
        int x;
        int y;
        char symbol;

        objPos();
        objPos(objPos &o); // copy constructor
        objPos(int xPos, int yPos, char sym);

        void setObjPos(objPos o);
        void setObjPos(int xPos, int yPos, char sym);
        void getObjPos(objPos &returnPos);
        char getSymbol();

        bool isPosEqual(const objPos* refPos);

        char getSymbolIfPosEqual(const objPos* refPos);
};

#endif
```

```cpp
#include <iostream>
#include "MacUILib.h"
#include "objPos.h"
#include "Player.h"
#include "GameMechs.h"
#include "Food.h"

//constants
#define DELAY_CONST 1000000

using namespace std;

Player* myPlayer;
GameMechs* myGM;
Food* myFood;


void Initialize(void);
void GetInput(void);
void RunLogic(void);
void DrawScreen(void);
void LoopDelay(void);
void CleanUp(void);



int main(void)
{

    Initialize();

    while(myGM->getExitFlagStatus() == false)
```

```
Boardsize:30x15
##############################
#                     o      #
#                      o     #
#                            #
#                            #
#      **                    #
#     ***                    #
#      *                     #
#                            #
#          +                 #
#          o                 #
#                            #
#                    o       #
#                            #
##############################
Your Score: 6
Snake Size: 7

Food -> '+' = +10 points, 'o' = +1 point and +1 snake length

Key Bindings:Movement -> UP = W, LEFT = A, DOWN = S, RIGHT = D  Press the space bar to exit
```

- For function (check_self_collision):

```cpp
void RunLogic(void)
{
   myPlayer->updatePlayerDir(); //change direction depending on input
   myPlayer->movePlayer(); //move player based on current state
   myGM->clearInput();
}
```

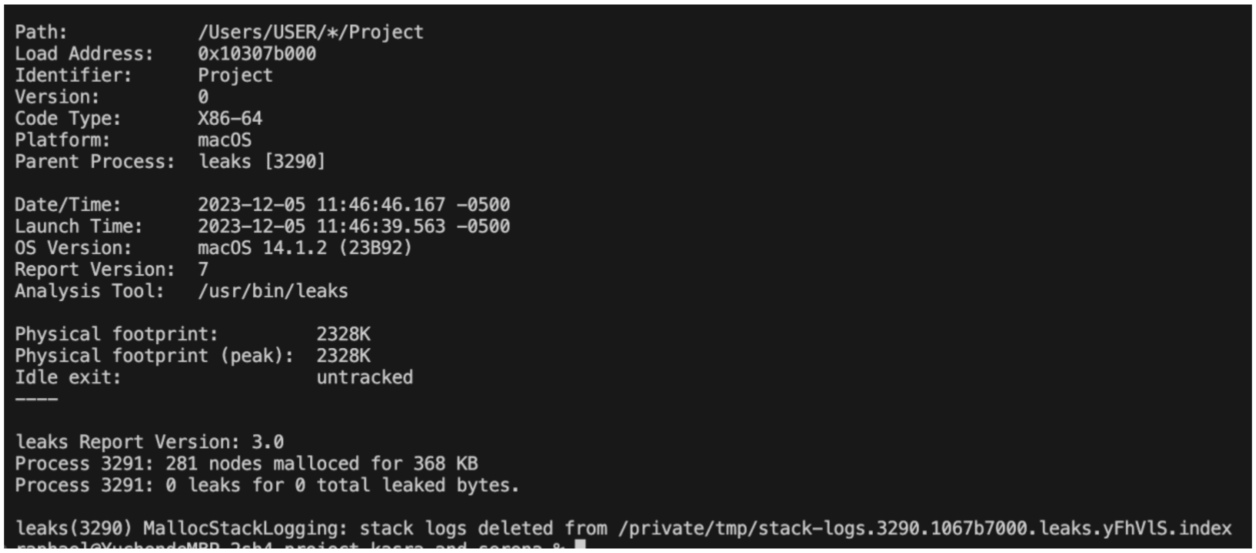
This might because of the main logic of this feature. The code represent check the collision first in the function ( myPlayer->updatePlayerDir(); ). Even though, they change the lose score to true, but they will still be going into movePlayer function. That is not the right logic to do that. After we get the lose score, we should not let the snake move.

- Something about quit the game:

After we pressed the "space bar", we should be able to quit the game, instead of still moving forward once and quit the game. That might because of the while loop, and the position of the (check the input).

2. **[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.

There is no Memory Leak.



```
Path:            /Users/USER/*/Project
Load Address:    0x10307b000
Identifier:      Project
Version:         0
Code Type:       X86-64
Platform:        macOS
Parent Process:  leaks [3290]

Date/Time:       2023-12-05 11:46:46.167 -0500
Launch Time:     2023-12-05 11:46:39.563 -0500
OS Version:      macOS 14.1.2 (23B92)
Report Version:  7
Analysis Tool:   /usr/bin/leaks

Physical footprint:         2328K
Physical footprint (peak):  2328K
Idle exit:                  untracked
----

leaks Report Version: 3.0
Process 3291: 281 nodes malloced for 368 KB
Process 3291: 0 leaks for 0 total leaked bytes.

leaks(3290) MallocStackLogging: stack logs deleted from /private/tmp/stack-logs.3290.1067b7000.leaks.yFhVlS.index
```

## 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.

For GitHub, every time when one person changes something and git push, if the other person also change something, the second person cannot git pull. Which will make the work harder.

Also, if some part of the base code is not write by me. It will take me some time to understand how the functions work. But, if there is comments or talk with my partner will help a lot.