COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Belal Armanazi/Aarib Syed

Team Members Evaluated Sydney Sochaj

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.

6/6 Each class in the program is clearly declared, with clear intentions on their usages. The classes have both sufficient private and public members, with clear naming conventions that display their relation to one another. The ability to predict how each class will interact with each other is clearly visible and have observed no negative aspects in any of the header files.

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.

5/6 The main program logic is very comprehensive and easily readable. Accurate predictions can be made about the outcomes of the program by observing the program loop alone for the most part. However, the end game procedure part of the code is slightly unclear. The first thing is the user is prompted to “Press ESC to exit” after the losing message appears. This is redundant and already prompted by the compiler. Also, there was no sign of the regular exit (non-losing) logic available in the main program loop.

```
if (myGM->getLoseFlagStatus())
{
    MacUILib_clearScreen();
    MacUILib_printf("YOU LOST!!!!\n");
    MacUILib_printf("Press ESC to exit.");
}
```

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.

5/5

Pros:

- Easier to predict outcome of program.
- Less procedural style of programming promoting conciseness and organized code structure.
- Enhanced visualization of game mechanics as a result of dedicated classes for different aspects of the game e.g. game mechanics class, player class, etc.

Cons:

- Removes procedural aspect, leading to the requirement of a wholistic understanding of many different parts of code rather than one big block or file.
- Required to maintain and debug several files and sections of the code, rather than one common section.

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.

4/5 Yes, the code offers sufficient comments to help me understand the codes functionality for both the C++ files as well as the header files used for OOD. There is a good number of comments, for major code blocks and where the main games functionality takes place. The one shortcoming I see however, in the code is that there are too many comments which serve no purpose. Many of these comments come from the project help sessions where the instructor adds comments for us programmer to better understand his code and what to do in certain areas of the code. These blocks of comments were used to communicate what to do for specific iterations of the code and once they are completed serve no extra purpose apart from reducing code readability. I would recommend deleting these blocks of comment for the code files as in a professional setting when your code is complete there should only be comments for the purpose of giving your colleagues an easier time understanding the codes functionality and not the steps you took to finish the block of code. I have provided images below which depicts the blocks of comments (not all of them) to be removed as they are not needed.

```
void Initialize(void)
{
    MacUILib_init();
    MacUILib_clearScreen();

    myGM = new GameMechs(26, 13); // make the board size 26x13
    myPlayer =  new Player(myGM);

    objPos tempPos(-1,-1,'o');
    myGM->generateFood(tempPos);

    // Think about when to generate the new food...

    // Think about whether you want to set up a debug key to call the
    //    food generation routine for verification

    // remember, generateFood() requires player reference. You will need to
    //     provide it AFTER player object is instantiated

    // this is a makeshift setup so i don't have to touch generateItem yet
    // you need to do this yourself
    // objPos tempPos(-1, -1, 'o');
    // myGM->generateFood(tempPos); // turn this into array list operation

}
```

```
void GameMechs::generateFood(const objPos& playerHeadPos)
{
    // generate random x and y coord, make sure they are NOT boarder or blockOff pos.

    // check x and y against 0 and boardSizeX / boardSizeY

    // remember, in objPos class you have an isPosEqual() method. Use this instead of comparing
    //      element-byelement for your convenience

    srand(time(NULL));

    while (true)
    {
        int foodX = rand() % (boardSizeX - 2) + 1;
        int foodY = rand() % (boardSizeY - 2) + 1;

        objPos tempPos(foodX, foodY, ' ');

        bool overlap = false;
        objPosArrayList playerPosList;
        for (int i = 0; i < playerPosList.getSize(); i++)
        {
            objPos playerPos;
            playerPosList.getElement(playerPos, i);
            if (tempPos.isPosEqual(&playerPos))
            {
                overlap = true;
                break;
            }
        }
```

```
class GameMechs
{
    // Construct the remaining declaration from the project manual.

    // Only some sample members are included here

    // You will include more data members and member functions to complete your design.

    private:
        char input;
        bool exitFlag;
        bool loseFlag;
        int score;

        objPos foodPos;

        int boardSizeX;
        int boardSizeY;
```

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.

4/4 Yes, the code offers a sufficient amount of white spacing between certain lines of code. The programmer provides an organized and respectable indentation and whitespace manner between function definitions, conditional statements, and class organization. Nowhere in the code exists crammed and very densely placed lines of code and offers program wide readability.

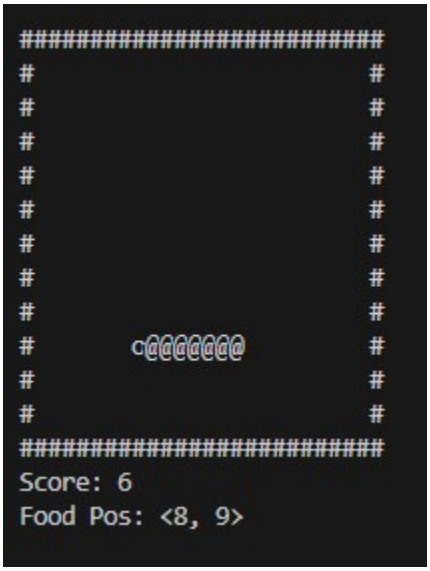
Part III: Quick Functional Evaluation

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)

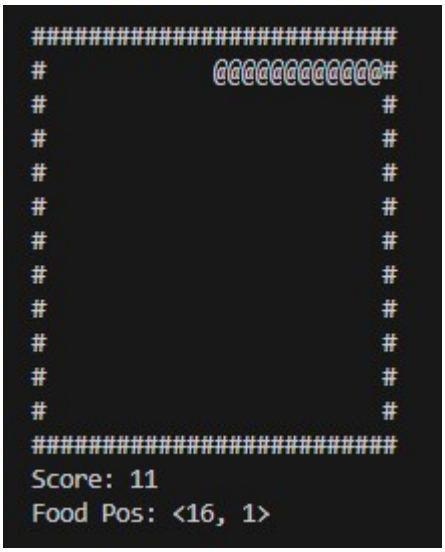
5/8 After compiling the code, running, and playing the Snake Game, for the most part it seems to be fairly smooth playing experience apart from 3 details I noticed. In the process of the playing, I realized that there was a slight bug in the game experience which comes whenever the snake head collides with the food on the game board. Every time there is a collision there is a slight delay which is visible to the user's eye, causing the game to almost freeze for a slight second. This causes the game to feel as if it is lagging whenever the food is consumed. The second bug I noticed was the random food regeneration on the screen sometimes overlaps with the snake itself. After the food is consumed and another food is randomly generated on the screen, a few times it is generated on the same position as the snake causing the food to be hidden until the snake moves it position. This causes the user confusion as they are not sure where the next food item is and thus creates a large bug in the playing experience. There is one other shortcoming I saw which isn't considered a bug but rather has to do with the user-interface which

is that there are too few player UI messages being printed out onto the screen. There is only score and food pos being printed out whereas there should have been a message for player position as well as direction. I have provided screenshots for both bugs taking place on the gameboard, however as they can be only truly seen on video by the naked eye, it is hard to truly show the bug taking place in a photo.

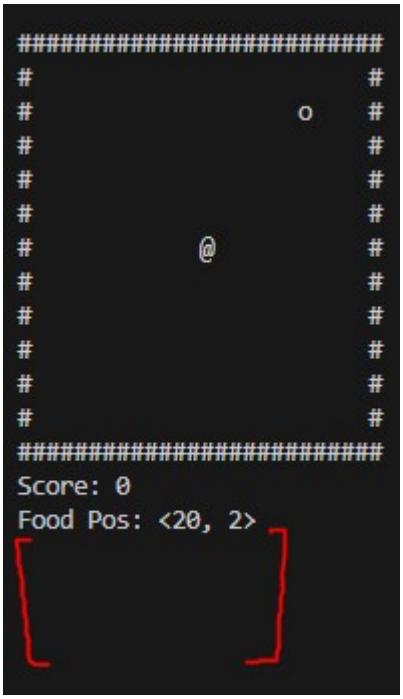
In order to debug these two found bugs, as I have also done the project I would first go and look into the Player.cpp file where the food collision takes place and then go into the GameMechs.cpp file where the random food generation takes place. For both cases especially the food collision, I would suggest going line by line in the appropriate member functions by using the GNU debugger and printing out the i locals for each code block. Making sure the loop is terminating properly and the conditionals used for the loop are correct. For the random food generation in GameMechs.cpp I would recommend creating debugging print statements in the DrawScreen() function within Project.cpp for both the foods position (already being printed) as well as the array list which is the snakes body position. By creating a for loop for each element in the array list and printing it as a debugging message will help figure out if the food is being generated onto the same position as the snake since we can compare the foods location to the snakes body. To fix the bug I would again recommend using the GNU debugger in the member function where the food generation is taking placing, going line by line and printing out the i locals to check and see where the root problem is likely taking place. For the UI messages I would recommend adding an extra two print statements into the DrawScreen() function for both the current player's position as well as the direction the player is currently in, as these two messages can help the playing experience.



(Moment where slight freeze occurs during food collision – hard to depict in a screenshot)



(Snake has collided with food, and a new food has been generated on the snake's body)



(Lack of player UI messages)

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.

6/6 No, the Snake game does not cause any memory leakage. After using the Valgrind memory check, the leak summary showed 0 bytes of lost memory in each category of memory leakage. However, there does seems to be 84 errors from 7 contexts in the overall memory profiling report, as well as 228,587 bytes still reachable. I believe that this doesn't have anything directly related to the code but instead the still reachable blocks may come from the OS execution framework for the underlying curses library. I have added an image of the Valgrind Leak Summary Report below.

```
==7223== LEAK SUMMARY:
==7223==    definitely lost: 0 bytes in 0 blocks
==7223==    indirectly lost: 0 bytes in 0 blocks
==7223==      possibly lost: 0 bytes in 0 blocks
==7223==    still reachable: 228,587 bytes in 131 blocks
==7223==         suppressed: 0 bytes in 0 blocks
==7223==
==7223== For counts of detected and suppressed errors, rerun with: -v
==7223== Use --track-origins=yes to see where uninitialised values come from
==7223== ERROR SUMMARY: 84 errors from 7 contexts (suppressed: 0 from 0)
```

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.

   Splitting up the code was actually much easier and more convenient than I had expected. We did not experience any interference issues or trouble communicating to each other. However, git pushing and pulling was extremely buggy and was not accurate. What I mean by that is, constant merge conflicts when git pulling, inconsistent updates when git pushing. Therefore, we had to resort to emailing each other all the code whenever someone had to git push. This was very time consuming. I would wish that Dr. Athar and Dr. Chen provided some sort of mini tutorial on git pushing and pulling or any general collaboration tips when it comes to VScode and github, as many of us in the course are new to these environments and tools, and have no experience in group software development.