COMPENG 2SH4 Project – Peer Evaluation

Your Team Members	Abdurahman Butt	Amna Khan
Team Members Evaluated	Kareem Odeh	Ahmad Molhim

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.

   Answer -

   Upon looking at the header files it is very clear to interpret the behaviours of each object involved in the program. I can see the food header and I know that that header file is going to handle the food, the GameMechs file is going to handle game mechanics etc. Also, the individual function names are very concise. For example, in the food file the function is called generateFood() and does exactly what it needs to. The positives are that it is clear and concise. The name conventions are consistent. Last positive is that there is error handling in the objPosArrayList file. One con I would say is that there are barely any comments throughout the code. Although not necessary at all it would help visually.

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.

   Answer –

   The main logic shows clear interactions between game-related objects like GameMechs, Player, and Food, using certain functions such as GetInput, RunLogic, DrawScreen, and LoopDelay. Some positives are that they have cleary initialized the game pointer and food pointer in the initialize, as well as player and food pointers. In addition the update and move player are clearly present in the RunLogic(). Also, a positive is that a PrintEnd() function is implemented. This is convenient if we had more than one method of losing so if we had to edit the lose statement, we can directly edit it in the function instead of looking for each instance. A con that I noticed is that there is a lot of of variables initialized in the DrawScreen(). I consider this a con because the DrawScreen function should be strictly for what its name implies with minimal additions that don't relate to the function (or none at all).

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.

Answer –

Project:

Pros –

- **Objects like GameMechs, Player, and Food interact through well-defined interfaces in the main loop.**
- **Game functionality is organized within classes, providing a clean separation of concerns.**
- **The code is organized into modular functions for distinct game logic components.**
- **Very minimal amount of global variables**

Cons:

- **Despite its advantages C++ is a more difficult programming language to learn than C.**

PPA3:

Pros –

- **Everything is on one page, so you don't have to navigate through multiple files.**

Cons –

- **Since everything is on one page it is also an eye sore and less organized**
- **There are a ton of global variables.**

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.

   Answer-

   There is little to no comments on the code or self-documenting coding style. I would add comments to help viewers understand the code. Although there are no comments the function's names are very concise. For example, PrintEnd() clearly is a function to print the end game statement. Also, FoodPos object is used for food position, GMPtr is a pointer to the Game Mechanics, etc. So, despite not having any comments the naming conventions are nicely done.

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.

   Answer –

   The code follows good indentation throughout the code but one thing it can touch on is add sensible white spaces. There are whitespaces in the code but it is inconsistent. In the updatePlayerDir() the switch case has blank spaces that are visually appealing and easy to differentiate. Also, in the initialize, runlogic, main all have the brackets under the statements for example,

```
---------------------------------------------------------------------------------------------------------------------------
int main(void)
        {
           Initialize();
---------------------------------------------------------------------------------------------------------------------------
```

This convention is nice as its not clustered. They are separated and easy to read. Whereas, in the DrawScreen()

```
---------------------------------------------------------------------------------------------------------------------------
        for (int i = 0; i < GMPtr->getBoardSizeY(); i++) {
              for (int j = 0; j < GMPtr->getBoardSizeX(); j++) {
                 drawn = false;
                  for (int k=0; k< playerBody->getSize(); k++){
                    playerBody->getElement(tempBody, k);
                    if(tempBody.x == j && tempBody.y == i){
                       map.setObjPos(i, j, tempBody.symbol);
                       drawn = true;
                       break;
---------------------------------------------------------------------------------------------------------------------------
```

this a bit clustered to look at and isn't the best when trying to differentiate between lines.

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)

Answer –

Overall, the code runs well without any major noticeable bugs when running the program. However, I did notice two small issues in the code which were that the game runs a bit slow compared to what we would expect. This is not a huge issue, but I personally feel it creates a less enjoyable playing experience. Additionally, there is a constant display of "YOU LOST, game over", which is incorrect as we haven't lost yet. Both these issues can be attributed to one part of the code, which is the implementation of a PrintEnd() function. I believe this function was implemented so that when the program ends on account of self collision, a lost statement is printed, and a longer time delay is implemented so it is easier to see before the program ends. However, this function should only be executed if the Lose Flag is set to true, which they have implemented. It appears the Lose Flag is set to true right at the beginning of the program which is most likely the cause of this issue. To solve this issue, I would venture to add some print statements throughout the program to determine where the Lose Flag is set to true, or analyze the initialization of the Lose Flag and see whether it begins at true or false.

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.

Answer –

The teams snake game does not cause any memory leakage. They have properly freed all memory allocations for correct memory handling in their code.

```
~~Dr.M~~       0 unique,     0 total unaddressable access(es)
~~Dr.M~~      11 unique,   100 total uninitialized access(es)
~~Dr.M~~       1 unique,     1 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)
```

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.

Answer –

Working with a partner had many benefits to this final project. To list a few, we were able to bounce ideas off of each other and talk through our logic for certain implementations which helped us debug and make efficient coding decisions throughout the project. Additionally, we were able to pass the code between us depending on our strengths, and if one of us couldn't understand one part, the other could help and vice versa. Also, since it was a big project, and near finals time, it was helpful to split the workload of a big project between two people. One of the downsides was file sharing. This was our first time working on a project where you had to pull edits from a repo throughout the project, and it wasn't the best experience for us. Since we couldn't work on a file together, we had to git pull and git push our projects, and when we were both working on stuff simultaneously, we had a lot of problems with merging edits and conflicts with the git pulls. Otherwise, this project experience was great!