COMPENG 2SH4 Project – Peer Evaluation

Your Team Members	Anuja Perera and Jack Vu
Team Members Evaluated	Deyontae Patterson and Victoria Black

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.

```
#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
```

Figure 1: ObjPos class

```
void getPlayerPos(objPos &returnPos); // Upgrade this in iteration 3.
int getPlayerDir();
void updatePlayerDir();
void movePlayer();
void growPlayer(objPos lastpos);
void checkFoodCollision(objPos prev);
void checkSelfCollison();
void printSnake();
int snakelen();
```

Figure 2: Player Class Header

When looking at their code it is easy to understand how each of the objects interact with each other. For example, how the objPos class and Game Mechanics class interact with the player class.

Looking at the screenshot of Figure 1 and Figure 2. We can see that the functions inside of this class are named well so that we understand what type of behaviours an object can have. For example, we can see the constructor gives every object a xPos, yPos and symbol on the board. We can also see that this object can do collision detection. Looking at the game mechanics.h file we can see that it interacts with the objPos object in the generateFood and getFoodPos functions. It is easy to interpret how the isPosEqual function of objPos and getFoodPos(objPos &returnPos, int index) of gameMechs interact with each other to do collision detection. We can also see how both of these objects interact with the player class in checkFoodCollision() and generateFood().

Some negatives of the code may be that the lack of spacing in the player.h file. Empty lines between function definitions and grouping together similar functions makes it easy to read. Other negatives may be the lack of comments in the header files. For example, it is hard to understand what objPos* check; means in the player.h file.

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.

It is easy to understand how the objects interact with each other in the program logic because everything is labelled with appropriate titles, allowing us to understand and easily follow through their steps. It is easy to follow along as the functions that are called are well named and called in a logical order. When going deeper in the main function it is easy to identify the interaction of the objects for example in run logic we can easily interpret how position objects are spawned and checked for collision. We can identify how each of the objects are used, such as GameMechs and Player. The group decided not to create a Food object, but they were able to seamlessly have their pointers used appropriately and be easily interpreted.

The group included a scoreboard and player statistics feature, taking the game above and beyond but the program logic for this became messy. The negative features of the main program loop would stray from their above and beyond features, complicating the main program loop and making it messier as you scroll down. This is where implementing another class would help, with OOD the main program loop would be much cleaner and allow for other users to easily use the created objects.

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.

- Pro: OOP encourages breaking down the program into smaller self-contained modules that are easy to reuse
- Pro: The code is more organized, and it is easier to fix and test for bugs in the code
- Pro: It is much easier to write code as partners as each team member can work on the functionality of a different class

- Pro: New classes can inherit properties and behaviours from other classes, allowing the main program loop to be very high-level and understandable. Which makes it very easy to break down the logic of the game.
- Con: It takes a lot of time to get used to the multi-file nature of OOP and to fully understand how the objects interact with each other
- Con: In some cases, OOP led to unnecessary complexity especially in a simple project like this. For example, generating multiple food objects on the board was much easier in procedural programming than in OOP.

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.

   Yes, the written code offers sufficient comments and deploys a coding style that enables readers to understand the codes' functionality. The group was excellent in providing comments before diving into any task, making it super simple for us to understand. Their comments weren't extensive but were sufficient in providing a brief idea of what's happening. Additionally, with their coding style, we were able to quicky follow along and break down what's happening in 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 the written code is extremely easy to read. For example, in the player.cpp fie there is adequate new lines between each part of code. For loops are formatted properly and there is good indentation so that it is easy to understand what is in a conditional statement or loop. New Line formatting is also used when printing out debugging statements. 1 shortcoming may be the lack of white spaces in the header files for example player.h as discussed in question 1. Overall, there are very few shortcomings. 1

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)

The snake game offers a smooth, bug-free playing experience for the most part. The only "buggy" feature that we ran into was their above-and-beyond features. They implemented a feature where the user gets to enter their own length and widths for the borders of the game but there are some

problems with it. When you enter 6 as the length, it becomes 30 long without telling the user that there is a 10 length and 5 width minimum. To fix this, using an if statement along with the “What length?” question, telling the user that the value must be a minimum of 10 would make the experience better. Additionally, having different types of food in the game is great but printing what each type of food does would enhance the experience for the user. We spent a great deal of time trying to figure out what each type of food did. This can be implemented through printing short statements in the Project.cpp file, using MacUILib_printf.

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.

   No, there are no memory leaks.

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.

   Anuja:
   The Project code was easy to work on in a partnership. We were able to communicate with each other to help fix bugs. The code being modularized helped it to be easy to work on however sometimes it was difficult to understand code the other person had written but it became easier as we wrote more code. Also being able to git push and git pull each others code was also very helpful. It was fun and not too difficult.

   Jack:
   Working with my partner to collaborate on this software development was a great experience. Having the code be split up into sections allowed us to work on our own parts while not affecting the others in the meantime. Although we had different coding styles, after communicating and using comments before git pushing our codes, it was a good experience working with Anuja. I also liked the fact that having someone to code with allowed for faster debugging as our opinions and thinking are very different, allowing new solutions to come up.