COMPENG 2SH4 Project – Peer Evaluation

Your Team Members: Daniel Oliveira, Abrar Tamjid

Team Members Evaluated: Aditya Hura, Mohammed Zeeshan Ansari

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.

   After taking a glance at each header file, the naming convention is good in the sense that it accurately depicts the role of the method or variable within the class. The classes are all organized in a manner that makes sense (private and public followed by constructors, getters, and setters. There are comments to explain the methods with more ambiguous names. A positive feature that was observed is the fact that there is a separate class that generates food. Not having it in game mechanics class makes it so that each class is more independent of each other which makes testing easier and improves modularity. A negative concern is the red squiggly lines causing the program to not compile. This could have been avoided if they correctly included the header files and paid more attention to detail.

   Evidence of comments that clarify the purpose of methods or variables:

```
public:
    SnakeFood(); // Constructor to set xRange and yRange

  // Need a reference to the Main Game Mechanisms
  GameMechs* mainGameMechsRef;
  SnakeFood* snakeFood;
```

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.

   The main logic makes it clear how the objects in the code interact. As previously mentioned, the naming convention played a big part of the code readability. The main program was very concise due to the nature of OOD. the Drawboard function had to most lines, but it was broken up into sections and comments were added to make it clear what was happening at each step.

There is some confusion when drawing the border, the first if statement seems unnecessary and redundant but there is no explanation as to why it was used.

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:

- Enhanced code reusability through inheritance and polymorphism.
- Encourages better organization and encapsulation of data and functionalities.
- Supports better scalability and maintainability for larger projects.
- Facilitates parallel development by enabling multiple programmers to work on different modules concurrently.

Cons:

- Initial learning curve for beginners transitioning from procedural to object-oriented programming.
- Overhead associated with designing class hierarchies and relationships.
- Potential performance overhead due to dynamic polymorphism.
- Can lead to complex designs if not properly planned or implemented.

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 code does not have any comments outside of the header file and the main program. This did not impact our ability to understand their program as we have also done the project but, it will hinder someone else's ability to fully understand the program. A way to improve this is to pretend to be a beginner programmer, if the purpose of a function is obvious, no comments required. If it a little unclear, comment on the purpose and how it was achieved thorough out the function.

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.

The code follows good indentation and has sensible white spaces. It also uses new line formatting which helps with the readability. They also use add a space before and after every operand which helps a lot with the 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)

The Snake Game does not compile and thus evaluating the playing experience is not possible. The errors responsible for causing an incomplete compilation are as follows: The code attempts to declare a pointer without including the necessary header file that defines the type it's pointing to. In particular, the code attempts to use composition within one of its classes by using an object of another class as a private data member without including the other class's header file. This leads to an incomplete type error during compilation since the compiler doesn't recognize the referenced type. The other issue with the code is a syntax error, where the implementation of a class method is missing a closing tag (e.g. '{').

As instructed, we will be evaluating our own group's relevant part for this question.

Our code does offer a smooth playing experience. Relative to the board, the snake is moving at a reasonable speed and abides by the wrap-around logic and in-game food collection mechanisms. However, there are some rare, occasional bugs that do occur and are associated with the bonus iteration of the project. The first bug is that at unknown intervals of time, the game will generate 4 food objects on the board rather than 5. It is hard to pinpoint the exact cause for this bug from inspection; it could be that 4 objects are being generated instead of 5 or that one food object is being consumed while the other 4 do not disappear, making it appear that the game generated 4 food objects instead of 5. This issue however seems to rarely occur, and the intended feature of randomly spawning 5 foods upon colliding with any one of them, works for almost all the time.

The second bug involves the intended feature of one of the 'superfoods' for the bonus iteration. One of the superfoods, denoted by a '+', is meant to increase the players score by 10. As the previous bug, its intended feature works perfectly almost always, however in some rare cases, the snake will freeze in place after colliding with the superfood, and the users score in the bottom will rapidly increase by 10 multiple times.

It is difficult to pinpoint the exact issues to both bugs, but two possible solutions could be to either increase the game delay to combat rapid food collection or to implement the deletion and regeneration of the food bucket in the run logic procedure of the main project, rather than implementing them within the player class.

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.

Generating a memory leak report is not possible since the code does not compile. We will be evaluating our own memory report instead.

Our code does not cause any memory leak. There are however some instances of 'total uninitialized access.' Upon inspecting the root cause of these issues, they are all associated with 'KERNALBASE' which appears to be some sort of library or 3rd party issue outside the scope of our code. None of the reported issues can be pinpointed to a genuine location within our code (e.g within a class or the project.cpp), and so, these issues displayed by the Dr. memory report are likely negligible and independent of our code.

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.

I had a great experience in my first collaborated software development project. Throughout high school and even in my first year of university, I was continuously exposed to one-person team projects and assignments. The 2SH4 project was insightful in the sense that it made me realize the nature of the industry – how working with a team is an inevitable necessity. As such, the 2SH4 project exposed me to a great opportunity to hone my collaborative and coding skills and help prepare me for the industry environment. I found it less difficult to work on my assigned tasks of the project, though when it came to re-grouping with my teammate and putting what we build together, I struggled. I was not used to having to actively interpret and use another person's code with my own.