COMPENG 2SH4 Project – Peer Evaluation

Your Team Members ___David Carnogursky__ __Ethan Huynh_____

Team Members Evaluated _____Raj Pandya_____ ____Tirth Nagar_____

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.

Looking at the header files for each object, the naming convention makes sure that the names provide an accurate description of the variable or method being used. It is fairly obvious at a first glance what each object behavior is generally supposed to accomplish, and how it might interact with other objects in the program. There are however little to no comments within the header files of each object, which could potentially be confusing for someone with little to no coding experience.

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.

Interpreting the way the objects in the code interact was fairly straightforward, the code utilized easy to understand naming conventions, making each object and object method within the main program loop easy to understand at a first glance. The main program loop in general was concise, and used minimal lines of codes to accomplish tasks. Specifically the draw board function in their main program loop was found to be elegantly done. There was one line of code within GetInput that provided no comments for explanation, and through some brief testing, was found to be seemingly useless for the functionality of the program. This was a minor point of confusion in an otherwise very easily understandable main program loop.

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:

- Smaller size of the main program loop.
- Easier readability if naming conventions are used correctly
- Easier to collaborate using the OOD approach
    - Modular design lets team members work on sections of code with less communication between team members

Cons:

- Can be more difficult to develop/understand, requires more strict organizational skills
- Can be more difficult to debug if not developed incrementally

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 contains a plethora of comments in every object that thoroughly explain the logic and functionality of every function. The comments were well written, and made the code very straightforward to read through. Occasionally the comments were mildly excessive, highlighting obvious functionalities or providing repeat information. However this was not a major detriment to the overall quality of code organization.

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 organizational practices. The use of white space and newline formatting makes the code easily readable, and rarely ever feels confusing or overwhelming to look at. The code is often organized such that it takes up a minimal amount of lines without being visually crowded. Indentation is consistent throughout, and makes the logic within the code much easier to track.

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 provides a smooth and bug-free experience. We could not find any bugs while playing the game normally ourselves. However we did notice a potential error within their code. The maximum size of the player position array was set to 200, however the total area of the game board was set to a much larger number. This means that after playing the game for long enough, the snake could theoretically exceed a size of 200 and cause a segmentation failure from trying to access out of bounds information. The other features of the game all seem to be functional, with proper food generation, self destruction, and game over screen.

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.

The snake game does not cause memory leaks. The memory leakage was accounted for within the destructors of each object, rather than the main program loop when compared to our group's code. The code only utilized two instances of heap data in the player object and objPosArrayList object that were accounted for in their respective destructors.

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.

David - I really enjoyed my first experience in collaborative software development because it improved my knowledge and skills with the coding language. My partner was far more experienced than me with c and c++ and completing this project side by side with him helped me fully understand the topics that were taught in class when going through each iteration and creating our code. The thing I struggled with most was knowing what to write down as code to complete each iteration, although my logic and ideas of what to use were spot on. Having a partner greatly benefitted my learning because we would combine our knowledge and through trial and error we would come out with a working code for each iteration. Overall I really enjoyed this team project and it greatly improved my skill with coding. I hope that for future coding projects we keep having the option to work with someone as a team.

Ethan - Overall this was a good learning experience in many aspects. Software development is not something I'm used to doing collaboratively, so it was also challenging in a few ways. I think that our initial strategy of splitting up the work evenly worked out at the beginning of the project, where we were developing objects independently. However, moving further into the project we needed to work in sync more often, especially when discussing how to incorporate each of our objects into the main program, and making them compatible with each other. Another interesting thing I found was that debugging was significantly easier when collaborating with my team members. Just being able to bounce ideas off of each other for what could be going wrong, and combining our coding knowledge helped immensely. I think the biggest change from coding on my own however was organization. I had to care much more about naming conventions and code organization or else explaining my sections of code to my team members was very difficult. Overall it was an enjoyable learning experience.