McMaster University

COMPENG 2SH4 Project – Peer Evaluation

Your Team Members __Harshith Duba____ ___Adam Smith__

Team Members Evaluated ___Aliyah____ ___Jenisha_____

Provide your genuine and engineeringly verifiable feedback. Ungrounded claims will lead to deductions.

Part I: OOD Quality

- **[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.

  - **Yes, all header files provide a clear modularization of the program. We can clearly see classes represent distinct objects with well-defined behaviors. All interactions between objects, such as game mechanics, food interacting with the player being well structured. Some positive highlights include well encapsulated classes, keeping code modular and maintainable, good use of enums in the Player class. However, the checkSelfCollision function in the Player class takes reference to objPos but doesn't provide sufficient information about its purpose or how it should be used. This could lead to potential misuse or undefined behavior.**

- **[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.

  - In the main program loop all code is easily interperetable and seeing the interactions between the objects is easy. One reason for this is that they have chosen sensible names for their objects and functions. One negative thing they did was on Project.cpp line 105 they called object->getPlayerPos() for no apparent reason as they already made playerBody and called that there, as well as the fact that on its own it returns the pointer and there was no pointer equating to it.

- **[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:

- Code modularity improved greatly

- Having every instance of an object contain all methods and states uniquely makes getting the behaviour you need easier.

- In procedural design the development is much more linear, where each feature is developed after another, in OOD the development can be more nonlinear.

- Troubleshooting is made much easier as the compartmentalisation makes it easier to track down the source of errors

- Any object class can be reused for a purpose that it was not originally meant for but is still applicable, speeding up development

Cons:

- Development requires more thinking in understanding the behaviour and interaction between each object and the program logic

- Objects can behave in less predictable ways than procedural coding so one has to take more care when programming with them

- Objects can cause memory leakage when interacting with eachother more often than in procedural and therefore must be programmed with more care

- Coding with OOD can take much longer.

- OOD is not always applicable to every problem, while procedural writing usually is.

Part II: Code Quality

- **[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 offers mostly sufficient comments but I was still left confused at the purpose/function of some lines. One such example is the point of calling loseFlagStatus on Project.cpp line 160. I was also unable to find a manual exit button within their code. I would improve it by better explaining some more complex code as well as removing any possibly redundant info that only confuses the reader of the code.

- **[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 does follow generally good indentation except in the case of Food.cpp lines 33-37 where it is wrong, just indent the inner lines to fix it. Their whitespace use is good throughout and the code is easily readable. The formatting of the output is well made and functions correctly.

Part III: Quick Functional Evaluation

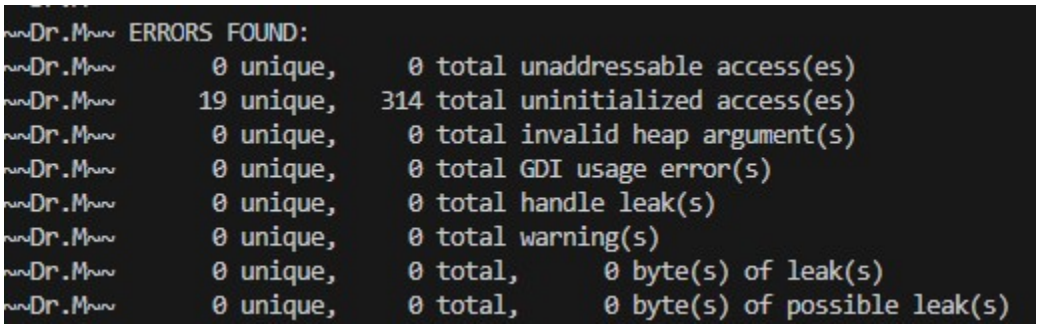
- **[8 marks]** Does the Snake Game offer smooth, bug-free playing experience? Document any buggy features and use your COMPENG 2SH4 programming knowledge to propose the possible root cause and the potential debugging approaches you'd recommend the other team to deploy. (NOT a debugging report, just a technical user feedback)
  The snake game was extremely smooth and had only one error which we noticed. The food does not seem to be able to generate on the rightmost and furthest down spaces. Through analysing the Food.cpp and Project.cpp code it is seen that their food generation method is flawed and only generates from 1-18 to the right and 1-8 on the left while their game board is 26x13. This results in the behaviour described above.

- **[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.

I could not find an exit flag so I had to add one for the memory report to be generated

```
67  void GetInput(void)
68  {
69      if (MacUILib_hasChar() != 0) {
70          input = MacUILib_getChar();
71          myGM->setInput(input); //using setter method to collect input
72
73          if (input == 'g' || input == 'G') { //debugging
74              // Clear away the current food
75              // Generate a new random position for the food
76              Player myPlayer(myGM, myFood);
77              myFood->generateFood(myPlayer.getPlayerPos());
78
79          }
80          if(input == ' '){
81              myGM->setExitTrue();
82          }
83      }
84
85  }
86
```

{Added 80-82}



There is no leakage inside the program and therefore all of the heap allocated memory was correctly freed/deleted.

Part IV: Your Own Collaboration Experience (Ungraded)

- 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.

This experience has been overall a positive one, working with another person was invaluable in problem solving and troubleshooting. One of us had worked for several hours on one computer to get a working prototype of the game without the bonus implemented, it worked on his computer but not the others. This problem could only have been spotted with the two people working together and using this information that error was fixed. Overall this was a good learning experience for coding with multiple collaborators on a more complex problem.