COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Imad Ali Gurmehr Bagga

Team Members Evaluated Youssef El Ashmawy Zahra Kazmi

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

Part I: OOD Quality

1. **[5 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.

All the files have the basic names given to them at beginning, except the Food file, which carries the Food class. It makes sense to have the food class in a separate file although it can be done in the gameMechs file as well since there is not any extra features added. The names for each file is sensible since almost each file represents a different class like the Player class and file, and having more descriptive names would hinder the development process, by narrowing a developers ideas. There is not any negatives I can see in the names itself.

2. **[5 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.

With a brisk and later more concentrated look through the program it is very apparent how each object interacts with eachother. FIrst skimming through the code, All but the draw screen function can easily be interpreted. Later looking through more concentrated the draw screen can also be read and interpreted. All function names have been chosen to represent their actions. Negatives are that in odd cases without looking at the class I cannot tell the point of certain lines of code, an example is in the draw screen function:

```
    grid.getObjPos(grid);
    MacUILib_printf("%c", grid.getSymbol());
```

I do not understand the purpose of the first line if grid already holds the objPos needed then why did they get the position again. Although we are nitpicking for even this example, besides this everything else is self explanatory.

3. **[3 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:

- More easily adaptable and small changes can be made to drastically change the program.
- Having more functions leads to a much simpler main program which at times could even be understood by non developers.
- Less repeated code.
- Easier to follow code, and debug at times.
- Easier to collaborate.
- Bottom up programming.
- Professional looking code.

Overall I believe the C++ OOD makes the ladder half of the programming process much easier, since you already have the materials to make the rest of the code. Having a greater number of functions and more thought out interactions between objects also makes the program more adaptable. By adding another class or a couple functions the original prorgam can be enhanced much further. Also in terms of debugging OOD makes it easier to debug problems in the main program, and at times following a bug to it's source can lead to fixing a larger bug effecting the code. A large bonus to OOD is the collaboration aspect, in that different members of the project can work on different parts at the same time. WIth focus on collaborating and conversation's about the code with your partner it can make the programming process a lot faster and more effective. programming bottom up can be a asset or a liability depending on how well a developer understands the task. In developing bottom up I believe a greater understanding of the task and overall how the objects will interact with one another is needed compared to top down programming, where a developer can just focus on the execution. Although there are many benefits there are also downsides to it as well.

Cons:

- Debugging
- More time consuming

The biggest problem in programming I faced was the bugs, after writing only a couple lines the next 5-20mins are taken by debugging. One small problem can lead through 3 different functions interacting weirdly. Before I said the ladder half of programming is made easier with OOD, but the first half is much harder. The first half requires a great understanding of the task and can get confusing. The back and forth of debugging is what made OOD problematic for our group. Besides the bugs the only other problem I saw was the amount of time it took, since in making the code more adaptable we are sacrificing time. Simply instead of having a few functions and variables, everything is a object with it's respective members and properties and at times all that is not needed. PPA2 can be made into a snake game by adding a food system similar to the one from PPA3 and adding a 2d array with all x coordinates in one and y in the other. I believe both approaches of development to have their pros and cons, I would use OOD for larger projects especially if I have partners or I want to change the prorgam later, and for the procedural design process I would favour it for simpler or quicker programs or to test code separated from the original program.

Part II: Code Quality

1. **[4 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.

There is very few comments albeit having comments was not in the rubric, it is good practise to have them for more dense sections of the code. Simply adding comments would make reading the program a lot easier, and this could come in handy months later if the group was to rework their program. Something I would change is that at times there is new variables added in the middle of the function which should be created at the beginning.

2. **[3 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.

Honestly the indentation and everything is fine. There is no problems with it, and adding more white space would be pointless since typically each section of code should work one thing at a time, which their code does.

<u>**Part III: Quick Functional Evaluation**</u>

1. **[6 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 program runs smoothly, Bugs I noticed were that randomly there is a chance for food to spawn ontop of the snake and be hidden until the snake moves off it. And the team forgot to put in the exit key so pressing space does nothing and the only way to lose is eating yourself.

2. **[4 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 memory leakage.

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.

This was not my first time collaborating with a team on a coding project, albeit this was my first in university. Not knowing the partner you are completing a project with changes a lot, since not only are you learning about programming your learning abut eachother as well. Many aspects of the project caused problems like not knowing eachothers skills and having opposing views on the process of programming. Early on this brought problems with sectioning the program and later on in debugging eachothers code. Having a partner also caused us to delay certain portions of the code till the other person had completed theirs. Benefits include the debugging process as well as effectively cutting the amount a person has to program by half. Next time around I would further get to know my partner early on as well as looking at their previous work to deduce what they are good at.