COMPENG 2SH4 Project – Peer Evaluation

Your Team Members	Kevin Huang	Nathan Lo
Team Members Evaluated	Ecem Heywood	Erika Schaible

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, it is easy to determine the purpose of each function in the program. The function names are well named and self-documenting. One exception to this is getPlayerPos in Player.h. This returns the entire objPosArrayList, and I believe the title of this function is misleading, as it implies that only one objPos should be returned. It could be named getBody instead.

The player object could be put inside GameMechs, so that GameMechs contains an instance of the player. This is a better method of code modularization, as it no longer requires pointers to be passed around.

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.

In the main program loop in Project.cpp, most of the functions can be easily interpreted. However, the DrawScreen function is very confusing. Helper functions could be used here to reduce the amount of effort required to interpret all the for loops and if conditions. For example, the for loop used to check if a player is on a space could easily be put into a helper function. This would cut down the nesting, greatly improving readability.

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.

OOD pros:

- Keeps code readable
- Adding new features is made much easier with inheritance
- Code can be easily reused
    - Ex. The player class from this game can be easily moved and modified to fit another 2d game

- The use of the access level keywords allows the programmer to define how a user can use their class, without introducing vulnerability through direct variable access
- Easy to define interactions between objects

OOD cons:

- Not useful for simple programs
- The more complex the program, the more files will be added
- In complex projects, it can be difficult to parse and interpret all the different classes

Procedural pros:

- Useful for small programs
- Easy to interpret (assuming it's a simple project. If it were more complex, OOD should be used)
- Calling functions from objects can create overhead, especially as classes get inherited
    - Functions calls must be traced backwards if not overridden
- Easy debugging

Procedural cons:

- Adding functionality is not as easy
    - Must determine interactions with other functions, need to copy paste if something similar, but not the same is needed. Inheritance solves this issue
- Not easy to export functions to use elsewhere
    - Interactions between functions is more direct, and harder to decouple from the project than classes in OOD
- No access levels means all variables can be accessed, meaning it is harder for a programmer to manage interactions between functions

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 features a good amount of comments but it lacks comments in the header files other than the comments that were already there as the template of what to do. As the header files seem like a good place to understand the code structure on a high level, it is useful to have them commented, especially when there are more than 2 members in a class.

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.

Some of the getter and setter methods can be done in one line, saving space and improving readability. Some examples are the getBoardSize methods in GameMechs.cpp. Also, the curly braces can be put in the same line as the function declaration, rather than the line below, saving space without sacrificing readability.

Most functions are written like this:

```
void function()
{
...
}
```

But could be written like:

```
void function() {
...
}
```

Getters and setters are written like this:

```
int GameMechs::getBoardSizeX()
{
      return boardSizeX;
}
```

When they could easily be written like this, without sacrificing readability, as the function name is self-documenting

```
int GameMechs::getBoardSizeX() { return boardSizeX; }
```

Additionally there are random whitespaces scattered throughout the project that harm readability such as in GameMechs.cpp on line 33 where there is a random space in the function prototype. This also extends to newlines where in Player.cpp there are 9 newlines at the end of the file. There are also places in the codebase where the indentation is not consistent, such as in Project.cpp lines 60 and 72 where instead of the 4 space indent used everywhere else, the lines are indented by 3 spaces. The code uses a lot of nesting, which harms readability, as it becomes hard to track all the loops and conditions. The worst case of this is in Project.cpp, at line 103.

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 game runs smoothly, everything is as expected. There is a bug where holding down an input "locks" your character to move in a certain direction, but this is caused due to ncurses, which is only used in Linux. Therefore, I cannot say anything wrong was done.

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 memory leaks occur when running the game normally. The classes that use a heap allocation with the "new" keyword all have a delete statement in their destructor. These would be objPosArrayList, and Player. In Project.cpp some heap allocations are used for the GameMechs and PLayer object but these are properly handled by deleting them in the CleanUp function.

```
nate@DESKTOP-9BFCHIA:~/COMPENG_2SH4/2sh4-project-ecem-heywood-and-erika-schaible$ valgrind --leak-check=full -s ./Project
==36393== Memcheck, a memory error detector
==36393== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==36393== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==36393== Command: ./Project
==36393==
==36393==
==36393== HEAP SUMMARY:
==36393==     in use at exit: 178,748 bytes in 216 blocks
==36393==   total heap usage: 233 allocs, 17 frees, 264,621 bytes allocated
==36393==
==36393== 9 bytes in 1 blocks are possibly lost in loss record 8 of 69
==36393==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==36393==    by 0x4BB358E: strdup (strdup.c:42)
==36393==    by 0x489ED54: ??? (in /usr/lib/x86_64-linux-gnu/libtinfo.so.6.3)
==36393==    by 0x48A2098: _nc_tiparm (in /usr/lib/x86_64-linux-gnu/libtinfo.so.6.3)
==36393==    by 0x4872238: newterm_sp (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==36393==    by 0x4872B4C: newterm (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==36393==    by 0x4872BDF: initscr (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==36393==    by 0x109E94: MacUILib_init() (MacUILib.c:59)
==36393==    by 0x10A59C: Initialize() (Project.cpp:49)
==36393==    by 0x10A54F: main (Project.cpp:31)
==36393==
==36393== 24 bytes in 1 blocks are possibly lost in loss record 28 of 69
==36393==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==36393==    by 0x4C2B18E: tsearch (tsearch.c:337)
==36393==    by 0x489ED6E: ??? (in /usr/lib/x86_64-linux-gnu/libtinfo.so.6.3)
==36393==    by 0x48A2098: _nc_tiparm (in /usr/lib/x86_64-linux-gnu/libtinfo.so.6.3)
==36393==    by 0x4872238: newterm_sp (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==36393==    by 0x4872B4C: newterm (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==36393==    by 0x4872BDF: initscr (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==36393==    by 0x109E94: MacUILib_init() (MacUILib.c:59)
==36393==    by 0x10A59C: Initialize() (Project.cpp:49)
==36393==    by 0x10A54F: main (Project.cpp:31)
==36393==
==36393== 168 bytes in 1 blocks are possibly lost in loss record 52 of 69
==36393==    at 0x484DA83: calloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==36393==    by 0x489ECCD: ??? (in /usr/lib/x86_64-linux-gnu/libtinfo.so.6.3)
==36393==    by 0x48A2098: _nc_tiparm (in /usr/lib/x86_64-linux-gnu/libtinfo.so.6.3)
==36393==    by 0x4872238: newterm_sp (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==36393==    by 0x4872B4C: newterm (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==36393==    by 0x4872BDF: initscr (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==36393==    by 0x109E94: MacUILib_init() (MacUILib.c:59)
==36393==    by 0x10A59C: Initialize() (Project.cpp:49)
==36393==    by 0x10A54F: main (Project.cpp:31)
==36393==
==36393== LEAK SUMMARY:
==36393==    definitely lost: 0 bytes in 0 blocks
==36393==    indirectly lost: 0 bytes in 0 blocks
==36393==      possibly lost: 201 bytes in 3 blocks
==36393==    still reachable: 178,547 bytes in 213 blocks
==36393==         suppressed: 0 bytes in 0 blocks
==36393== Reachable blocks (those to which a pointer was found) are not shown.
==36393== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==36393==
==36393== ERROR SUMMARY: 3 errors from 3 contexts (suppressed: 0 from 0)
nate@DESKTOP-9BFCHIA:~/COMPENG_2SH4/2sh4-project-ecem-heywood-and-erika-schaible$
```

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.

Nathan:
This collaborative experience has been good, but there are some concerns. I learned a lot about using git collaboratively (although this was mostly my partner showing me, as he is more experienced with git). It was also a valuable opportunity to test my communication skills. However, there are some downsides to the collaborative approach. My partner lives in the same house as me. Had this not been the case, communication would be slim, and the project would be made more difficult, especially with conflicting git commits. Certain features such as adding an instance of the Player class to GameMechs requires more communication than most students are willing to commit to.

Kevin:
This collaborative experience has been pretty good, being roommates helped a lot as it allowed greater communication. As this was not my first collaborated software development experience, I applied a lot of what I learned from past experiences like proper commit messaging and merging to this experience.