COMPENG 2SH4 Project – Peer Evaluation

Your Team Members Ahmed Khaleel Tharshigan Vithiyananthan

Team Members Evaluated Malek Abouelkhair Mohamed Alkouni

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.

   Based on the header files of each object, each object is easily interpretable as the data members and functions within the class are named such that the purpose of each data member and function is easily identifiable.

   The deconstructors are properly defined as methods with the correct function signature for any classes that needed deallocation of heap data members.

   One point of interest was the private resize array method that was defined for objPosArrayList, as this was a new method, but it makes sense to implement it for better efficiency. Also, it's parameter newCapacity makes sense as you would have to pass a new size or a constant unit size to expand the list by, but I wonder if this is necessary as many dynamic arrays usually just double the size to expand.

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.

   Starting off, in the initialization I notice that the playerPosition and playerBody are two different elements and are being passed to generateFood to be blocked off. This could be simplified to be just the list of snake body objects, where the player's position is the head in that list. Also, the whole if statement to check if the list exists to get the head element is unnecessary since this is in the initialization routine and so we are adding the first element to start with.

   A very important point is the initialization of the GameMechs object. In project.cpp, it is called just fine with the board dimensions which uses the custom constructor. However, if we go look at the implementation of the GameMechs class, we see that the default constructor with no params is not even implemented which can be dangerous and prone to error.

   In the run logic routine, the player collision detection with the food object is written out here which is fine, and makes sense, however instead of passing `!false` for setHasEatenFood, this could be simplified to `true`.

The remaining functions are all implemented correctly and understandably. I am able to read the code and understand exactly what is happening – it is not obscure.

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.

   The pros of C++ OOD approach in the project are reusability of certain features. Features such as updating snake head can easily be implemented in multiple places by using a class method. Furthermore, the OOD approach encourages organization within code. The OOD forces programmers to be intentional with what methods belongs to a certain class and how different features interact with each other through different classes. However, this aspect of OOD design can also be a con as a programmer can easily forget where an implementation of a feature is, provided that there are many class header and implementation files. Furthermore, with poor documentation, poor naming conventions, and code modularization, finding an implementation of a feature will be much more difficult.

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.

   For the most part, the code offers sufficient comments as the comments have helped explained segments of code, and the naming convention of certain functions and variables has made their purpose within the code identifiable. However, more comments could have been added to help others understand code functionality more efficiently.

   Specifically, in Project.cpp at lines 130 to 140, it would have been helpful if a comment was added to explain the general purpose of that for loop as there are many different methods and logic to print out the gameboard.

   Furthermore, in Player.cpp at lines 107-122, comments could have been made to explain the implementation of moving the snake head element, as there were many function calls incorporated within calculating the new position of the snake head, making it slightly difficult to understand the movePlayer implementation.

   In GameMechs.cpp there is quite a lack of comments, and it would be nice to have some more understanding on what is happening and how the code works, whether that be in the generateFood() or even the notation used for implementing the custom constructor.

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.

   Overall, the readability is pretty good, with nice indentation and nicely spaced-out lines and chunks separating concerns. All expressions are nicely formatted with spaces for readability. However, there are a few improvements and comments that can be made.

   Firstly, in Player.cpp in updatePlayerDir(), it is well thought out to include both uppercase and lowercase detection for input however, this can be simplified to have both cases right after each other which acts as an or statement since the cases fall onto each other and break at the end.

   Also in Player.cpp, the movePlayer() function is a bit hard to read and has unnecessarily long lines for updating the coordinates. It is also a bit hard to see/understand the modulo operator in the expression they wrote. Separating the update and the wraparound would be beneficial. Not to mention the fact that getBoardSizeX() and getBoardSizeY() is being called multiple times here despite the fact that they do not change. These can be stored to a local variable for readability and efficiency.

   In the current implementation of the GameMechs custom constructor, the notation used is hard to read since the line is very lengthy and it does not follow standards.

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 runs smoothly as the snake grows when it consumes food. However, there's a small bug with an edge case.

   When the snake has only 4 characters (size of 4), when moving the snake in the smallest square possible, which is a 2x2 square by pressing WASD immediately one after the other, no collision should occur as there are 4 characters that occupy all 4 spaces in a 2x2 square. However, a collision occurs.

   The possible root cause of this bug is within the implementation of self collision. When checking self collision, the tail isn't removed. Thus, there is technically 5 elements within the snake, which would lead to a self collision as 5 elements doesn't fit within a 2x2 square. Thus, to fix this bug, self collision should only be checked after the new head is inserted and tail is removed.

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 has 0 bytes of memory leak as all heap members has been correctly deallocated from the heap through the CleanUp() function of Project.cpp and within the destructor methods of classes that contain data members allocated on the heap.

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.

   The timeline of the project was constructed well and using GitHub for collaboration helped us not only efficiently and safely contribute to code development but also taught real life industrial skills with Git to effectively collaborate with other people. Furthermore, working on the project through GitHub provided the ability to look into past iterations of the project and edits made by the other person through effective commit messages, making it easier to understand what changes were made.

   The iterative workflow structured the content of the project very nicely and provided for a safe, tested, and understandable approach to developing the project. Classes and methods being implemented and tested separately first definitely helped, making the object interaction in the main logic much easier and more intuitive.

   One aspect that could possibly be improved is the separation of work between team members since it is quite difficult to both be implementing the same features when both students have different schedules.