COMPENG 2SH4 Project – Peer Evaluation

Your Team Members: Leandra Andrews Aditi Dhaka

Team Members Evaluated: Prawin Premachandran Aun Maqsood

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.

Positives:

- members are initialized in the correct locations based on their use (public, private)
- correct use of the enumeration which holds the player direction, this is a way to make the code more readable
- the variable and function names are consistent throughout and easy to decipher the functions purposes

Negatives:

- the member initialization int arrayCapacity=200 in ObjPosArrayList is redundant as there is already #define ARRAY_MAX_CAP 200 at the top of the header file
- Player and Game Mechanics header files both include the same unused variable, int speed_adjustment

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.

Positives:

- Code is well commented throughout the main program to allow for a better understanding of the thought process
- Logic is correct for the printing of the symbol and is implemented with good readability.
- Does allocate space on the heap for their instances of game mechanics and player

Negatives:

- Objects are initialized in global scope as opposed to in the initialize routine
- Exit flag is initialized twice in global scope and initialize routine
- Object are initialized and print statements are added to the main function as opposed to within the initialize or draw routine
- Input variable is created in GetInput() even though it is never used
- The height and width of the board are defined at the top of the file,

which is unnecessary, should be using appropriate getter functions
- hasRun variable is initialized to false twice in DrawScreen(), unnecessary.
- No destructors in cleanup function

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 of the OOD approach:**
        - Easier to keep data 'hidden' (private and public)
        - Modularity: Separate files for each object that perform specific functions related to that object; easy to navigate and correct issues if a certain object is not behaving correctly
        - Reusability: Easily reuse and access code as many times as we need without the need for constant modifications
        - Minimized lined of code in the main project file
            - Fewer global variables
            - Easier to keep track of process, and follow along with what the code is doing
        - Overloading: Functions can have the same name but perform different tasks based on its parameters

    - **Cons of the OOD approach:**
        - Greater number of items stored on the heap when compared to PPA3
        - Updates to a class file may require updates to the header file, which is a step that is easy to forget
        - Multiple files for multiple classes

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.

    - Code does offer sufficient comments in certain files such as the Player Class, Project Class, and the header files. However, classes such as **ObjPosArrayList** would benefit from further commenting as the thought process behind certain functions and design features is unclear.
        - Within the default constructor it is unclear as to why sizeList is set to 2
        - Missing comments for function ObjPosArrayList(const ObjPosArrayList &arrayVar). Would improve code understanding if more comments were added to explain the purpose of this constructor, and what the code is doing within it.

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.

   [Additional notes]

   GameMechs:

   - Each function is easily distinguishable due to proper formatting and indentation, making it easy to follow.
   - However, the indentation and formatting are not uniform throughout the entire file but does not impact the readability of the code
     - Formatting changes for function getScore() and incrementScore()

   Player:

   - Indentation is uniform throughout code and readability for the most part of code is good
   - Minimal spaces between different lines/blocks of code make it slightly hard to follow. This issue is exclusive to movePlayer() function
     - There could have been spaces between each case, as well as additional spacing within each individual if statement to easily distinguish between normal direction movement, and the snake body movement
   - "#define DELAY_CONST 1000000" on top of file is unnecessary

   ObjPosArrayList:

   - Code is easy to follow, however unnecessary spacing between code and end brackets in multiple functions. This does not impact readability, but would look aesthetically better is the additional spaces were removed
   - Included "**ObjPosArrayList.h"** on top twice, remove one of them

   Project:

   - Each individual function within the project file is easy to follow due to the uniform indentation, and sensible use of spaces.

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)

i. Player is initialized outside of the game board instead of in the middle as recommended
   - **Issue:** starting player coordinates are never initialized and set
   - **to fix:** within the constructor in player.cpp initialize the player in the middle of the game board using the necessary setter function and getter functions

ii. Player length also begins at 2 [i.e. **] as opposed to 1 [i.e. *]
   - **Issue:** the player list in ObjPosArrayList is currently initialized to 2, thus the games begins with a snake of length 2
   - **to fix:** Inside ObjPosArrayList.cpp modify the constructor and change sizeList to equal 1

iii. The food is always in the middle of the game board at the start of a new game. Positions of new generated food are not randomized either
   - **Issue:** Missing srand(time(NULL))) in generateFood(). Without this the results generated by rand() are the same each time the game is run
   - **to fix:** add srand(time(NULL))) to the beginning of the do-while loop in the generateFood() function

iv. Movements are very fast, even when the delay constant is changed the speed of movement does not change
   - **to fix:** must include an exit command if the board has been printed and the objects have been printed, in the DrawScreen() function there is a hasRun variable that holds the Boolean which checks if each element on game board has been printed (except spaces). However, there is no break or continue commands for when the hasRun variable is set to true. Therefore, the board is repeatedly printed, causing the screen to move very fast.

v. The snake freezes in its place if keys besides the four movements keys [w a s d] are pressed, but resumes play as soon as one of the four keys is pressed
   - **Issue:** when a non movement associated key is pressed, the switch case in the updatePlayerDir() refers to the default case which sets the players direction set to INITIAL. Since there is no movement defined for INITIAL in the movePlayer() function, the player simply stays stationary on the screen.
   - **to fix:** in the updatePlayerDir() function, remove “myDir = INITIAL” from the default switch case

vi. There is no custom message dedicated specifically for the instance the snake collides with itself
   - **to fix:** When self collision occurs, the lose flag is set to be true. Set up an if statement in DrawScreen() and check if lose flag is true. If it is true, print out a statement to notify the user the game has ended due to self collision

vii. Fix implementation of self collision
   - **to fix:** within the for loop, the value of i should be initialized to 1 since we are looking to see if the head of the snake (index 0) collides with any parts of the body, and the body begins at index 1

viii. Fix implementation of generate food

- **to fix:** The generate food function should take in the playerPosList as a parameter to ensure no food is generated under the snake's body

ix. Game crashes without even starting at certain points in time due to the instance of "bad_alloc" thrown by C

```
PS C:\COE2SH4\ProjectEval\2sh4-project-prawin-premachandran-and-aun-maqsood> ./Project.exe
terminate called after throwing an instance of 'std::bad_alloc'
  what():  std::bad_alloc
PS C:\COE2SH4\ProjectEval\2sh4-project-prawin-premachandran-and-aun-maqsood>
```

- **Issue:** unable to allocate enough memory due to insufficient memory space
- **to fix:** ensure all items created on the heap have been deallocated within each class and the project file using destructors and "delete"

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.

```
PS C:\COE2SH4\ProjectEval\2sh4-project-prawin-premachandran-and-aun-maqsood> ~~Dr.M~~
~~Dr.M~~ ERRORS FOUND:
~~Dr.M~~       0 unique,     0 total unaddressable access(es)
~~Dr.M~~      13 unique,   151 total uninitialized access(es)
~~Dr.M~~       0 unique,     0 total invalid heap argument(s)
~~Dr.M~~       0 unique,     0 total GDI usage error(s)
~~Dr.M~~       0 unique,     0 total handle leak(s)
~~Dr.M~~       0 unique,     0 total warning(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of leak(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of possible leak(s)
~~Dr.M~~ ERRORS IGNORED:
```

No memory leakage.

Note: This is odd as no heap data is deleted in the clean up routine of the project file, even though items are created on the heap. Moreover, playerPosList created on the heap in the Player class file is also not deleted.

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.

Leandra: Throughout the project effective communication was crucial and it was one of our strengths. We had clear and frequent meet ups in person so that we could work together on the project and ensure that we were on the right track. Another tool we used was well documented code, so that the other partner could have a clear understanding if they did not write that section of code. What was difficult

was the inability to work on the code editing at the same time and therefore having to miss out on certain knowledge sharing between members. However, this also created a very collaborative environment open to constructive feedback on both ends.

Aditi: The overall experience during the course of the project was positive. We regularly communicated any questions or concerns through teams to facilitate remote communication. Additionally, multiple in-person meetings were also scheduled to collaborate on any obstacles we faced such as setting up iteration 1, displaying the snake body and more. It was easy to work with my partner as she actively listened to any ideas, or concerns I had. The only issue we encountered during the project was working with GitHub's push and pull features. However, this issue was easily solved by assigning each team member specific tasks and waiting until the person was done with their part or working together on one computer to avoid conflicting changes.