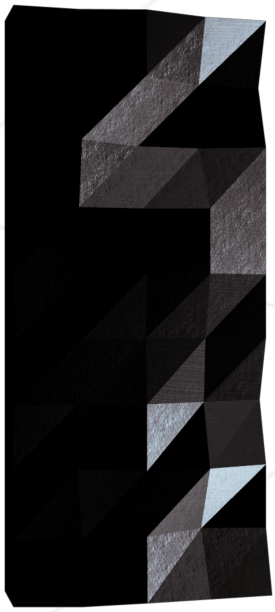

Summer 2022 Review Campaign

Code Refactoring

Summary: Time to clean up some messy code!

Contents

Chapter I

Foreword

Eminem - Lose Yourself
His palms are sweaty, knees weak, arms are heavy.
There's vomit on his sweater already, mom's spaghetti.

What makes good code? Bad code? Spaghetti code?
In this project, you will learn about the principles of clean code.

Chapter II

Instructions

- Chapter 3 contains the base requirements for the code.
- You will be primarily evaluated on chapter 4: Refactor.
- However, in order to be evaluated on chapter 4, the program must be functional!
- If your program does not pass the criteria from chapter 3 you will receive a 0.
- Some aspects of your evaluation will be subjective. Be prepared to defend your code!
- You are not required to abide by the norminette for this project.

Chapter III

Requirements

Program name	`tetris`
Turn in files	`Makefile and all the necessary files`
Makefile	`Yes`
Arguments	
External functs.	
Libft authorized	
Description	`Write a program that draws and print the biggest possible square in the given area`

- The biggest square :
  - Running "make" produces an executable called "tetris".
  - Running "./tetris" clears the terminal, and draws a tetris board at the top of the screen.
  - The board is continuously redrawn at an interval, updating the state of the board with each frame.
  - Tetris pieces spawn at the top of the board, and move down with each frame, until reach the bottom of the board, or another piece.
  - The falling piece can be moved laterally with the 'A' and 'D', and rotated clockwise or counter-clockwise with 'W' and 'S' respectively.
  - If a row is completely filled the following occurs:
    - All blocks within the completed row disappear.
    - All blocks above are moved down a row.
    - The score is updated to reflect the number of points gained. (points = 100 * number of blocks removed)
  - If a block exceeds the top of the board, the game is over, and the program exits, returning to the previous terminal and printing the final state of the board.

- Here's an example of how it should a board should look:

```
42 Tetris
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. # . . . . .
. # # # . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

Score: 0
```

It is a square indeed. Even though it might not look like it visually.

Chapter IV

Refactor

You will be provided a starting code base from which to improve.
Watch the following video, and refactor the code base to reflect the principles covered.
https://www.youtube.com/watch?v=BVwxan6WGpI&ab_channel=CodamCodingCollege

- Readability
  - Clear intent
  - Expressive and meaningful names
  - Simple and straighforward logic
  - Helpful comments
- Redundancy
  - No code duplication
- Scalability/modularity
  - Easily understood and modified
  - Easy to maintain
  - Easy to extend
- Organization
  - Intuitive file structure
  - Reasonable function size
  - Doesn't surprise the reader

To be evaluated on this chapter, your code must work as defined in chapter 3.

The above principles will be evaluated looking at primarily looking at the following:

- Naming
- Functions
- Comments
- Implementation patterns

If your code doesn't require a particular concept, don't add code to fulfil it. You will be reviewed on how often your code violates these principles, not how often it complies with them.