OOD Explained: Board Generation and Turns

Board Generation

Board generation will begin with the UI class, calling the proper constructor based on the number of arguments the UI was given. If the number of tanks was passed, it will check to see if the number of tanks is greater than 25. If it is, the UI will prompt the user to enter in a new number of tanks, because it is impossible to fit more than 25 tetrominos in a 10x10 board.

When the Board constructor is first called, the game board will be generated by first filling up our 2D Array of Unit objects with unoccupied, “fog” objects. Then the Tank constructor is called, which in turn calls the Tetromino constructor. It will randomly place a starting block on the board that will be built into a tetromino. Initially using only the four units up, down, left, or right of the starting unit, one unit from a list of possible units will be added to the tetromino, and the possible units that this new block offers will be added to the array of possible units the tetromino can be built from. This process is repeated until a proper tetromino of size 4 is created. Once the size of this tetromino is 4, the constructor will stop and this tank will be added to the Board. This process will be repeated n times, for the number of tanks that will be in the game.

At any point, if the Tank/Tetromino constructor cannot be completed because it has run out of space, the program will stop and a message will be printed stating it was unable to randomly place all n number of tanks on the board due to running out of space.

The Tetromino constructor should be able to maximize the number of tanks that can be placed on the board because of its dynamic nature. By only creating a tetromino based on the available, unoccupied spaces, it will fill up most “holes” that are created from the Tanks that were created early.

Turns

Every turn, the following operations will occur:

1. User is prompted to make a move by the UI.
2. User inputs the coordinate of the Unit they wish to fire at, and the Board will pass that coordinate to the Fortress.
3. reveal() that unit, which will make its contents visible to the player if it was not already visible.
    1. If that unit was also occupied and has not been fired at before, then the tank whose unit was fired at will take damage, decrementing its health and lowering the damage it deals.
        1. If that tank has 0 health after being attacked, it will be destroyed and the number of tanks alive will be decremented.
    2. If it was not occupied or it has already been shot at, the appropriate message will be displayed to the user via the UI, telling them that they either missed or fired at a coordinate they have already fired at before.
4. Board will check to see if all tanks have been destroyed. If so, then the player has won and the game will end after the end screen is outputted by the UI.
5. The tanks that are alive will all fire at the fortress, taking damage based on the damage that each tank is able to deal.
    1. If the health of the fortress hits 0, the player has lost and the game will end after the end screen is outputted by the UI.
6. UI will print the state of the board will be printed along with the actions that occurred that turn.
7. User is brought back to 1.