Andy Wu - 301308902
Alvin Ho - 301311102

Object Oriented Design Explanation

Action: Calculating how many pieces can be added to the current board (getMaxTanksCanAdd())

Precondition: Only used during board generation and user parameter for tanks is valid. N <= 25.

Algorithm: Our algorithm will count the number of pieces that can fit on the current board. This helps ensure when we are adding pieces that we have space still.

1. The algorithm goes through the board and checking each tile if it is empty and has not been checked.
2. When it finds a tile that fits the requirements, it then calls the flood fill method with the location to find connected empty spaces and changes those squares into checked.
3. It then returns how many empty spaces it found and stores it into a list of region sizes.
4. This keeps going until the entire board is checked.
5. It then adds each of the regions/4 truncated and returns that number.

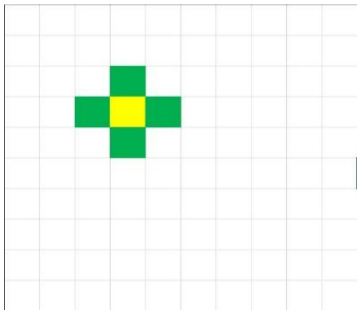
Action: The placement of tanks when the program starts

Precondition: User parameter for tanks is valid. N <= 25.

Algorithm: Our algorithm will try to address the issue of random generation being unable to create a possible board state as N increases. How will our algorithm be able to do this?

1. In this example we will be calling creating a board with 25 tanks. The algorithm gets used in the construction of the board.

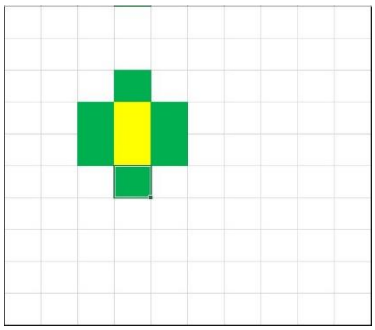
2. We called a createTank() function to pick a random tile to start with, let's call that Tank A. Then we initialize that tile to be a tank tile from the enumeration tile class. Note that tank isn't created until 4 tiles are chosen.



3. We then find all adjacent tiles to that random tile. If they are a possible (meaning they aren't out of bounds or conflict with another tank), we add these tiles into a list of possible tiles (this is list is a temporary list).

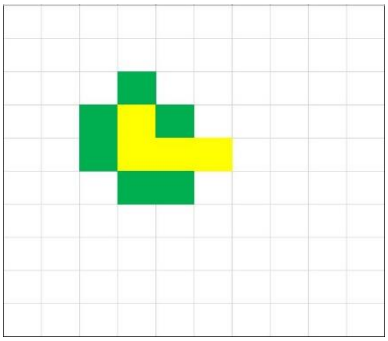
Andy Wu - 301308902
Alvin Ho - 301311102

4. The next step is to pick a random tile from the list of possibilities and remove the picked tile from the list of possibilities. Since we now picked a new tile we must find all adjacent tiles to that new pick tile. Add these to the list of possible tiles.

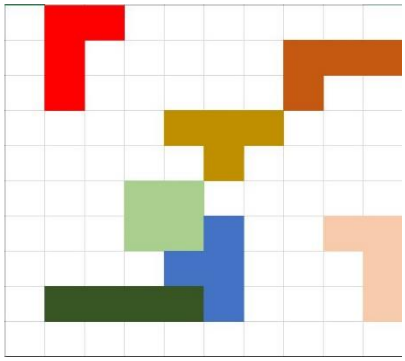


5. Repeat 3. for the third tile.

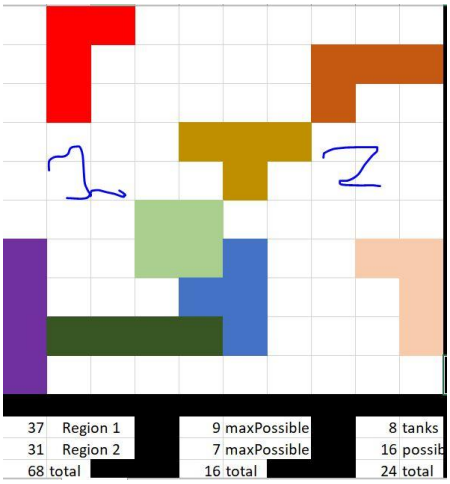
6. However, at the fourth tile we use the getMaxTanksCanAdd() method to check the number of tanks we can add on to the board. In this case, we get 24, which mean this piece is fine to be added.



7. Let's also observe the case where we have we have multiple flood fill regions. How do we resolve that issue where the chosen 4th tile gives us an impossible state? Here is an example of how this occurs. Let's add a tank to the given picture.

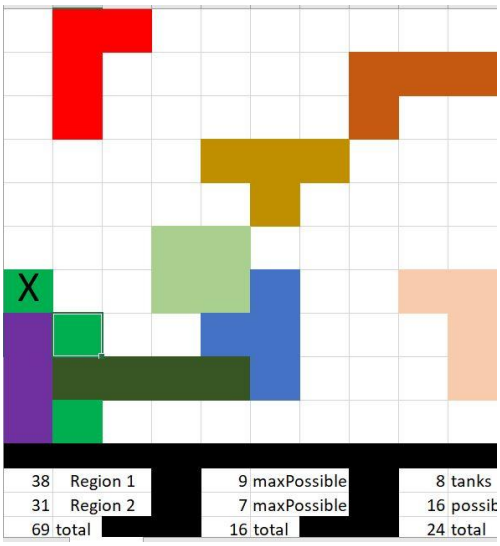


8. Upon picking a 4th tile we add the getMaxTanksCanAdd() with the pieces we have added. As you may notice this gets us 24 which is less than the goal of 25. Therefore, we must edit this piece so that we are able to obtain 25 total tanks.

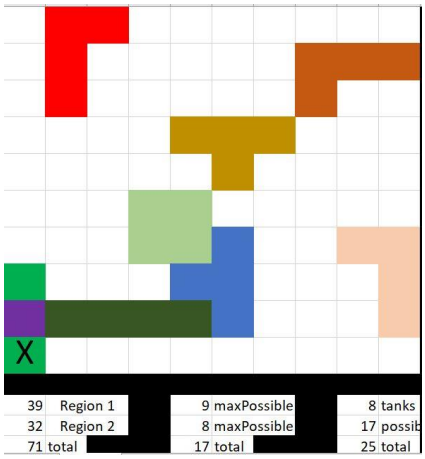


Andy Wu - 301308902
Alvin Ho - 301311102

9. We remove that 4th tile and we add the getMaxTanksCanAdd() with the pieces we have added. Because this still gets us 24, we remove the 3rd tile added and so forth.



10. Eventually we get a point we will get to a point where calling getMaxTanksCanAdd() finally gets us 25 tanks. Turns out it removes everything except the 1st piece. We check for possibilities again however, the 2nd tile we removed is no longer a viable option because it was the source of the problem. We now pick the other tiles and it should work with at least one of the possibilities.



11. Now we arrive at this possible configuration of a tank and during the process of picking a piece we are constantly calling getMaxTanksCanAdd(). By theory we should be able to generate such a board where it can fit all 25 tanks.