Greedy Algorithms: Flood It

C343 — Assignment 7

Due: March 3rd, 2015 at 11pm

1 Preliminaries

- Please form teams of 2-3 for this assignment.
- Please intall `pygame`. This is not necessarily easy!
- Download the file `a7.py` from Oncourse.
- If you haven't played `Flood It` before, please play several rounds. It is important to get the right intuition before trying to implement anything. There is one online version at `http://floodit.appspot.com`.

2 Flood It

You are given a file `a7.py` with the following components:

- a class `C` to represent individual colors;
- a class `Colors` to represent a collection of colors to use in the game;
- a class `Tile` to represent one of the tiles of the game; each tile has a position (specified using `x` and `y` coordinates), a color, and a size; a tile knows how to draw itself on a given surface;
- a class `Game` which is the main driver that draws the board and goes into a loop processing mouse clicks;
- a class `Board` which is incomplete; this is the only class you will modify. There is a block of four methods called `move`, `greedy1`, `greedy2`, and `greedy3` that you should implement.

Your implementation of the four required methods **must** scale gracefully as the size of the board increases! The main data structure that the four methods manipulate is the *current flooded region*. This is the region that starts from the top-left corner and recursively includes all neighbors of the same color. A tile is the neighbor of another tile if it is to the east, west, north, or south of that tile. (An interesting variation of the game could also include the neighbors along the diagonals but that would be a different game.) In more detail, the methods to implement are:

- `move`: the method takes a new color for the top-left tile. All the tiles in the current flooded region are updated with the new color. Additionally, all adjacent regions of the same color are absorbed in the current flooded region.
- `greedy1`, `greedy2`, and `greedy3` which take no arguments and return the "best" color for the next move. Each method uses a different greedy algorithm to calculate the best move. The first method maximizes the number of tiles in the current flooded region. The second method maximizes the perimeter of the current flooded region. The third method chooses your own strategy.

3 Deliverables

Each group should submit one instance of `a7.py`. Your submission should include the code for the four required methods. Add a paragraph describing your strategy and your assessment of the best strategy for playing the game.