CS152L – Assignment 4

Summer 2008 – Matthew Barrick
Due 07/31/2008 11:59 PM

Overview:

You will be implementing a cellular automaton similar to Conway's Game of Life. It should be noted that I didn't come up with this automaton; it's a common one in artificial life and scientific modeling classes. The model is based around the game of Rock-Paper-Scissors (RPS).

Traditionally this game is played between two people. Each person secretly selects one of the three options, rock, paper, and scissors. Both people reveal their selection simultaneously, and the winner is determined by the following (questionably rational rules):

- Paper covers rocks.
- Rock smashes scissors.
- Scissors cuts paper.

If both players select the same option, it is a tie.

For the cellular automata model you will implement, you will have a 255x255 cell grid of players. Each player will have a "strategy", that is it will have a state (rock, paper, or scissors) which it will always play.

On each time step, it will play its neighbors (will we use von Neumann neighbors, that is the four cells north, south, east, and west, for this assignment because a player having eight arms to play RPS with would be, as it has been pointed out, ridiculous). So it will play RPS with its four neighbors. If it beats or ties with all its neighbors, the cell will assume that its strategy is good and stick with it for the next time step. If any one of its neighbors beat it, it will change its strategy to that which defeated it.

For example, if a cell beats two of its neighbors, ties with one, and is beaten by the last. It will use the option (rock, paper, or scissors) that beat it for the next time step. If none of its neighbors beat it, it will stay with the same strategy.

Visualization:

You will provide two ways to visualize the model as it evolves. The first will be an animation of the cell grid where each cell is represented by 3x3 pixel box (I suggest that you use `Graphics2D.fillRect()`) in an image that is 765x765. You may use either the `Animator` code from the book or the method used in the class `Koch` that is posted on the website (of the

two, the latter is recommended). You need to show at least the first 100 time steps with a reasonable delay between each step (experiment to find one that is good). Save the `Picture` at the beginning and end of this visualization to an image file (you'll need them, see below). You should use red for rock, blue for paper, and green for scissors.

The second visualization will be a graph of the number of rock, paper, and scissors cellse at each time step over 10,000 time steps (use a log-scale for the time-step axis of the graph if appropriate). Note that this means you should keep a count of the number of each, and write the three counts to a file at each time step. The file should look something like:

```
4320        13004       42019       10002
```

for time step 4320.

Initial States:

You will have three classes `RandomInitialState`, `CheckeredState`, and `BarState`, each of which will have only a `main` method that will start the model with a different initial state. The logic of the model should be entirely within the classes `RPSGame` and `RPSPlayer`.

`RandomInitialState` will select the initial state of each cell/player at random.

`CheckeredState` will checker the options in the order RPS, so the upper left will look like this:

```
RPSRPSRPS
PSRPSRPSR
SRPSRPSRP
```

`BarState` will alternate vertical bars of the same state as shown below:

```
RPSRPSRPS
RPSRPSRPS
RPSRPSRPS
```

Implementation Requirements:

Your program should consist of at least the source files:

`RandomInitialState`	-	main method that starts `RPSGame` with proper state
`CheckeredState`	-	main method that starts `RPSGame` with proper state
`BarState`	-	main method that starts `RPSGame` with proper state
`RPSGame`	-	contain the logic for running the model (similar to the class `Conway` in Assignment #3)
`RPSPlayer`	-	Contain the state of each RPS player/cell automaton (similar to the class `Cell` in Assignment #3)

You code *must* use a **switch** statement somewhere (There are a number of place where one should be suitable, so this should not be hard).

Hand-in Requirements

Of course, you should hand in the `.java` files for the classes specified above. Additionally, you must submit a write-up of your assignment as described below. The write-up should preferably be in an open, cross-platform format like PDF or postscript. OpenOffice.org files, or if you must Microsoft Office Documents are fine if that is only option easily available to you. (I'll deal with whatever you give me, but a PDF would be easiest).

The write-up should include the pictures of initial and final (after 100 steps final that is) state for each of the initial states. Also the graph of the rock/paper/scissor populations described above. Additionally for each, a short (~100 word) description of what the automata did. (What shapes/behavior emerged, did it reach an oscillating or convergent state, etc). Be sure to have your name, the date, and the class on the top of this write-up.