Ray Alfano

7/1/12

Biologically Inspired Computing

Assignment #4

Instructions for simulator:

This application is written in Java. Ideally, import the project Alife into Eclipse and utilize the package structure contained in Alife > src > fourLayerGrid. After much coding and recoding I narrowed my implementation to two classes and a properties file.

The main class of the simulator is located in EcoAlifeSim.java. The properties file containing the configuration data is called localsettings.properties and should be located in the same folder. The majority of the simulation code is contained in the file AlifeGrid.java. The file EcoAlifeSim.java functions primarily to read in the data from the configuration file and load the simulation.

Until recently, there was another java file included that would load a JFrame and show the color states of each grid sector using color data at the end of the simulation. However, I was unable to make my implementation work on my Mac platform as it did on my primary Windows platform and therefore resorted to console output of my data. According to my research online there is an issue with defining background and foreground values for JButtons on Mac OSX.

Notes on data structure

My initial implementation expressed all values in terms of color, so my console-based implementation maintains the color representations of the four required sectors. Each grid sector (contained at position [n][m]) is expressed as a color channel.

* R, the red channel, corresponds to an integer 0-255. This represents the “fire” value in the local area. 0 is no fire, 255 is a new and active blaze.
* G, the green channel, corresponds to an integer 0-255. This represents the vegetation value in the local area. 0 is none, 255 is max.
* B, the blue channel, corresponds to an integer 0-255. This represents water in the local area. 0 is no water, 255 is maximum water.
* A, the alpha channel, corresponds to an integer 0-255; this represents the clouds in the area. 0 is no cloud cover, 255 is maximum cloud coverage.

Parameters:

Please change the parameters in localsettings.properties. The parameters are (in order):

Global parameters:

N\_VALUE and M\_VALUE are the N\*M grid size values required in part 1 of our implementation structures. They should be integers. By default they are each 3.

TURN\_LIMIT is the amount of steps to be shown.

Water-specific parameters:

WATER\_LEVEL is the initial value for water in each grid cell

EVAPORATION\_RATE is the loss of water to clouds per turn

Fire-specific parameters:

LIGHTNING\_PROBABILITY is the probability in any given turn of a lightning strike in the local cell. This really refers to the probability of a fire starting at all, since it is independent of rain co-occurring. By default it is a 7% chance of fire if the area is dry enough, expressed as 0.07

BURN\_RATE\_LOCAL is both the rate at which a fire destroys vegetation and the rate at which a fire burns itself out. If all local vegetation is destroyed (G=0) the fire goes out automatically.

RAIN\_FIRE\_SUPPRESSION is an additional acceleration value to put out fires if it is also raining in the local area.

Cloud-specific parameters:

CLOUD\_RAIN\_THRESHOLD is a percentage value applied to the maximum possible cloud coverage at which rain should begin. That is to say, a threshold of 0.6 would mean 0.6\*255=153. In that case, a cloud value (A>153) will cause rain to begin. A cloud value under that threshold will stop the rain event on the next turn.

CLOUD\_ABSORPTION\_LOCAL is the rate at which clouds absorb evaporated water from the local water supply. By default this is 0.2 such that up to 8 neighboring squares get 0.1 of available evaporated water each.

CLOUD\_ABSORPTION\_NEIGHBOR is the rate at which clouds in neighboring sectors absorb water from the local water supply. By default this is 0.1.

RAIN\_INTENSITY is the amount of water lost by a cloud during a rain event. For a value of 50, every turn that the cloud level A is above the rain threshold will cause the cloud to lose 50 water to the environment.

NEIGHBOR\_WATER\_CONTRIBUTION is the amount of water drawn from neighboring grid sectors (that have water) for local vegetation. The value adds directly to the vegetation locally.

LOCAL\_WATER\_CONTRIBUTION is the amount of water drawn from the local water supply every turn to create more vegetation.

BURN\_THRESHOLD\_LIGHTNING is the level where vegetation begins to burn in a lightning strike. If the vegetation level is above this threshold, a lightning strike will not cause a fire. I am using vegetation as the determining value for fire starting because ambient water does not immediately cause a dry area to flourish in real life. An area with abundant water and dry plants should be fire-prone to be realistic, as in a prairie environment.

FIRE\_SPREAD\_THRESHOLD is the vegetation level at which a neighboring sector’s active fire will spread locally. This allows for fires to spread in dry neighboring areas.

Simulation explanation:

All output is directly to console. The first section of the output displays all parameter values. The second section is the simulation itself and shows the following information in order:

* Lightning strikes (if they occur) and the sector in which it occurred by it’s (n,m) value. Note that effects on vegetation will appear in the turn after the one immediately following the announcement as it takes time to burn.
* The step number for the environment state output
* The “row” currently being displayed, with cell (0,0) being at the top-left and cell (n,m) at the bottom-right.
* Next to the row, contained in brackets, are the specific grid sectors.
* Each area between brackets “[ ]” is one grid sector. The values for water, vegetation, clouds, and fire are displayed with labels.
* The grid sectors for each row area displayed in order, side by side as a text-based representation the GUI that I re-implemented in the console.
* If rain is active or a fire spreads in a sector, a string is displayed to the right of the associated sector’s bracket to notify of that event.

Implementation structure notes:

Parameters are loaded in the main class, then passed into the primary class, which then creates the initial world state’s water, cloud, vegetation, and fire levels. By default, no fires are active at the start of the simulation, so fire (R) is zero until a lightning strike or a fire spreads locally. Water is the same in each sector initially. Vegetation and clouds are also defined by the initial water level, but clouds benefit from initial evaporation at the start and vegetation starts with groundwater contribution from neighbors.

In the initial portion of the simulation each of the (n\*m) grid sectors are defined as containing one RGBA Color value. All operations are performed on each individual channel as explained previously. If the cloud (A) value is above the rain threshold value, it is raining in that local area. If the fire (R) value is greater than zero, there is a fire in the local area. Rainfall will drain clouds of water, distribute water to neighbors, and accelerate fire extinguishment. Fires will destroy vegetation at a constant rate and automatically spread to neighboring sectors if their vegetation (G) level is below the threshold, but only if it is NOT raining in those neighboring sectors (for realism purposes).

At the beginning of each turn or step, each individual grid sector applies it’s own internal environmental processes. In order:

* Rain begins if the threshold is met.
* Fires burn vegetation (although at a slower rate if it is also raining).
* Vegetation grows (or does not during a local fire due to natural water conservation processes).
* Vegetation decays towards the local water level if the water level is less than the vegetation level.
* Water evaporates and “feeds” clouds.
* Clouds (if raining) deplete themselves of water, giving back an amount locally according to the threshold value.
* The probability of a lightning strike is then computed; if lightning does strike and the environment is dry enough a fire begins with intensity equal to the amount of local vegetation.

After these local processes are resolved internally, the effects of neighboring sectors are then contributed using the preceding turn’s environmental state values. Since any given sector may have up to 8 neighbors (due to the nature of a grid), the neighboring sectors’ indices are checked for whether or not they are valid locations. For example, (-1,-1) or (n+1, m+1) or (n+1,m) or (-1,0) are not valid locations. Each valid neighboring sector contributes some of its groundwater to the local area, grants some evaporated water, distributes rainfall if applicable, and spreads fires if the threshold is met and it is not currently raining locally.

After the internal processes are completed and the influence of neighbors is taken into account the simulation is updated and the world state reflects both.

Experiment notes:

Included with my submission in the zip file is a folder of text files of my console output with some explanations of what they reflect at the top.

Results for a 2x2 grids (with 4 sectors), were very consistent. Each area affects each other area and as such there were generally common values and trends for each.

Results for larger grids, 3x3 and greater, were much less consistent. Because areas along the grid axes cannot rely on their neighbors as much for vegetation growth, outlying areas were often less stable than central areas with more neighbors. An issue I encountered is that areas with more neighboring sectors can draw more groundwater and are more likely to have rainfall from other areas. Thus, they tend to be more “verdant”. They also become more susceptible to fires depending on the area and the parameters. These results are realistic if we conceptualize the total simulation space as an island or otherwise self-isolated area, as islands often tend to have denser growth towards the center and away from the coastline if standing bodies of water are not often found and heavy rainfall is distributed frequently, such as in the tropics.

Rainfall in areas with fewer neighbors created a loss of water, as the parameters define a situation wherein a fixed percentage of rainfall contributes to local water and a fixed percentage of rainfall contributes to neighbors’ local water. Since, for example, sector (0,0) has 3 neighbors, by default it receives only 20% of it’s own rainfall back and contributes 10% to each theoretical neighbor. That constitutes 50% waste of water, and it’s neighbor’s supply it with commensurately less water than central locations would receive. However, in the island view of this scenario, this is also realistic as coastal rainstorms “wastefully” dump water into the sea.

Generally speaking, the lightning and burn rate probabilities provided for the most consistent results in larger areas, either routinely destroying vegetation entirely or allowing for uncontrolled growth. Changing the evaporation rates past 10 often caused areas to dry out entirely in some simulations. Several other interesting results are that lightning probability does not necessarily allow one to anticipate fires, as in some simulations high probabilities of lightning did not necessarily generate frequent or widespread fires. The burn threshold values are far better predictors of widespread fire damage, and the addition of that local rain can stop fires from spreading also occasionally controls wildfires from getting “out of control” helped provide some balance. The contribution of neighbors was rather profound, and my somewhat simplistic implementation of their effect still consistently showed that the interrelation of different sectors could be hugely influential. Interestingly, in most situations without extremely low burn thresholds, influential values for neighboring sectors provided more overall balance than I hypothesized. I had originally anticipated that neighbors would synergistically contribute towards maximum water values and maximum vegetation, but with relatively realistic-seeming parameters the simulation contained all sectors within the middle 80% of the 0-255 range over time.