Design Document for “Ant Lions”

INFT2100 - 702, PROG2100 - 702

Authors	Ben Allen 1111111, Isaac Benoit 2222222, Jamie Curnew 3333333, Will Matheson 4444444
Initiated	November 28th, 2012
Current Rev.	1.0
Revision Date	December 11th, 2012

Key:

	y →					
x ↓	0,0	0,1	0,2	0,3	…	0,19
	1,0	1,1	1,2	1,3	…	1,19
	2,0	2,1	2,2	2,3	…	2,19
	3,0	3,1	3,2	3,3	…	3,19
	…	…	…	…	…	…
	19,0	19,1	19,2	19,3	…	19,19

	y →		
x ↓	x-1,y-1	x-1,y	x-1,y+1
	x,y-1	**x,y**	x,y+1
	x+1,y-1	x+1,y	x+1,y+1

Program begins. A grid of 20 columns by 20 rows is initialized. (This will be an array of type Organism: `Organism *grid[20`[1]`][20];` held inside a World object.) Each space can be occupied by one ant or one antlion (henceforth ‘lion’), or it can be empty. Inheritance is used: Ants and Lions are both Organisms. The grid does not wrap – the edge of the grid is the edge of the world.

The game begins with a random distribution of 5 lions (shown as ‘X’) and 100 ants (‘O’). The easiest approach is probably to generate a random row number between 0 and the grid height less one, and a random column number between 0 and the grid width less one, and if there is something there already, try a different combination of random numbers, continuing in this manner until all ants and lions are distributed. Note that a true random distribution should have clustering and not be uniformly scattered.

The initial configuration of ants and lions is displayed to the user. We must use color in the program, so ants could be indicated with a green O and lions with a red X.

[1] - These should be constants inside the World header so as to make it easier to run smaller simulations during development and testing.

Every n^2 seconds, a time step is executed. Here is what happens in a time step:

1. Reset "already moved" flags on all critters.
2. All of the lions move (may include eating ants).
3. All of the (remaining) ants move.
4. Lions, eligible and able, breed.
5. Ants, eligible and able, breed.
6. Lions that haven't eaten starve.

Since the array is of the superclass type (Organism), some logic will be required to apply the correct actions to the correct types.

Ant Behaviour	
MOVE	Randomly try to move up, down, left, or right (one space). If the adjacent space in the selected direction is occupied or would be off the edge, the ant stays where it is.
BREED	If the ant has survived for three time steps, it may breed into an adjacent empty space. If no empty cell is available, nothing happens, but the breed counter is reset as if it bred.

Lion Behaviour	
MOVE/EAT	The lion will move to any adjacent cell containing an ant (including diagonals!) and eat the ant. If no ants are available, it will randomly try to move up, down, left, or right (one space) and if the selected direction is occupied by a lion or would be off the edge, the lion stays where it is – same as how the ant moves.
BREED	If the lion has survived for eight time steps, it may breed into an adjacent empty space. If no empty cell is available, nothing happens, and the breed eligibility is maintained for subsequent turns.
STARVE	If the lion has not eaten an ant within three time steps, at the end of the third time step it will die (and disappear from the grid).

The moving and eating and breeding happens on the spot – each critter will only move (or breed) once in a time step (this could be ensured via a flag on each critter object), but as we loop through the array we can have the critter do what it's going to do right then and there (unlike the situation in the "Game of Life" scenario where we needed to have everything happen at once).

The current iteration should be displayed when the display refreshes. At least 100 iterations must be run – programmers responsible for the individual implementations can decide how to implement quitting (perhaps through encouraging the user to press Enter). A successful program will run at least 60% of the time without an "extinction event" (all ants or nothing at all[3]).

[2] - This is up to the programmer. You could even have a prompt asking the user how many seconds.
[3] - The "All lions" lions would eventually starve and the grid would become empty.

Project Closure Report Version Control

Version	Date	Printed By	Change Description
0.1	Nov 28 2012	WM	Initial draft (of “Ant Lion” – “Game of Life” has been deemed a dead end)
0.2	Dec 1 2012	WM	1. Clarified that the lion (when not eating) *tries* to move in a random direction, and will do nothing if moving in that direction is impossible. 2. Logic will be required to apply correct actions to correct types.
0.3	Dec 7 2012	WM	1. Made ants and lions simply breed into “an adjacent” empty space (not necessarily random). The requirements didn’t say this had to be random, and time is curtailing ambition. 2. It certainly *is* possible to end up with an empty grid if the lions “corner” the ants so that none can escape / breed.
0.4	Dec 10 2012	WM	WM: Further to the above, in an attempt to improve the longevity of the lions, I made the spaces to eat ants from (if there’s a choice) and to breed to (if there’s a choice) random, but there was no significant improvement in the execution. The random number generator I am using might not be very good. If I find a good one and it helps, I’ll make a note of it in the design and specify that the eating / breeding spaces should be random.
0.5	Dec 10 2012	WM	Changed “between 0 and 19” to “between 0 and (appropriate dimension) less one”.
1.0	Dec 11 2012	WM	Final submission.