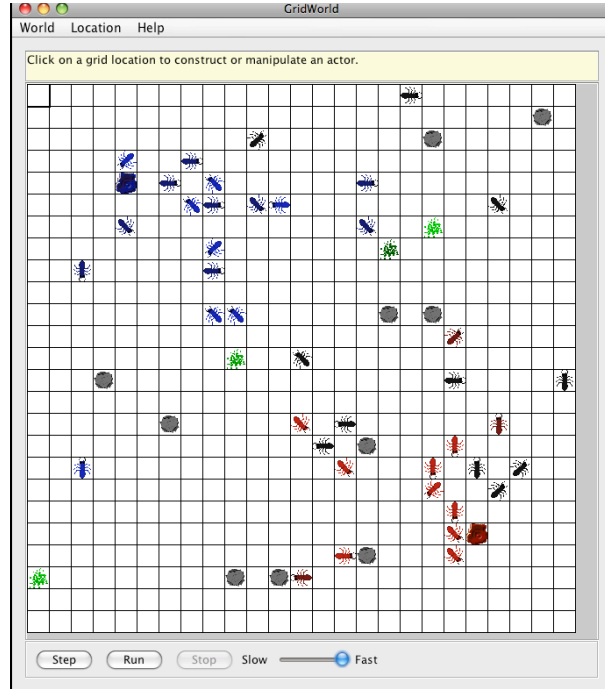


APCS :: Lab 19 - Emergent Behavior of Ants (GridWorld)

Description:

Ants walk away from their home anthill randomly until they find food. When they find food they take a piece and return to their home anthill (they remember its location) to deposit the food. Once an ant knows the location of a food source it can travel back and forth between it and its home anthill until the food is gone. If two ants come into contact, one can communicate the location of food to another. (In reality, Ants leave a trail of pheromones along their path from the anthill to the food. Other ants pick up on the pheromone trail). Over time you'll see emergent behavior from the ants.



From this description you might be able to dive right in and give it a shot. If not here is how I suggest you proceed:

Your world will need 3 kinds of actors:

1. Food
2. AntHill
3. Ant

Step 1: Make an ant with random walk.

An ant should be a critter. Ant's default movement is a "random walk" which is defined as follows:

- An ant always tries to move to one of the 3 locations in front of it.
- It has a 50% chance of going straight forward and a 50% chance of going to the forward-left or forward-right location.
- If any of those three locations is available it should move to one of them.
- If none of the 3 forward locations is available it should move to a random empty neighboring location (default critter behavior).

Test your ant to make sure it does a random walk properly.

Step 2: Make an AntHill

An AntHill is an Actor. Each time it acts it should have a 10% chance of producing a new Ant at a random location adjacent to the AntHill. When an Ant is constructed it should be facing a random direction and its constructor should accept a reference to its home AntHill. An AntHill also stores food units (an int) - we'll get to this later, but add the variable now. And an AntHill can only produce a certain number of total ants - for now, set max ants to 25.

Test this out to make sure that the Ant Hill randomly produces Ants and that there is a max of 25 of them.

Step 3: Make Food

Food is an Actor. It has a particular number of food units. It should have a public method to `takeSomeFood()` which decrements its food units by 1. The Food's `act` method should change the color of the food based on the number of food units remaining (perhaps shifting from green to black?). When there are no more food units left, the Food should remove itself from the Grid.

You can test this by putting an instance of Food into the world and using mouse clicks to take some food. Make sure the food changes color properly and removes itself from the grid.

Step 4: Add to Ant behavior

Ants need to look at their neighboring locations for 3 things: Food, AntHills and other Ants. Here's what to do for each:

4.1 Food:

If one of the neighbors is Food, an Ant should:

1. `takeSomeFood` from it - and record the fact that it "has food"
2. save the Location of the Food so it can come back to it later
3. turn in the direction of its home AntHill.

Then, as long as the ant has a piece of food, it should always set its direction toward its home AntHill (if you do this the random walk will actually get the ant back to its home rather directly. Think about it).

4.2 an AntHill:

Discovering an AntHill in a neighboring location is only important...

IF AND ONLY IF the Ant has food and it's the Ant's home AntHill:

1. deposit the food with the AntHill (add a method to AntHill called depositFood(), which increments the units of food stored by the AntHill.
2. record the fact that it no longer has a piece of food.
3. turn in the direction of the known food source.

4.3 Another Ant:

Discovering another Ant is only important

IF AND ONLY IF the Ant does not currently know of a food source:

1. Ask the neighboring Ant for the location of its known food source - the neighboring Ant might, or might not, have one.
2. If you get a location from the neighboring Ant turn in the direction of the food source.

Step 5: Enhancing the output.

Whatever you do, your runner class should set up a scenario that clearly demonstrates what your Ant simulation can do.

You may find it helpful for debugging (and for figuring out what's going on) to color-code the actors in various ways depending on what's happening. For example, the Food changes from green to black as it depletes. In my example, I had the Ants set their color to the color of their home AntHill if they knew of a food location, brightened that color if they actually had food, and set the color to black otherwise. There are probably more effective ways to color.

You can also get fancier with the ant behavior. In nature, ants can only "remember" locations for a certain amount of time. If an ant has to travel too far from home it might get lost.

AntHills might also produce more ants if enough food is deposited in them to promote reproduction. Similarly, the AntHill (as a community) might consume food at a certain rate, and ants might die if they cannot get enough food.

You could go on dreaming things up for some time.