swarmathon 2

advanced bio-inspired search

# follow the trail

Have you ever seen a solitary ant find a pile of food, only to be followed by a whole line of ants just a few minutes later? Have you ever wondered how they do that? Stigmergy is the answer.

## what is stigmergy?

Stigmergy is communication that occurs through the environment, rather than from person to person. Ants employ stigmergy by laying a chemical pheromone trail that other ants can follow. Laying pheromone allows an ant to signal to other ants where resources are without direct communication. Ants also reinforce existing trails if they are still useful.

*NatGeo*

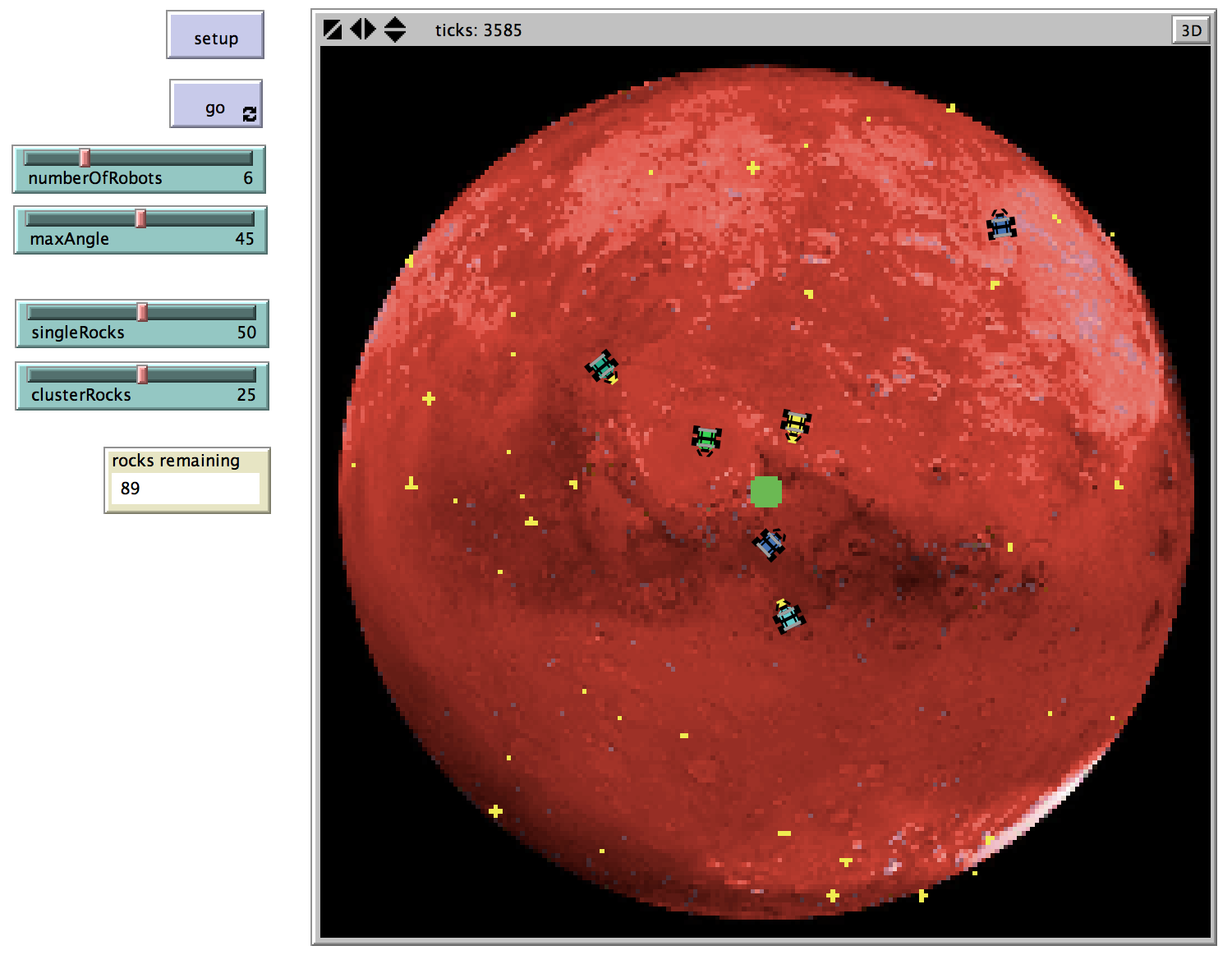
In Swarmathon 2, we will build on some base concepts from Swarmathon 1. In Swarmathon 1, we worked extensively with robots-own variables that represented each robot’s memory and state. The robots used their memory to employ the site fidelity strategy. The go procedure controlled the robots’ behavior based on their current state.

To implement stigmergy, we’ll use both robots-own variables and the go procedure again. But we’ll also need to work with the “ground” in Netlogo—the patches. To this end, patches-own variables and variables that change their value over time are introduced.

# getting started

## file setup

As in Swarmathon 1, we will be using Netlogo base code and a background image.

* Create a folder named *yourlastname \_Swarmathon2.*
* Place the .nlogo file and the .jpg file in your new folder.
* Open the .nlogo file.

Click the setup and go buttons. The robots should search for rocks and return them to the base.

## review swarmathon 1 challenges

Click on the Code Tab. Note that the **End of Section Challenges** from Swarmathon 1 are also completed. Check your answers!

* On the Interface and in the setup procedure: A slider controls the number of robots. ([Sw1] Section 2)
* In the go procedure: The robots return to the base when no rocks remain. ([Sw1] Section 3)

# the trail to success

*Shutterstock*

## what do we need to add?

To implement pheromone trail following and laying in the robots, we’ll need to code the following behaviors:

main agenda

1. Pheromone has a time limit so that trails evaporate.
2. Robots need to know if they are using pheromone.
3. Patches need to know if they have pheromone on them, and how long it has been there.
4. We should create some larger clusters of rocks to test the pheromone’s effectiveness. If our pheromone is working correctly, robots should lay trails from the large cluster and follow trails to the large clusters.
5. Our robot controller statements in the go procedure need to handle the state in which the robots are using pheromone. Note that an additional state, returning?, has been added for you already.
6. The go procedure should also control how pheromone evaporates on the patches.
7. The look-for-rocks procedure will need to be modified. When a robot finds a rock, it should check if there are other rocks in the immediate area and turn on pheromone if there are. By doing this, it will be able to lay a trail back to the base to signal to other robots that there are rocks at the end of the trail.
8. The return-to-base procedure will need to be modified. While a robot who turned on pheromone is returning to the base, it should lay pheromone. Also, if a robot is at the base after dropping off a rock, it should check to see if there are any trails it can follow.
9. State switching by turning on and off the robots-own variables will be managed throughout the program.

## modifying globals & properties and the setup procedure

Let’s begin by tackling agenda items 1 – 4.

agenda 1 – 4

1. Pheromone has a time limit so that trails evaporate.
2. Robots need to know if they are using pheromone.
3. Patches need to know if they have pheromone on them, and how long it has been there.
4. We should create some larger clusters of rocks to test the pheromone’s effectiveness. If our pheromone is working correctly, robots should lay trails from the large cluster and follow trails to the large clusters.

Scroll to the top of the file, where the Globals and Properties section is.