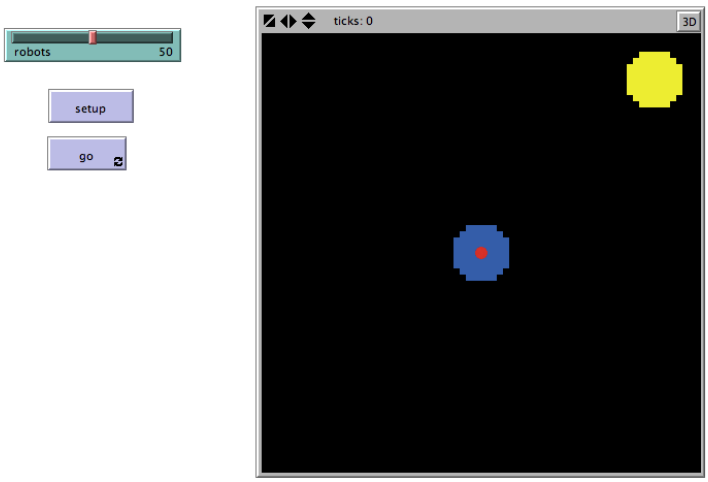
Lunar Explorers

Imagine a spaceship landed on the moon and has the mission to collect a set of rock for lab analysis. The spaceship has marsupial qualities and inside there is a swarm of robot explorers that have to look for rocks and bring them back to the ship.

Challenge nº 1

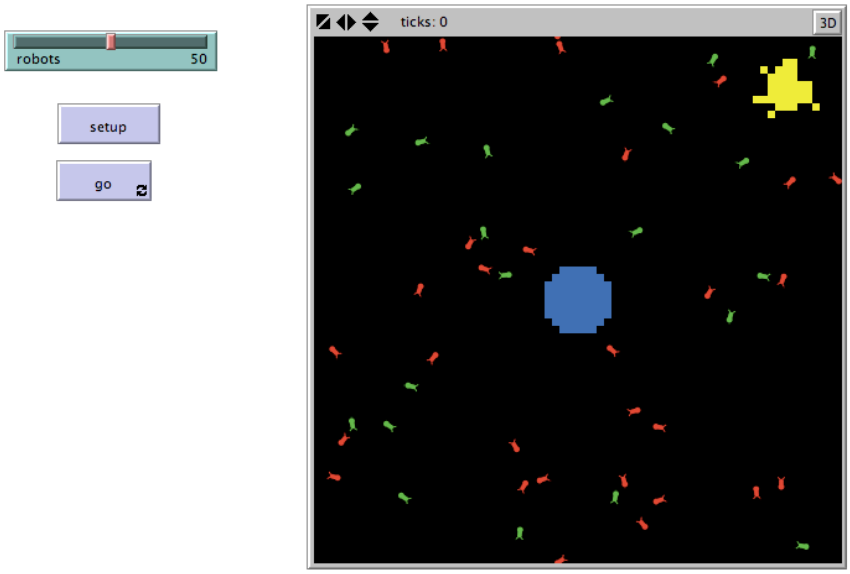
We'll have a circular ship formed by blue patches centered at (0,0) with a radius of 5 and a rock circular spot formed by yellow patches centered in (0.8 * max-pxcor, 0.8 * max-pxcor) with a radius of 5.
We have a "slider" with the global variable *robots*, which indicate the number of robots inside the marsupial ship.
WE wish a size of 2 for turtles, starting all at the ship center. We want also that the robot shape is "bugs".
We will create two buttons: The button *setup* that creates the ship, the rock spot and the explorers; the button *go* (a forever button) that, as the name indicates, executes repeatedly the explorers behavior. Each yellow patch will correspond to a single rock.



The precedent figure illustrates the *interface* we wish and the state of the world after clicking in the *setup* button.

The robotic behavior is the following. The zigzag randomly looking for rocks (they do not have the slightest idea where they are located) e when they find them they collect them and start the way home. They do not know where the ship is located an, thus, they will have to zigzag randomly, looking for the ship. When they arrive to the ship, they will deposit the rock they carry and will go back to the collecting activity, exploring the moon. Each robot can only carry a single rock. The robots that are looking for rocks should be colored red and the robots that are carrying rocks should be colored green. As we can see above, the red center corresponds to the pile of 50 red robots (in the initial situation they are not transporting anything, they have just arrived from the earth, hungry for rocks). We will ignore robot collisions.

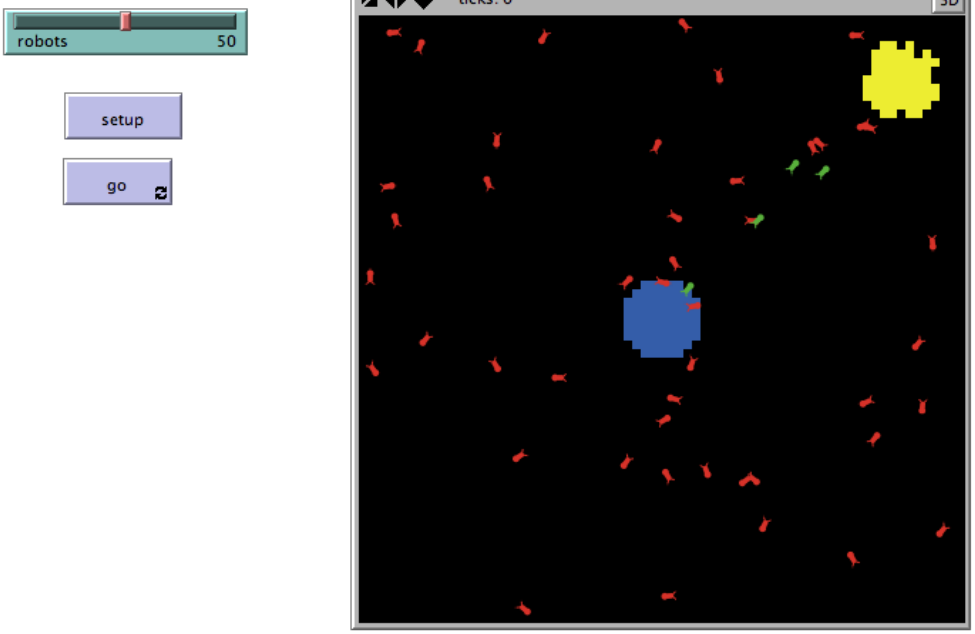
The following snapshot was captured during the exploratory mission. We can see red and green turtles (robots), actively seeking for the rock spot and the ship, respectively, and also the rock spot already "eaten".



Challenge nº 2

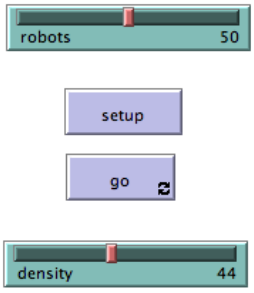
Let's maintain the scenario of Challenge 1 and make a slight modification in the capabilities and in the explorers behavior. Now they are able to orientate themselves toward the ship, so that they will go directly towards it, with no delay

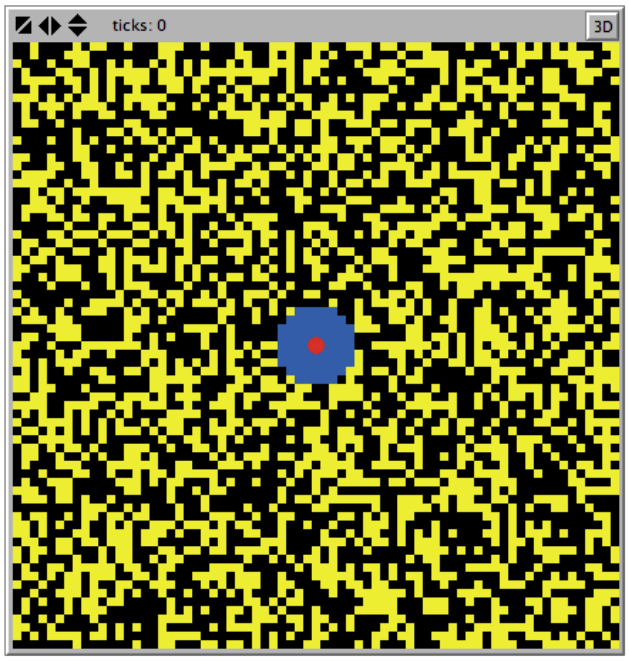
In the next figure we may see a snapshot somewhere in time: the greens are faced towards the ship, following the shortest patch towards the mother kangaroo.



Challenge 3

We will modify the rock landscape—the rocks are scattered randomly in the moon surface. We will have another *slider* to indicate the rock density in the moon surface.
Note that the robots conserve the behavior of Challenge 2.

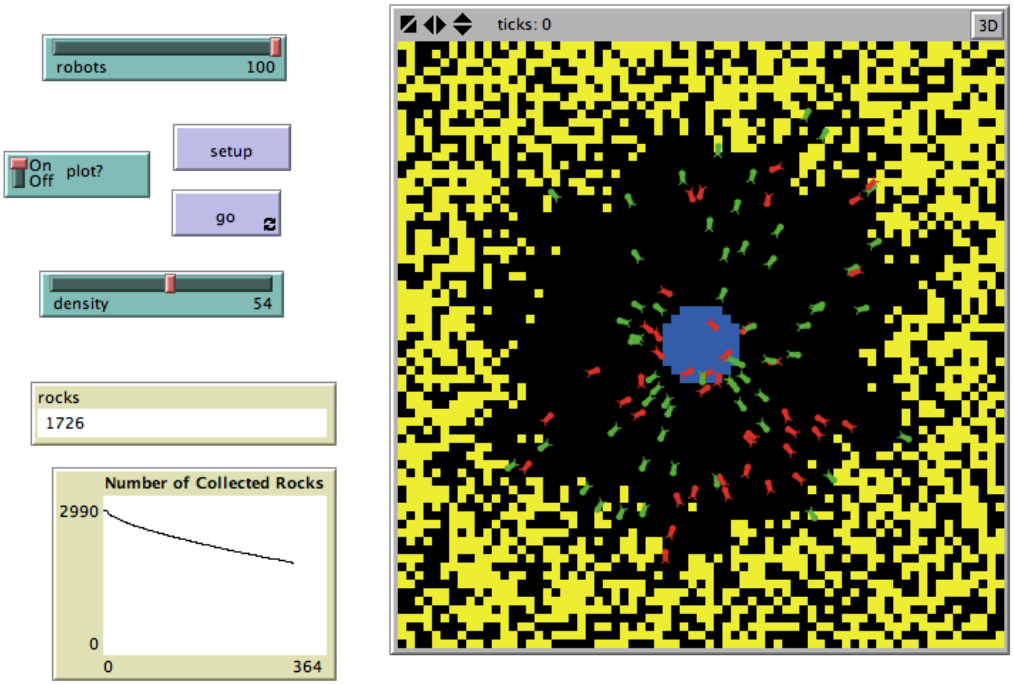




Challenge nº 4

In this Challenge we want to add a *plot* that indicates the evolution of the number of rocks not collected and a *monitor* that indicates the precise number of rocks left on the moon surface.
We consider that the rocks that are being transported and have not yet been deposited are included in the not yet deposited rocks.

Onve again, we show a snapshot of the mission when there were a certain number of rocks left to be deposited. Some of them are on their way to the mother kangaroo robot.



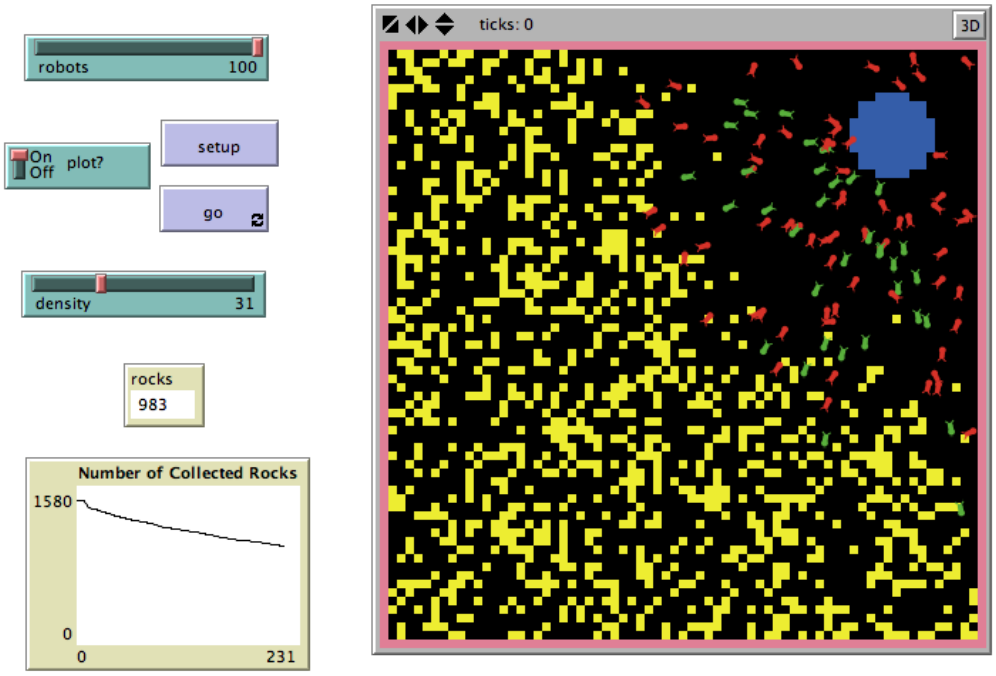
As you may have noticed, the explorers do not have the notion that they have depleted the moon surface. We want to stop everything after all the rocks have arrived to the ship: a mission accomplished button.

Challenge nº 5

Let's make a two variation on the last Challenge.

(1) We change the mother ship location to the top right of the surface.
(2) We are going to limit the surface with a pink wall so that the explorers cannot go through it, being confined. So, when a robot approaches a wall it has to avoid it.

The situation is depicted in the following figure



Challenge nº 6

Now, the robots may carry a number of robots limited to 3. But, there will be robots with different capabilities, they will be born with limited to 1, 2 or 3 rocks.
A robot only returns home after has collected the rocks that correspond to its maximum capacity or after some constant time without finding any rock.

Challenge nº 7

The robots do not like to collide with the others. It will be impossible to have more than 1 robot in each patch. If the patch in front is free a robot will go forward otherwise will change direction randomly.
We want to count the number of robots in each patch and visualize them, so a patch will be colored red if there are two or more robots inside. We also want to monitor the maximum number of robots in every patch.

Challenge nº 8

The conflict avoidance will be ruled by status. The id (*who* variable) of each turtle will correspond to its status. So turtle 0 will have a lower rank than turtle 1, 2, 3, etc. Each turtle will have a vision radius and whenever they see a turtle with a higher rank they will wait until they can move. They will go on moving if they do not see any other robot or when they see only lower rank robots. We would like to follow the lowest and highest ranked robots, for contrast. Create two buttons for activating/inhibiting the following behavior.