1.2 What Is a Model?

A model is a purposeful representation of some real system (Starfield et al. 1990). We build and use models to solve problems or answer questions about a system or a class of systems. In science, we usually want to understand how things work, explain patterns that we have observed, and predict a system's behavior in response to some change. Real systems often are too complex or develop too slowly to be analyzed using experiments. For example, it would be extremely difficult and slow to understand how cities grow and land uses change just with experiments. Therefore, we try to formulate a simplified representation of the system using equations or a computer program that we can then manipulate and experiment on. (To *formulate* a model means to design its assumptions and algorithms.)

But there are many ways of representing a real system (a city or landscape, for example) in a simplified way. How can we know which aspects of the real system to include in the model and which to ignore? To answer this question, the model's purpose is decisive. The question we want to answer with the model serves as a filter: all those aspects of the real system considered irrelevant or insufficiently important *for answering this question* are filtered out. They are ignored in the model, or represented only in a very simplified way.

Let us consider a simple, but not trivial, example: Did you ever search for mushrooms in a forest? Did you ask yourself what the best search strategy might be? If you are a mushroom expert, you would know how to recognize good mushroom habitat, but let us assume you are a neophyte. And even the mushroom expert needs a smaller-scale search strategy because mushrooms are so hard to see—you often almost step on them before seeing them.

You might think of several intuitive strategies, such as scanning an area in wide sweeps but, upon finding a mushroom, turning to smaller-scale sweeps because you know that mushrooms occur in clusters. But what does "large" and "small" and "sweeps" mean, and how long should you search in smaller sweeps until you turn back to larger ones?

Many animal species face similar problems, so it is likely that evolution has equipped them with good adaptive search strategies. (The same is likely true of human organizations searching for prizes such as profit and peace with neighbors.) Albatross, for example, behave like mushroom hunters: they alternate more or less linear long-distance movements with small-scale searching (figure 1.1).

The common feature of the mushroom hunter and the albatross is that their sensing radius is limited—they can only detect what they seek when they are close to it—so they must move. And, often the items searched for are not distributed randomly or regularly but in clusters, so search behavior should be adaptive: it should change once an item is found.

Why would we want to develop a model of this problem? Because even for this simple problem we are not able to develop quantitative mental models. Intuitively we find a search strategy which works quite well, but then we see others who use different strategies and find more mushrooms. Are they just luckier, or are their strategies better?

Now we understand that we need a clearly formulated purpose before we can formulate a model. Imagine that someone simply asked you: "Please, model mushroom hunting in the forest." What should you focus on? On different mushroom species, different forests, identification of good and bad habitats, effects of hunting on mushroom populations, etc.? However, with the purpose "What search strategy maximizes the number of mushrooms found in a certain time?" we know that

- We can ignore trees and vegetation; we only need to take into account that mushrooms are distributed in clusters. Also, we can ignore any other heterogeneity in the forest, such as topography or soil type—they might affect searching a little, but not enough to affect the general answer to our question.
- It will be sufficient to represent the mushroom hunter in a very simplified way: just a moving "point" that has a certain sensing radius and keeps track of how many mushrooms it has found and perhaps how long it has been since it found the last one.

So, now we can formulate a model that includes clusters of items and an individual "agent" that searches for the items in the model world. If it finds a search item, it switches to smaller-scale movement, but if the time since it found the last item exceeds a threshold, it switches back to more straight movement to increase its chance of detecting another cluster of items. If we assume that the ability to detect items does not change with movement speed, we can even ignore speed.

Figure 1.2 shows an example run of such a model, our simple Mushroom Hunt model. In chapter 2 you will start learning NetLogo, the software platform we use in this book, by programming this little model.

This searching problem is so simple that we have good idea of what processes and behaviors are important for modeling it. But how in general can we know whether certain factors are important with regard to the question addressed with a model? The answer is: we can't! That is, exactly, why we have to formulate, implement (program in the computer), and analyze a model: because then we can use mathematics and computer logic to rigorously explore the consequences of our simplifying assumptions.

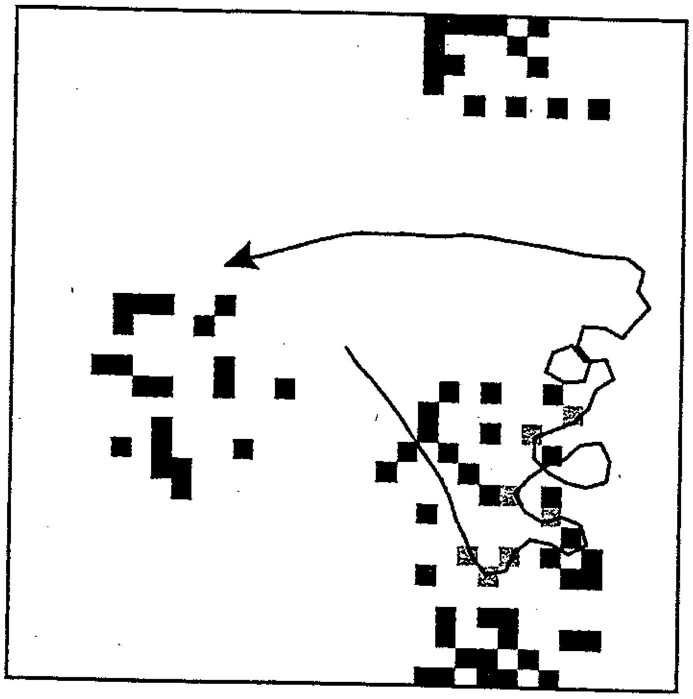

Figure 1.2
Path of a model agent searching for items that are distributed in clusters.

```
globals
[
  num-clusters
]

turtles-own
[
  time-since-last-found
]

to setup

  ca
  set num-clusters 4

  ask n-of num-clusters patches
  [
    ask n-of 20 patches in-radius 5
    [
      set pcolor red
    ]
  ]

  crt 2
  [
    set size 2
    set color yellow
    set time-since-last-found 999
    pen-down
  ]

  reset-ticks

end


to go

  tick

  ask turtles [search]

end


to search

  ifelse time-since-last-found <= 20
    [right (random 181) - 90]
    [right (random 21) - 10]

  fd 1

  ifelse pcolor = red
  [
    set time-since-last-found 0
    set pcolor yellow
  ]
  [
    set time-since-last-found time-since-last-found + 1
  ]

end
```