Modeling Nature's Emergent Patterns with Multi-agent Languages[1]

Uri Wilensky
Center for Connected Learning and Computer-Based Modeling
http://ccl.northwestern.edu/
Northwestern Institute on Complex Systems
Northwestern University
Annenberg Hall 337
Evanston, IL 60208
uri@northwestern.edu

Adapted from Wilensky, U. (2001). Proceedings of EuroLogo 2001 – Linz, Austria

Abstract

NetLogo is a multi-agent modeling language, a parallel extension of Logo. NetLogo is designed to enable scientists to conduct their research by building and analyzing agent-based models and for learners to explore and construct models of emergent phenomena. By exploring and constructing such models, students make connections between the micro-level of agents following rules and the macro-level patterns and regularities that constitute the world of natural and social phenomena.

Keywords: complexity, emergence, modeling, agents

Introduction: the emergence of pattern

Everywhere we look, we see regularities, patterns, order. Many of these patterns have a kind of haunting beauty: the growth of a snowflake crystal, the perimeter pattern of a maple leaf, the advent of a summer squall. Other patterns, such as the dynamics of the Dow Jones or of a fourth grade classroom, seem messier, inchoate, yet still exhibit a familiar and recognizable general "shape". The characteristic shape can unfold in space or in time, sometimes striking and unmistakable and sometimes more hidden, needing probing observation or ingenious experiment to uncover it.

Why is there so much pattern in the world? While grappling with this question in full would take us far afield, we can start with a simple observation: large scale patterns in the world are usually the result of the interactions of large numbers of smaller pieces that somehow combine in surprising ways to create the large-scale pattern. Such large-scale (macro-) patterns that arise out of the interactions of numerous interacting (micro-) "agents" are called "emergent phenomena" – that is, phenomena that emerge from interactions at a lower level or scale.

Visualize a flock of birds winging in the autumn sky or the amazing synchronized fireflies that blink in unison lighting up whole trees in the Far East. How do these patterns come about? All of

[1] This paper was adapted from the original in 2008 and again in 2013 to include link agents and a third example NetLogo model, virus-on-a-network. Thanks to Arthur Hjorth for his helpful comments on the revision.

these patterns are emergent, there is no leader bird which other birds follow, no conductor firefly leading the band -- these patterns emerge out of the behavior of individuals and the adjustment of that behavior in interaction with other individuals.

The study of emergent phenomena is the principal occupation of a developing field of science, the study of complex dynamic systems (Gleick, 1987; Waldrop, 1992; Gell-Mann, 1994; Kelly, 1994; Holland, 1995; Kauffman, 1995). This broad new field seeks to understand how systems of interacting components evolve over time. In the minds of many, however, complex systems theory is not a new branch of science, but rather a new framework, a new perspective that allows us to see old scientific content in new ways. This new perspective and the methods it brings to bear have been adopted across a wide array of natural and social sciences. An understanding of complex systems is becoming an essential part of every scientist's knowledge and skills. The time has come for these ideas and methods to become a central part of *every* student's learning.

Despite its adoption by practicing scientists, the complex systems perspective is largely absent from the K-16 curriculum. One reason for the slow transfer to schools is the heavy reliance of complex systems methodologies on the use of powerful computational technologies. By enabling the rendering, simulation and visualization of the evolution of complex systems over time, the computer has proved an indispensable tool for making sense of complex systems and emergent phenomena. Most of the tools used by experts to explore complexity in their domain of interest are highly domain specific – designed for use by experts to study a particular class of phenomena. Until very recently, no general-purpose tools existed for students to render and explore systems of many interacting parts that can exhibit emergent behavior.

NetLogo: a multi-agent modeling language

Over the past decade, a host of computer languages have been developed for the purpose of constructing models of emergent phenomena. At Northwestern University's Center for Connected Learning (CCL), we have developed such modeling languages (and associated materials) that enable scientists, teachers and students to create dynamic models of complex phenomena. The languages we have developed are now in use by thousands of scientists, students, teachers and researchers worldwide. The latest of these languages, called NetLogo[2] (Wilensky, 1999a), is open-source and freely available at http://ccl.northwestern.edu/netlogo/. NetLogo is the most widely-used of a new class of multi-agent modeling languages (AKA agent-based modeling languages) that have emerged from the complex systems community.

NetLogo is a parallel extension of Logo (Papert, 1980). In NetLogo, however, instead of driving a single turtle, the user can drive (or, perhaps better to say, orchestrate) thousands of turtles. Instead of drawing with pens, turtles "draw" with their bodies. By that, I mean that the emergent shape of all the turtles' positions constitutes a drawing in NetLogo[3]. Allow me to illustrate with a simple example in which turtles take the shape of little triangles or points:

[2] NetLogo is the latest in a family of massively parallel Logos. The first of these was CM-StarLogo implemented on a parallel supercomputer called the Connection Machine (Resnick, 1994; Resnick & Wilensky, 1993). Subsequent to that there have been several newer versions including MacStarLogo (Begel & Resnick, 1995) and StarLogoT (Wilensky, 1997).

[3] In fact, in NetLogo, "turtles" do not typically look like turtles -- they are general purpose "agents" that can take on any shape.

If we initiate a NetLogo session with the command:

```
create-turtles 1000
```

1000 little triangles will appear in the graphics screen. However, because they are initialized to start in the middle of the screen, they are all piled on top of each other and appear as a single point.

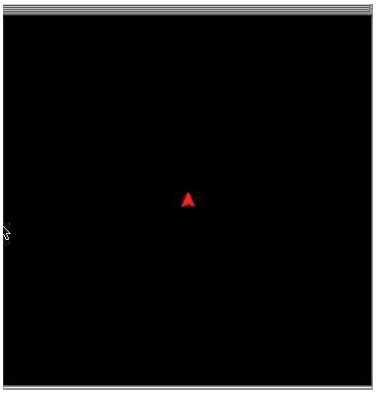


Figure 1: Create-turtles 1000

If we then type the command;

```
ask turtles [forward 20]
```

All of the turtles move forward 20 screen units

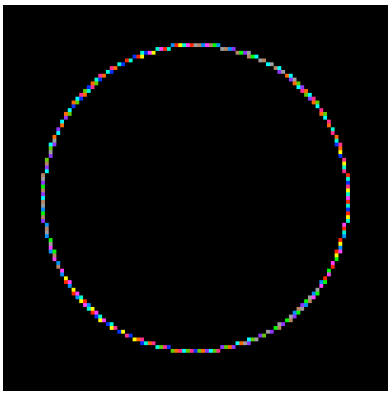


Figure 2: Forward 20

Note that because the turtles were initialized with different “headings”, that is they faced in different directions, they drew the shape of a circle. This is already a simple example of emergent behavior. The fact that there were enough turtles so that by random chance, they were likely to fill the holes, ensured that a coherent circle emerged from the motions of independent turtles.

At first glance, the reader might wonder how the turtles can do anything different and interesting if they all follow the same commands. The power of NetLogo comes from the fact that each turtle is an independent agent. Because each turtle had an independent heading, they all moved in different directions when we typed “fd 20”. Since it is possible for turtles to have as many states as the user likes, the response of turtles to the same commands can vary markedly.

In addition to this difference amongst turtles, each turtle does its own separate computation. To see how this makes a difference, we can type the command “back 20” to get all of the turtles back to the middle of the screen, then invoke the command, “forward random 20”. The function “random” computes a random value between 0 and 20. Because each turtle does its own computation, each one gets a different value for “random 20” and thus will move forward a different amount.

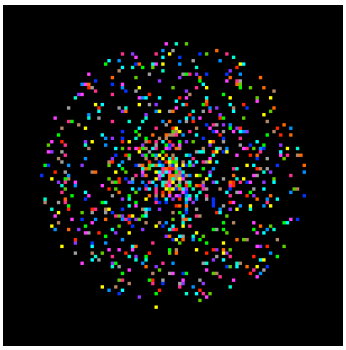


Figure 3: forward random 20

(Beginning students often want to reverse this operation and therefore try the command “`back random 20`”. However, this has unexpected results. Try it.)

In addition to turtles, NetLogo has a second kind of agent that we call a “patch”. Patches are very much like turtles except that they are always around and do not move. The screen is initialized to a (user resizable) grid of patches. In other words, even though the graphics screen looks like empty black where there are no turtles, in reality the patches are invisibly lurking there waiting for commands[4].

If we type the command

```
ask patches [set pcolor red]
```

all the patches will change their color to red.

[4] The grid of patches is essentially a cellular automaton with each patch being a cell.

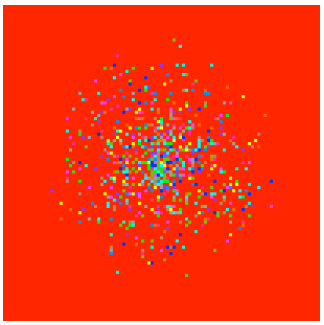


Figure 4: set pcolor red

If we then type the command:

```
ask patches [if pxcor < 0 [set pcolor green]]
```

Then all the patches to the left of the origin turn green. The key point to keep in mind is that they do not do this because they are "told" to do it. They each examine their own position on the screen, determine if they are to the left of the origin and, if so, they turn themselves green.

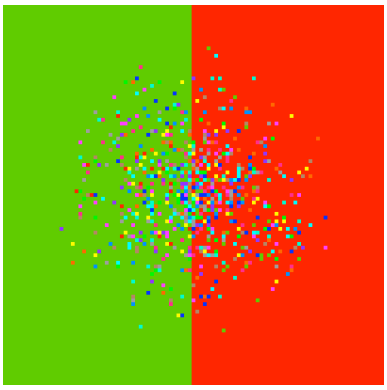


Figure 5: If pxcor < 0 [set pcolor green]

NetLogo has a third basic type of agent, links. A link makes a connection between two turtles. Let's clear the screen and create 100 more turtles scattered randomly about the screen.

```
create-turtles 100 [setxy random-xcor random-ycor]
```

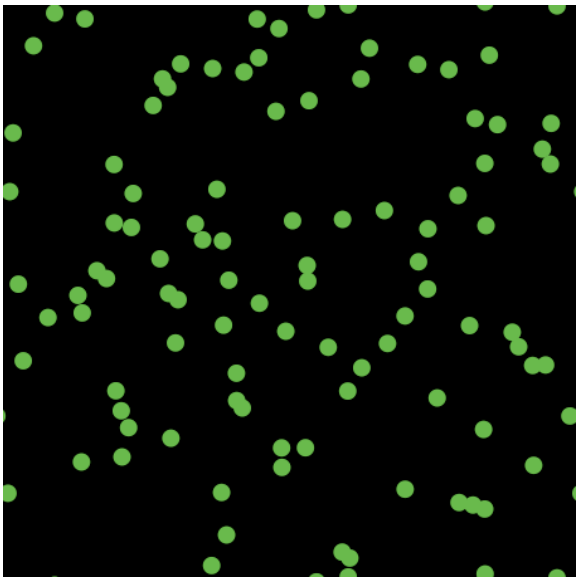


Figure 6. One hundred turtles randomly scattered

We can then ask the turtles to form links with each other:

```
ask turtles [create-link-with one-of other turtles]
```

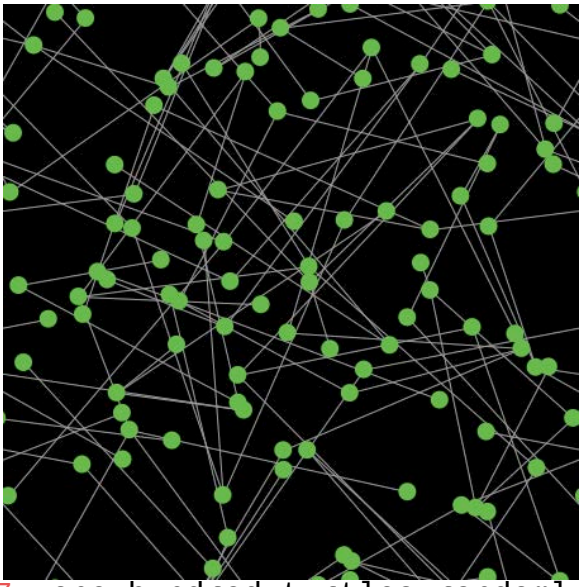


Figure 7: one hundred turtles randomly linked

The network structure is a little hard to see in this picture, so we can use a layout command to better display the network. The layout-spring command causes turtles to attract other turtles that they are directly connected to, and to repel those that they are not directly connected to. The result is a more neatly organized network as seen below.

```
layout-spring turtles links .2 5 1
```

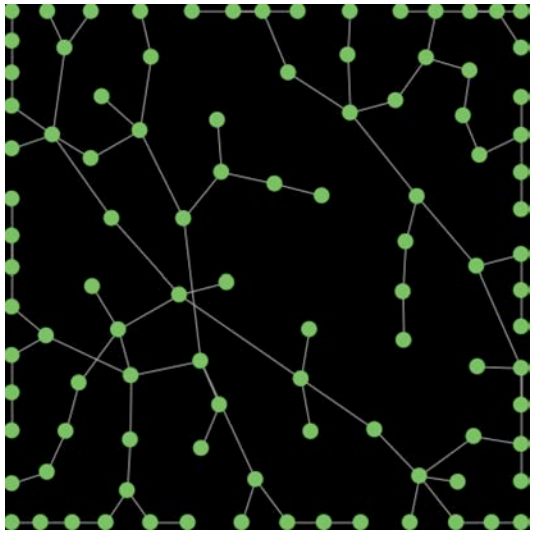


Figure 8. The network displayed in a spring layout

And just like turtles and patches, links can be asked to perform actions as well. For instance, if we want to model a weighted graph, we can give links a ‘weight’ variable. In the code below, we simply assign it a random floating-point number between 0 and 1. Finally, in order to better visualize the weighted structure of the network, we then ask links to set their visual thickness to their weight.

```
ask links [
set weight random-float 1
  set thickness weight
                ]
```

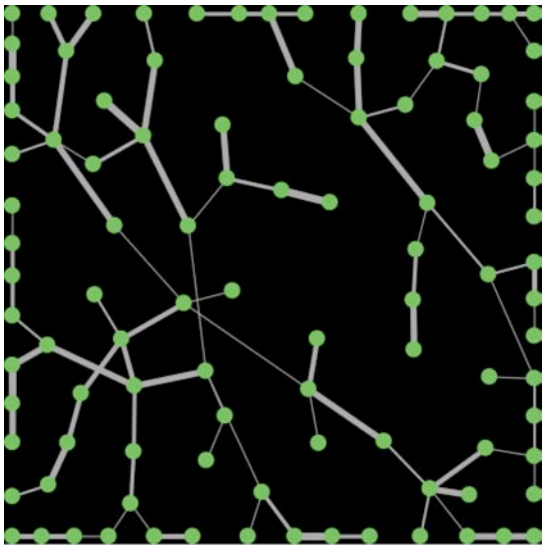


Figure 9. Links display their weight by the thickness of the lines

With these basic tools, we can now create models and dynamic simulations of many different kinds of complex systems.

There is a saying that goes: "If all you have is a hammer, the whole world looks like a nail". With the powerful hammer of the NetLogo language, it becomes easier to see emergent phenomena everywhere. Not only the classic emergent phenomena described in the complex systems literature, but many every day and scientific phenomena can be viewed through the lens of emergent phenomena. While, at first glance, the topic of 'emergent phenomena' seems like an exotic add-on to the curriculum, we see it as a powerful amplifier of understanding for virtually all scientific topics. By enabling us to make cogent and testable connections between the micro- and macro-, the individual and the collective, the element and the system, the new lens makes them easier to understand for novice and for expert learners alike.

The NetLogo environment was designed to be used both by scientists for conducting their research and by teachers and students in learning contexts. NetLogo has achieved widespread adoption in both these contexts, with hundreds of thousands of downloads, thousands of scientific papers and thousands of schools using it in classrooms from middle school though graduate school. I focus below on some examples of secondary classroom use.

Three Examples: Wolf/Sheep Predation, Gas-in-a-Box and Virus-on-a-Network

I will now present three examples of the kinds of models students can build, modify and explore with NetLogo. Students have built models across a wide range of topics and disciplines, including natural sciences such as physics, biology and chemistry social sciences such as economics, history and psychology as well as professional practices such as medicine and law. The two examples presented below are drawn from typical high school science curriculum. The first example is a model of a simple predator-prey ecosystem – a popular model with high school students using NetLogo.

In a typical such model (Wilensky & Reisman, 2006), students model a predator (say a wolf) and a prey (say a sheep). They need to give rules to individual wolves and sheep so that they can move and interact. Many sets of rules are possible. A typical set of rules might assign an energy level to each wolf and sheep and decrease their energy when they move, increase their energy when they eat (wolves eating sheep, sheep eating grass). If their energy falls below 0 they would die. At every turn, they get a random number (roll an imaginary 'die") and if they are lucky they reproduce. Such a model is illustrated below.

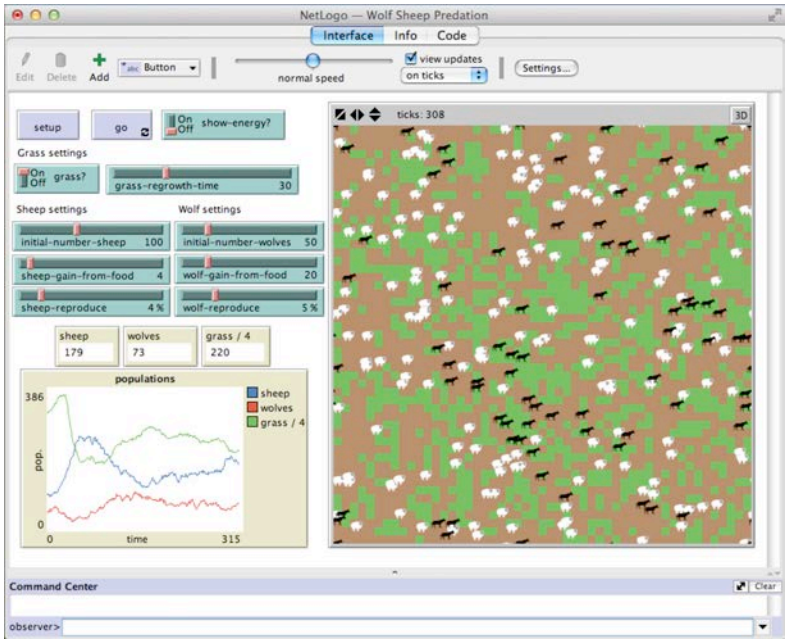


Figure 10: The NetLogo layout for the wolf-sheep predation model (Wilensky, 1997a). In the upper left is the interface area that allows users to set model parameters and run simulations. To its right is the NetLogo graphics window that displays each individual wolf and sheep. Below the interface is a population plot window. At the bottom is a command center for interactively running commands.

A dynamic graph of the population levels of sheep, wolves and grass can be viewed alongside the graphics screen. If the rule sets are chosen appropriately, a typical result is that the population graphs look like out-of-phase sine waves, sheep populations increase till the wolves have so much to eat that they increase which reduces the population of sheep which eventually, in turn decreases the population of wolves which results in an increase in the sheep population.

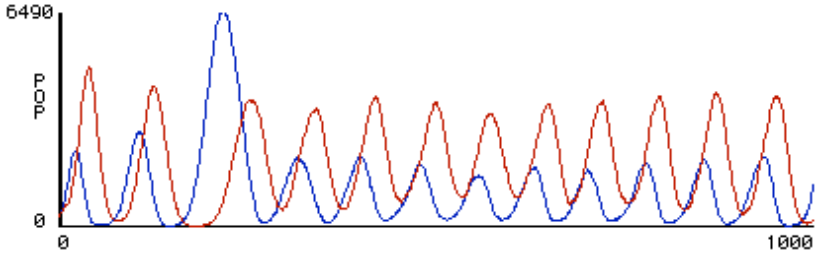


Figure 11. A typical population graph from a student-created rule-set. Red represents size of predator population, while blue represents size of prey population.

This is a classical result, but seen here through the lens of emergent phenomena. The students control the behavior at the micro- level of the individuals and then observe the results at the macro- level of the populations. It is through experimenting with the dynamics of this connection that a powerful understanding of the predator prey dynamics can be achieved.

A second example is a model called Gas-in-a-Box (Wilensky, 1997b), one of a suite of NetLogo models in a package called GasLab (Wilensky, 1999b; Wilensky, Hazzard & Froemke, 1999).

Gas-in-a-Box was originally created by a physics teacher but the original model has been refined by dozens of students who have also created many variants and extensions of the original model.

The basic idea is a box containing thousands of gas molecules. Gas molecules are modeled as turtles that collide like elastic billiard balls, that is they collide with the box and with other molecules without loss of energy. The user can set the mass and speed of any molecule. The speed of molecules is color coded; blue for slow, green for average speed, and red for fast.

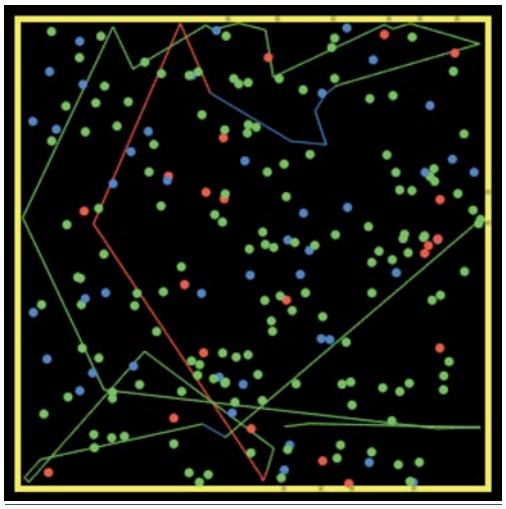
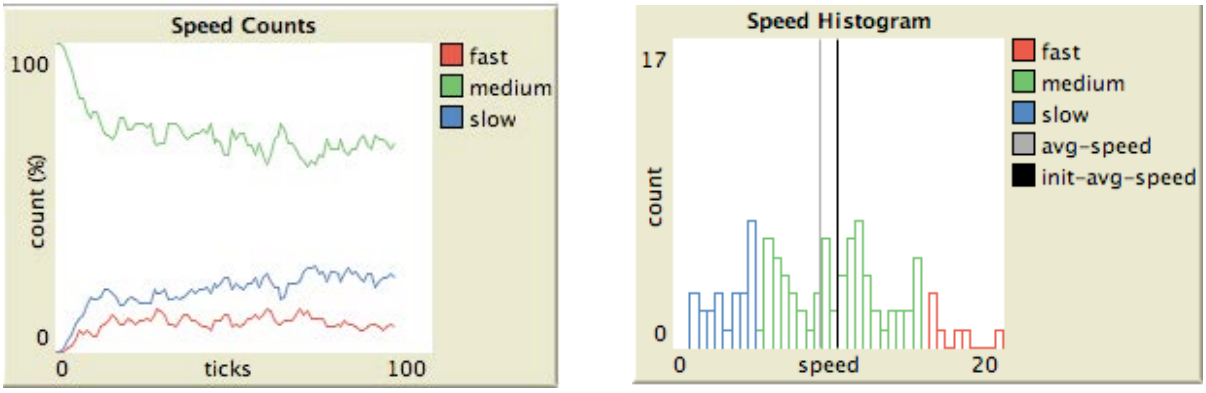


Figure 12: Gas molecules in a box. Fast particles in red, mid-speed in green and slow particles in blue. The zig-zag lines trace the movement of a single particle. Molecules' speed *and* direction change when they collide, explaining why a change in a trace lines' color is always associated with changes in its direction.



a: plot of number of fast, medium, slow particles **b**: histogram of particle speeds

Figure 13: NetLogo displays "live" plots and histograms of the molecules in the graphics window

In a typical first use, students initialize the molecules with equal masses and equal speeds but with random positions and headings. Thus all molecules start out green. They run the model and are usually surprised to see that the molecules turn color quite quickly and that many more of them turn blue than turn green. In other words, more of the particles slow down than speed up. Although this result is a direct consequence of a known law of gases, the Maxwell-Boltzman

distribution of molecular speeds, taught in high school physics, it is not recognized by students in this form. (In our experience, not just students, but even some physicists are surprised by this result.) Again, the key insight here is that the Gas-in-a-Box model allows students to see the gas from an emergent perspective. They come to see the connection between the micro- level of billiard ball collisions and the macro- level of the general characteristics of the gas as an ensemble. These two levels of description are typically taught separately in the high school curriculum. However, it is in understanding the connection between these two levels, how one emerges from the other that leads to a powerful understanding of statistical thermal physics. The connection has been thought to be too hard for high school students, as it usually involves advanced mathematical machinery. But, through the use of multi-agent modeling languages such as NetLogo, these ideas can be accessible to high school learners.

A third example is a model called Virus-on-a-Network (Stonedahl & Wilensky, 2008). This model demonstrates the spread of a virus through a network. It uses NetLogo's newest agent-type, links. Although the model is somewhat abstract, one interpretation is that each node represents a computer, and we are modeling the progress of a computer virus (or worm) through this network. Another is that each node is a person on a social networking site, and that we are modeling the spread of a meme. For the purpose of this description, we will focus on the former, but also briefly discuss the latter interpretation. Each node may be in one of three states: susceptible, infected, or resistant. Such a model is sometimes referred to as an SIR model for epidemics.

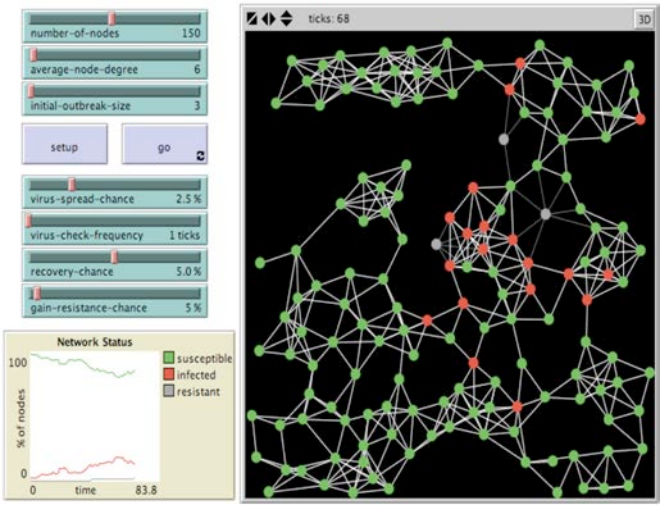


Figure 14. The NetLogo Virus-on-a-Network model. Green nodes are not infected, red nodes are infected and spreading through the network.

The model is initialized with a spatially clustered network with a settable average degree centrality. Each time step, each infected node (colored red) attempts to infect all of its linked neighbors. Susceptible neighbors (colored green) will be infected with a probability given by the VIRUS-SPREAD-CHANCE slider. This might correspond to the probability that someone on the susceptible system actually executes the infected email attachment. Resistant nodes (colored

gray) cannot be infected. This might correspond to up-to-date antivirus software and security patches that make a computer immune to this particular virus.

Infected nodes are not immediately aware that they are infected. Only every so often (determined by the VIRUS-CHECK-FREQUENCY slider) do the nodes check whether they are infected by a virus. This might correspond to a regularly scheduled virus-scan procedure, or simply a human noticing something fishy about how the computer is behaving. When the virus has been detected, there is a probability that the virus will be removed (determined by the RECOVERY-CHANCE slider). If a node does recover, there is some probability that it will become resistant to this virus in the future (given by the GAIN-RESISTANCE-CHANCE slider).

When a node becomes resistant, the links between it and its neighbors are darkened, since they are no longer possible vectors for spreading the virus.

Students notice that at the end of a run, after the virus has died out, some nodes are still susceptible, while others have become immune. They typically explore what is the ratio of the number of immune nodes to the number of susceptible nodes, and how is this affected by changing the AVERAGE-NODE-DEGREE of the network. They also explore under what conditions the virus can live on and how this is affected by the GAIN-RESISTANCE-CHANCE, the RECOVERY-CHANCE and the VIRUS-SPREAD-CHANCE.

This same model could show how a meme spreads on social networking sites: “Susceptible” nodes could be people who would find the meme interesting or funny, but who have not seen it before. “Infected” nodes could be people who recently saw the meme, liked it, and are considering sharing it. “Recovered” nodes could be seen as people who already saw and shared the meme, or who simply did not find that particular meme funny or interesting to begin with. A few modifications would be necessary for this new interpretation of the model: Whereas all infected computers in the model are equally likely to spread a virus, some people “share” a lot online, while others do not. To account for this difference between individuals, we could give each node a percentage chance of sharing. Some people may be more or less likely to share a meme that they have seen many times before. Each node could be asked to count how many times they have seen the meme, and base their decision of whether to share it or not on that. Finally, some people may be more likely to share a meme posted by good friends of theirs. Edges in the model could be weighted to show the closeness of the personal relationship between nodes, and the decision to share could take this into account as well.

In contrast to this simple model, both real computer networks and real social networks are often found to exhibit a "scale-free" link-degree distribution, using a Preferential Attachment (Barabasi & Albert, 1999; See also Wilensky, 2005) mechanism. Using the NetLogo Networks extension[5] students are able to easily experiment with various alternative network structures, and see how the behavior of the virus differs. Such network models enable students to combine concepts from network theory with ABM methodologies so they can better explore dynamics on networks, both in terms of spread of a virus on fixed networks as well as dynamic formation and breaking of

[5] Available from: https://github.com/NetLogo/NW-Extension .

links. In this way, NetLogo allows learners to explore how the combination of the structural network centrality, and the propensities of individual nodes contribute to the dynamic aggregate behavior of networks.

NetLogo in the classroom

The multi-agent modeling paradigm is in use by many students, teachers and classrooms. In its years of use, we have assembled a large library of "extensible" models.[6] These sample models are drawn from a wide range of disciplines including physics, biology, mathematics, computer science, chemistry, materials science, ecology and economics. These models are created by students, teachers and researchers and go through a process of checkout and refinement before becoming a part of the distribution archive.

In the classroom, NetLogo is typically used in roughly five phases:
a) In the first phase, the teacher typically leads the students in off-computer activities (known as participatory simulations or emergent activities) that provoke thinking about emergent phenomena. In these activities, students typically enact the role of individual elements of a system and then discuss amongst themselves what global patterns they detect and how those patterns could arise from their individual behaviors. These participatory simulations can be done without technology in what we call "StarPeople" (Resnick & Wilensky, 1998) or "NetPeople" or they can be supported by wireless technologies as in the HubNet architecture (Wilensky & Stroup, 1999) which enables students to use handheld devices to control agents within a NetLogo simulation.
b) In the second phase, the teacher presents a "seed" model (a simple starting model) to the whole class, projected upfront so that everyone can view it. The teacher engages the class in discussion as to what is going on. Why are they observing that particular behavior? How would it be different if model parameters were changed? Is this a good model of the phenomenon it is meant to simulate?
c) In the third phase, students run the model (either singly or in small groups) on individual computers and explore the parameter space of the model.
d) In the fourth phase, each modeler (or group) proposes an extension to the model and implements that extension in the NetLogo language. Modelers starting with GasLab, for example, might try to add to the model by building a pressure gauge, a piston, a gravity mechanism, or heating/cooling plates. The extended models are added to the project's library of extensible models and made available for others to work with as "seed" models.
e) In the final phase, students are asked to propose a phenomenon and build a model of it from "scratch" using the NetLogo modeling primitives.

There are other simulation software packages in use in school settings that enable students to engage in phases b & c. However, because the packages are "black-box", not inspectable or modifiable by students, they do not enable students to engage in phases d) and e) or to go more deeply into understanding the models. Yet other simulation packages, notably STELLA (Richmond & Peterson, 1990), are "glass-box" like NetLogo, but they ask student modelers to model at the level of populations. By enabling students to model at the level of individuals,

[6] The largest collection is the NetLogo Models Library (Wilensky, 1999c), which is part of the basic NetLogo distribution and is also viewable at http://ccl.northwestern.edu/netlogo/models.

NetLogo makes it easier for students to begin modeling as they can base their models on their own experience of individuals.

We have worked with classrooms in all five of these phases. Generally, the depth of understanding of complex systems and emergent phenomena would be expected to increase as the student more actively builds, modifies, and explores with the model. The results that students can achieve with model extensions and designing their own models are often quite dramatic. Because of the great variations in available technology, learning time, and classroom organization, each phase has valuable applications.

Working in phase d), what we call the "extensible modeling" approach, allows learners to dive right into the model content. Learners typically start by exploring the model at the level of domain content. When they are puzzled by an outcome of the model, they design an extension to the basic model. This extension usually requires only a few language primitives to implement. This allows learners to follow a gently sloping path towards full NetLogo language mastery — skill with the general purpose modeling language is acquired gradually as learners seek to explain their experiments and extend the capabilities of the model.

Conclusion

The inclusion of a complex systems perspective in school curriculum would have many benefits for learners:

• We live in an increasingly interconnected world. Rainforest destruction in South America leads to Greenhouse effects and weather pattern changes in Africa. Market collapses in the Far East can wreak great consequences on economies in the West. Traditional science, which studies phenomena in isolation, is not equipped to analyze and understand such systemic effects. Scientists are beginning to use complex systems tools and methods to study such phenomena. Informed citizens as well in such a highly interacting world need tools that can help them cope with these complexities.

• Though there is increased desire for interdisciplinary learning, students studying in a traditional curricular framework find it difficult to see the connections between different domains of knowledge. One strength of the complex systems theory perspective is that it enables us to see common patterns across traditionally separate fields: physical matter is the emergent result of molecular interactions; ecologies and biological niches are emergent results of interacting organisms; economies and markets are emergent results of the interactions of buyers and sellers.

• Many everyday phenomena and experiences arise from the interactions of many different factors. Because these have been hard to study using traditional methods, they are excluded from the curriculum. Introducing complex systems allows students' personal experiences to be included in the curriculum – thus students see science as more personally relevant.

• An understanding of patterns as emergent phenomena, rather than as results of equations, is both a more accurate picture of nature AND easier for most people to understand. Science becomes more accessible, not less, as a result of this change in viewpoint.

By introducing a perspective of complexity and emergent phenomena, we make science more accurate, more inclusive and more accessible to the great majority of students.

References

Barabási, A.-L. & Albert, R. (1999). "Emergence of scaling in random networks". *Science* **286** (5439): 509–512.

Forrester, J.W. (1968). *Principles of Systems*. Norwalk, CT: Productivity Press.

Gell-Mann, M. (1994). *The Quark and the Jaguar*. New York: W.H. Freeman.

Gleick, J. (1987). *Chaos*. New York: Viking Penguin.

Holland, J. (1995). *Hidden Order: How Adaptation Builds Complexity*. Helix Books/Addison-Wesley.

Kauffman, S. (1995). *At Home in the Universe: The Search for the Laws of Self-Organization and Complexity*. Oxford: Oxford University Press.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

Resnick, M. (1994). *Turtles, Termites and Traffic Jams: Explorations in Massively Parallel Microworlds*. Cambridge, MA: MIT Press.

Resnick, M. & Wilensky, U. (1993). Beyond the Deterministic, Centralized Mindsets: New Thinking for New Sciences, Paper resented at *the annual conference of the American Educational Research Association*, Atlanta, Ga.

Resnick, M., & Wilensky, U. (1998). Diving into Complexity: Developing Probabilistic Decentralized Thinking Through Role-Playing Activities. *Journal of the Learning Sciences*, 7 (2), 153-171.

Richmond, B., & Peterson, S. (1990). *Stella II*. Hanover, NH: High Performance Systems, Inc.

Stonedahl, F. and Wilensky, U. (2008). NetLogo Virus on a Network model. http://ccl.northwestern.edu/netlogo/models/VirusonaNetwork. Center for Connected Learning and Computer-Based Modeling, Northwestern Institute on Complex Systems, Northwestern University, Evanston, IL.

Waldrop, M. (1992). *Complexity: The emerging order at the edge of order and chaos*. New York: Simon & Schuster.

Wilensky, U. & Reisman, K. (2006). Thinking like a Wolf, a sheep or a Firefly: Learning Biology through Constructing and Testing Computational Theories -- *an Embodied Modeling Approach***.** *Cognition & Instruction.*

Wilensky, U. & Stroup, W. (1999). Learning through Participatory Simulations: Network-based Design for Systems Learning in Classrooms. *American Educational Research Association*. Montreal, Canada.

Wilensky, U. & Resnick, M. (1999). Thinking in Levels: A Dynamic Systems Perspective to Making Sense of the World. *Journal of Science Education and Technology*. Vol. 8 No. 1.

Wilensky, U. (1999a). *NetLogo*. [Computer software] Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University. http://ccl.northwestern.edu/cm/netlogo/

Wilensky, U. (1999b). GasLab—an Extensible Modeling Toolkit for Exploring Micro- and Macro- Views of Gases. In Roberts, N., Feurzeig, W. & Hunter, B. (Eds.) *Computer Modeling and Simulation in Science Education*. Berlin: Springer Verlag.

Wilensky, U. (1997a). NetLogo Wolf Sheep predation model. Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University. http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation

Wilensky, U. (1997b). NetLogo GasLab Gas in a Box model. Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University. http://ccl.northwestern.edu/netlogo/models/GasLabGasinaBox .

Wilensky, U. (2005). NetLogo Preferential Attachment model. http://ccl.northwestern.edu/netlogo/models/PreferentialAttachment. Center for Connected Learning and Computer-Based Modeling, Northwestern Institute on Complex Systems, Northwestern University, Evanston, IL.

Wiensky, U., Hazzard, E & Froemke, R. (1999). An Extensible Modeling Toolkit for Exploring Statistical Mechanics. Proceedings of the Seventh European Logo Conference – Proceedings of EUROLOGO '99, Sofia, Bulgaria.

Wilensky, U. (1999c). Netlogo Models Library. Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University. http://ccl.northwestern.edu/netlogo/models/ .

Wilensky, U. (1997c). *StarLogoT*. Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University. http://ccl.northwestern.edu/cm/starlogot/