Introduction to Complex Systems

Overview:

During the last three decades a leap has been made from the application of computing to help scientists ‘do’ science to the integration of computer science concepts, tools and theorems into the very fabric of science. The modeling of complex adaptive systems (CAS) is an example of such an integration of computer science into the very fabric of science; models of complex systems are used to understand, predict and prevent the most daunting problems we face today; issues such as climate change, loss of biodiversity, energy consumption and virulent disease affect us all. The study of complex adaptive systems has come to be seen as a scientific frontier, and an increasing ability to interact systematically with highly complex systems that transcend separate disciplines will have a profound affect on future science, engineering and industry as well as in the management of our planet’s resources (Emmott et al., 2006).

The name itself, “complex adaptive systems” conjures up images of complicated ideas that might be too difficult for a novice to understand. Instead, the study of CAS does exactly the opposite; it creates a unified method of studying disparate systems that elucidates the processes by which they operate. A complex system is simply one in which many independent elements or agents interact, leading to emergent outcomes that are often difficult (or impossible) to predict simply by looking at the individual interactions. The “complex” part of CAS refers in fact to the vast interconnectedness of these systems. Using the principles of CAS to study these topics as related disciplines that can be better understood through the application of models, rather than a disparate collection of facts can strengthen learner’s understanding of these topics and prepare them to understand other systems by applying similar methods of analysis (Emmott et al., 2006).

Learning Objectives:

This is a one-hour class designed to introduce middle and high school students to complex systems. In this class we will ask students to participate in several simulations of complex systems. Through their participation, students will gain an understanding of complex systems, learn the characteristics of complex systems, and experience scenarios in which the simple behaviors of agents produce emergent structure.

About Complex Systems

What are Complex systems?

(a.k.a. Complex Dynamic Systems or Complex Adaptive systems)

Complex = difficult-to-understand or difficult to predict

Dynamic = moving, changing

Adaptive = changing to adapt to an environment or condition

Complex Systems are collections of simple units or agents interacting in a system.

Large-scale system whose behaviors may change, evolve, or adapt.

Some examples through activities:

1. Turn and Walk (10 minutes)
In this simulation, participants are asked to stand in a circle. They are told that they are "agents" in a simulation. As agents they will have a very specific set of instructions that they will follow. First, they will turn to face the person directly to the right. Second, they are to remain pointing in that direction as they take three steps forward. This set of instructions will be repeated each time the instructor says "go".
Discuss what the outcome might be.

Try out the instructions.

Discuss what happened. What did you observe?
What would happen if the instructions were changed to 5 steps?
Discuss what would happen if they started off in a different configuration/shape.

2. Swords and Shields (20 minutes)
In this simulation, participants are asked to select one person to be their "sword" and a different person to be their "shield". They are told that their objective is to always have their shield between them and their sword (thus protecting them from the sword.)
Show the diagram. Ask the participants to walk around randomly and when told to go, they must move in such a way that their shield is always protecting them from their sword.
Ask for predictions on what might happen.

Try out the instructions.

Discuss what happened and why.

Wrap-up Discussion and demonstration of StarLogo models
Discuss characteristics of complex systems:
Patterns emerge from simple interactions of its agents
    What patterns emerged in the previous simulations?
There is no central control – it is a decentralized system
    How is this seen in the previous simulations?
The system self-organizes – it spontaneously generates a well-defined entity by self-assembling from individual components.
    Give an example from the simulations; ask for an example in nature…

What are some other examples of complex systems?

StarLogo models
1. Turn and Walk

2. Swords and Shields

Follow-up questions:
Why do we use computer simulations to study complex systems? What are the variables we can control? Will we get the same results each time a simulation is run? Why or why not?

Global climate patterns
A termite mound
Highway traffic patterns
The spread of a disease in a population
The evolution of ideas in a society
A food web in an ecosystem

Necessary tools
StarLogo 2.2
OpenStarLogo

Web references
http://education.mit.edu/starlogo
http://education.mit.edu/starlogo/models

Reference books
Science 2020
Adventures in Modeling, Klopfer

Other examples

Introduction to Computer Programming concepts using TNG

Overview
In this class, we will use the StarLogo TNG language to introduce basic concepts used in computer programs.

Learning Objectives
Over the course of this class you will learn
- Abstraction & decomposition of a problem
- Algorithmic thinking
- Instructions
- Simple control structures
- Evaluating expressions
- Repeating behavior
- How to create a control structure incorporating the above.

About Computer Programming
Computer programming is the process of planning and creating a sequence of steps for a computer to follow. In general, this process will help us solve a problem that is either too tedious or too difficult to work out by hand.

About StarLogo TNG
StarLogo TNG ("The Next Generation") is the newest branch of the StarLogo language that provides two significant advances over the previous version, StarLogo 2.2. First, the programming is now done with programming blocks instead of text commands. The primary advantage of this innovation is that the program has moved from the abstract to the visual. As many students are already familiar with using a computer by dragging and dropping objects, StarLogo TNG comes more naturally to them. The programming blocks are arranged by color and shape based on their function, and it enables students to associate similar programming blocks with each other. Since the programming blocks are puzzle piece shaped, only blocks can fit together only in syntactically sensible ways, which eliminates many sources of problems for students. StarLogo TNG also provides visual organization capabilities, such as "breed drawers", which provide similar functionality to object oriented programming. StarLogo TNG's second significant advance is a 3D representation of the world. This in turn presents several new opportunities. These include the ability to model new kinds of physical phenomena, and a feature that allows students to take the perspective of an individual agent in the environment, which helps them bridge the description of individual behaviors with system level outcomes.

Necessary tools
StarLogo TNG is available for free download at http://education.mit.edu/starlogo
StarLogo TNG requires a 1GigaHertz (GHz) or better processor and a graphics card that supports OpenGL. We recommend 512mb of RAM or more, but it may run with less.
If you are running StarLogo TNG on a Windows System, we recommend a Pentium 4/Celeron/Athlon 1.8GHz or better or Pentium M 1.4GHz or better for the procesor and a

nVidia (GeForce 5200 or better) or ATI (Radeon 7000 or better) graphics card. DirectX is required. If you are running on a Macintosh System, any newer 1 ghz or better iMac, eMac, Powerbook, or iBook will do. The Macintosh version is a Universal Binary so it runs natively on Intel and PPC Macs. You will need Mac OS 10.4.6 or higher along with Java 5 release 4 or later installed. If you don't have Java 5 release 4 or later installed, it is available via Software Update or directly from Apple for PPC or Intel machines.

Web references
Reference books
Contact information

Computer Programming basics

- Abstraction & decomposition of a problem
- Algorithmic thinking
- Instructions
- Simple control structures
- Evaluating expressions
- Repeating behavior

How to create a control structure incorporating the above.

Basic commands in TNG

An Example

Algorithmic thinking - PB&J robot example (give students 10 minutes)
Act out one or two student provided algorithms (5 minutes)
Abstraction & decomposition of a problem
A program is a list of Instructions
Simple control structures
Evaluating expressions
Repeating behavior
TNG Maze model: given a maze and a character, build control structure for user to control the character walking through a maze.

Introduction to StarLogo
Overview
Learning Objectives
About Computer Programming
About StarLogo TNG
Necessary tools
Web references
Reference books
Contact information
Computer Programming basics
Basic commands in TNG

An Example
Observer, Agent, Environment paradigm
Simple commands in StarLogo
M&Ms example (draw analogy to maze example in TNG)
States and Behaviors
Breeds and Patches
Litterbugs Model (open a template with breeds in place, students only fill in the behaviors and logic)

Introduction to Computational Science
Overview
Learning Objectives
About Computer Programming
About StarLogo TNG
Necessary tools
Web references
Reference books
Contact information
Computer Programming basics
Basic commands in TNG

An Example "You be the judge" game
where students vote on "is this an acceptable challenge project?"
need 5 or 6 example project descriptions (from archives)
need note cards with "Accept" & "Reject" & big stickies
Instructor reads description, students place initial votes, recorder records votes
Discussion of what makes a good project
Looking at the Challenge Rubric? our criteria
Allow students to change their votes - look at new vote counts
Unveil Challenge judges' responses (mock video clips)
The Computational Science Process Map using an Epidemiological example
Look at the Epidemic Model, what makes it a good CS problem?
Have the students conduct multiple runs of an experiment (each group using different settings)
need cards with what settings to use, and data sheets
Look at real data from English boarding school epidemic... (info is in Dick's PowerPoint)
Discuss whether model is "reasonable"

Computational Science Process and Map