Introduction to Computer Programming concepts using StarLogo 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

- What is a computer program
- Abstraction & decomposition of a problem
- Algorithmic thinking
- Instructions
- Conditional instructions
- 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 was created to

1. Lower the barrier to entry for programming by making programming easier.
2. Entice more young people into programming through tools that facilitate making games.
3. Create compelling 3D worlds that encompass rich games and simulations.

StarLogo TNG 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.

Our approach
The approach we use in this course is to allow students to construct on-screen computer games in which the concepts of agents with behaviors and environment is made tangible and visible. Onscreen agents (or objects) in StarLogo TNG populate a three-dimensional environment. Students create computer programs by dragging and dropping program blocks into place on the block canvas. We will introduce basic programming elements (if/then statements, loops, variables, expressions) using the TNG blocks and give students an early exposure to event handling.

Web references
StarLogo TNG documentation
StarLogo TNG video tutorials

Reference books
Contact information

Computer Programming basics

What is a computer program?
A computer program is a set of instructions that tell the computer what to do. Computer programming entails decomposing the problem into a sequence of steps and specifying them sufficiently, precisely, unambiguously, and primitively such that the computer interpreting the program can effectively realize the intended solution.

Abstraction and decomposition of a problem
By abstraction, we mean learning how to communicate complex ideas simply and to decompose a problem logically. For example, think how you might tell a robot how to bake a cake. You might break the problem down into smaller pieces: make the batter, bake the cake, decorate the cake. Each of these can in turn be broken down in to smaller pieces: make the batter becomes beat the eggs and the butter, stir in the flour and sugar, then pour batter into a baking pan. This process is called problem deconstruction and it simply means breaking a problem down into a list of simpler tasks.

Algorithmic thinking
Algorithmic thinking means thinking in terms of instructions that the computer can understand. Once a problem is decomposed into smaller pieces, we can think about how to tell the computer a set of instructions to carry out each of those smaller pieces. Algorithmic thinking is thinking about arranging a sequence of instructions to carry out a task.

An activity: Peanut Butter & Jelly Robot
Write an algorithm for a robot to make a peanut butter and jelly sandwich.

Instructions
An instruction is the simplest kind of direction to give a computer. In StarLogo TNG, a simple instruction (also called a command) you might give is "go forward two steps". The command in TNG would look like this

Moves all of the agents forwards 2 steps.

Another simple command is

Makes agents turn 90 degrees to the right.

We say an instruction is "executed" when it is run by the computer.

Arguments
Some commands like the ones above need arguments (in these cases, numbers) as part of the instructions. These numbers tell the computer how much to go forward or how many degrees to turn. Here's an example of an instruction that takes two arguments, an x position and a y position.

To complete this command, we need to add the arguments that fit to complete the command, for example,

sets the turtle's position to 3,8.

When modeling complex systems often we want some degree of randomness in an agents behavior. Suppose that instead of always turning 90 degrees to the right, we want to turn some random number of degrees to the right. To do this we would use the random function

This function returns a value between 0 and 89. When used as the argument to an instruction, the function provides a new random value as an argument to the instruction each time the instruction is executed.

Conditional instructions

More complicated instructions are used to tell a computer what to do under certain circumstances. For example, we might want to say "go forward three steps if nothing is blocking the way." For this we need a conditional instruction. Programming with conditional instructions allow for steps to be carried out only in specific situations. This is what the if/then block looks like:

To use it, we need to insert a test condition and tell the computer what to do if the test is true.

Evaluating Expressions

Expressions are questions that get resolved to a value. Conditional statements like the one above make use of expressions to check a current condition. Often checking this condition requires a comparison. For example, the test condition could be the "if the turtle is green". The expression "turtle is green" gets evaluated as true or false.

When used in a conditional statement, the test condition is the expression. If expression evaluates to true (because the turtle is in fact green) then the statement(s) in the "then" portion are executed.

In this example, if the turtle is green, then it turns 180 degrees to the right.

Repeating behavior

Computer programs use repeat loops that carry out the set of instructions with in the loop a number of times. Here's what the Repeat block looks like:

To use it, we need to insert how many times to repeat and tell the computer the statements to repeat. For example,

makes the turtle take one step backwards ten times in a row.

In TNG often we want the turtles to do something continuously, or forever, until turned off. For this we use the control structure

- add forever loop here

User-driven events

In creating a game, user input is vital to game play. StarLogo TNG has an easy way to check for user-driven events.

An Example

Now that you know some simple commands, conditional statements and repeat loops, let's build see them in action. Open the StarLogo TNG project called MazeWalker.sltng TNG Maze model: given a maze and a character, build control structure for user to control the character walking through a maze.

Create turtle, position turtle
Turtle POV
Turtle movement
Kill turtle

Turtle control by user
Conditionals

Expressions

Repeating

Breeds

Collisions