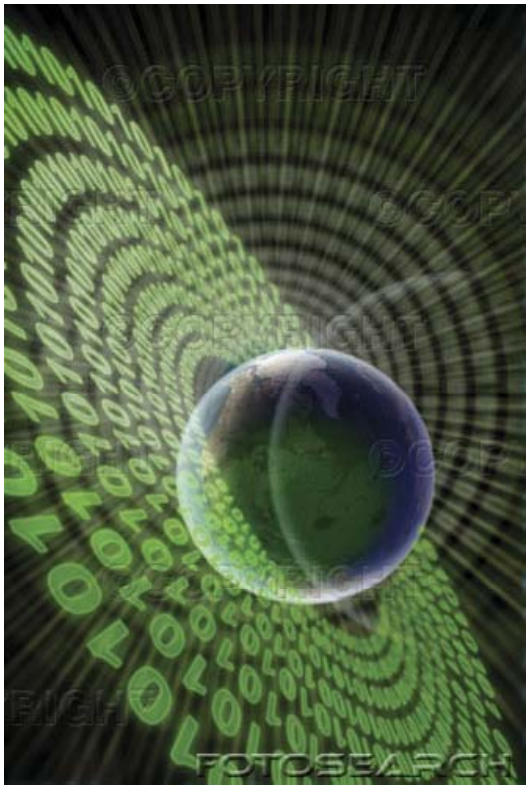
Drexel-SDP GK-12 ACTIVITY

Science

Scratch Programming and Problem Solving

Introduction to Scratch - Loops



Grade Level _6_ (_5_-_8_)

Lesson # *1*

Lesson Dependency Lessons in the Scratch group

Time Required: 2 Hours

Summary

In this lesson students will explore the Scratch programming environment, first in an unstructured way, and then with a directed challenge. This challenge will be solved using a scaffolded approach in which students are presented with the solution by discussing it as a class, and then reproducing the solution on their own in the Scratch environment. For now, more emphasis is placed on becoming familiar with the environment than with producing the actual logic steps, but students will get a little experience with this as well. Students will explore closed-path motion as a 360 degree net movement, and expand on this in the next lesson to programmatically form general tessellations.

Engineering Connection

A knowledge of software engineering facilitates in the general approach to problem solving. The goal is to provide students with the skills needed to identify processes appropriate to solving a particular problem. Often students have the requisite technical skills to solve a problem like division, but they do not have the skills required to identify when division is appropriate to solve a problem. Rather, they will only use division when it is explicitly asked of them. Computers are exceptional problem solving tools, but they require the user to direct them in their action. Scratch, as a kid-friendly programming environment, allows students to explore problem solving and obtain immediate feedback.

Keywords

- Scratch
- Programming
- Software Engineering
- Problem Solving
- Loops

PA Science Educational Standards

- 3.1.7A Explain the parts of a simple system and their relationship to each other.
- 3.1.7B Describe the use of models as an application of scientific or technological concepts.
- 3.2.7A Explain and apply scientific and technological knowledge.
- 3.2.7C Identify and use the elements of scientific inquiry to solve problems.

Learning Objectives

After this lesson, students should be able to:

- Run the Scratch environment.
- Add variables to the Scratch environment.
- Issue motion commands to the Scratch sprite.
- Transform motion commands into a loop structure in Scratch.

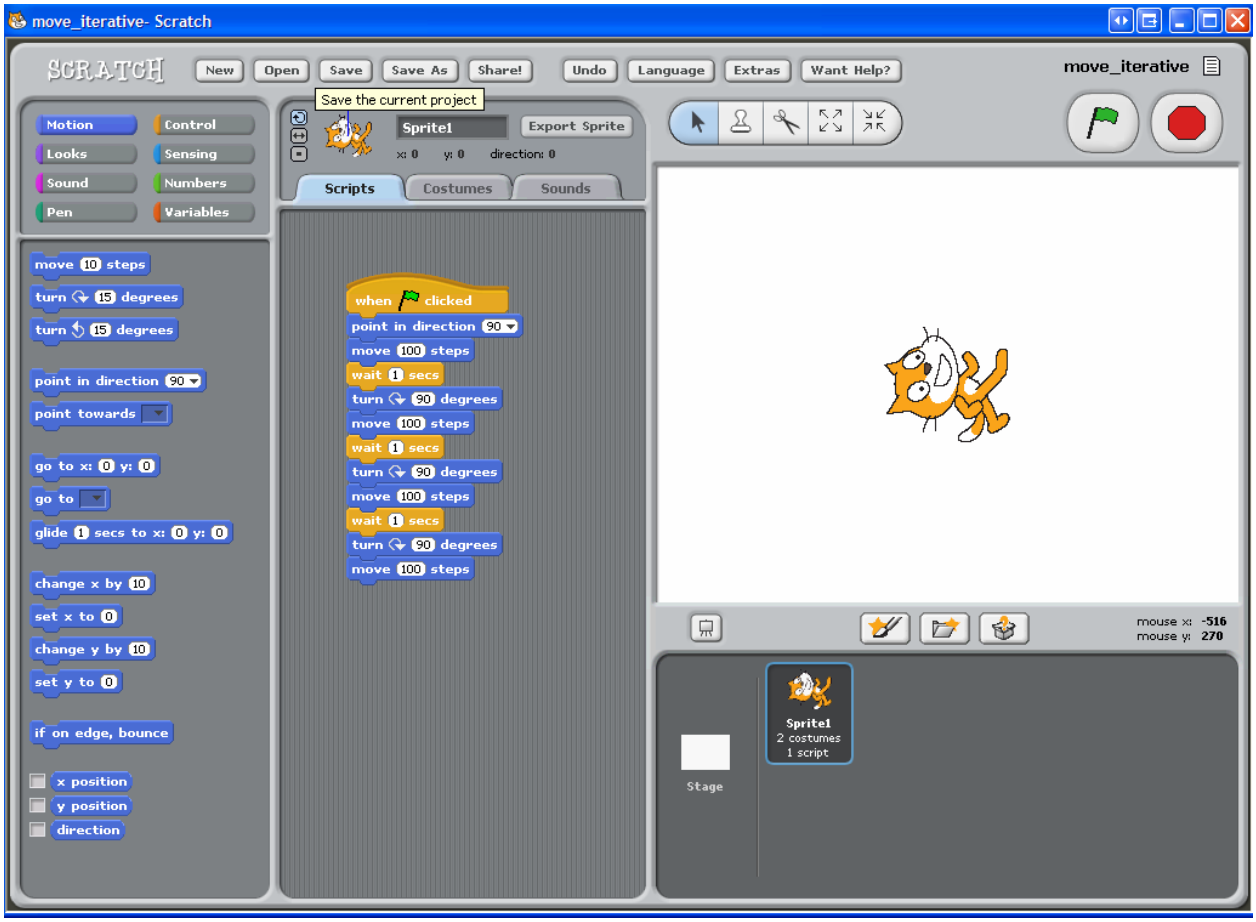
Introduction / Motivation

Scratch is a kid-friendly programming, story-writing, and problem solving environment. Under the hood, computers really work only on ones and zeroes. This means that computers are just a bunch of switches that are either on (1) or off (0). Different combinations of ones and zeroes (on and off switches) mean different things or direct the computer to do different things. Therefore, a program that we use every day is just a very complex combination of millions of ones and zeroes. This would be far too complex for humans to create, and as a result we have programming languages that let us write out a list of steps for the computer to accomplish. We have other programs that translate these commands into those combinations of ones and zeroes. Scratch is one such translator. We can drag and drop steps like lego blocks onto the Scratch pad, and the computer will execute them exactly in order. Even with this extra ease, it is essential that software engineers express exactly what they want the computer to do. One small mistake or ambiguity will result in error, because the computer will do only what it is asked to do.

Begin by observing the top left area of Scratch. It contains the groups of things that we are able to do. We will start every program with the Control item "When FLAG clicked." Notice the green flag and red stop sign on the top right. These are the "start" and "stop" buttons for the program. So the "When FLAG clicked" block means that the program should not begin executing until we tell it to – sounds reasonable!

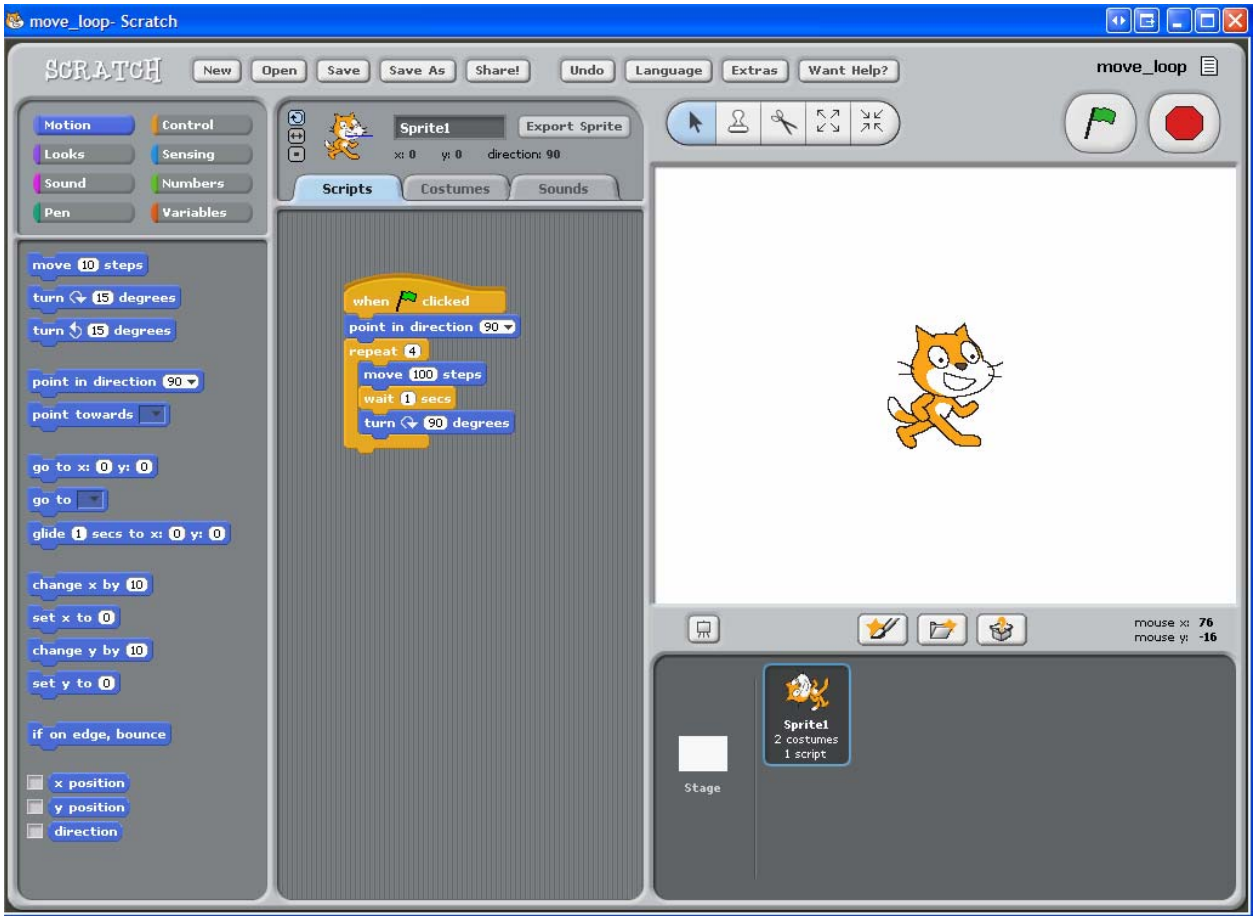
Now we can attach more commands to this block, and they will execute in order. Let's start with the motion tab. We can make the cat "sprite" (another word for "picture") move around the screen. The white sprite screen is actually a graph with an x and y axis, and when you move the mouse around it, you can see the "Mouse X" and "Mouse Y" numbers below change based on where the mouse pointer is on the "graph." The center (0,0) point is called the origin.

Let's add some commands to make the sprite move around the screen. If we tell the sprite to begin by pointing to the right, then move 100 steps, turn 90 degrees, and repeat four times, he will make a perfect square. To link up two blocks, drag the block just below the preceeding block, and you will see a white line which indicates that Scratch will connect the blocks for you when you let go of the mouse. You can also change the numbers in the white ovals by clicking and typing. You will see later how to drag steps inside of these ovals and do complex computations, but this will be another lesson. For now, I have added control items "Wait 1 secs" so that we can see what the sprite does when we run the program. Otherwise he would do all of these steps in a fraction of a second, and he would be done before we knew he had started! The result will look like the figure below.



And this is perfectly acceptable. However, notice that the steps we gave were to "repeat four times." It seemed like a lot of extra work to put all the same "move 100 steps" and "turn 90 degrees" items into the Scratch pad. It would be nice if we could actually tell Scratch to "repeat." We should think that it is possible, because it ultimately translates into ones and zeroes anyway, so these expressions are completely equivalent.

Notice the "repeat 10" block in the "Control" tab. This is exactly what we want. Also notice that it is shaped like a Pac Man or ET, and it can "eat" other steps. The steps we put inside of the repeat structure are the instructions that will be repeated. The number "10" represents how many times we want to repeat. How many times should we repeat our steps to make a square? Four! And how many degrees should we turn? 90, just like before. Our looped approach allows us to write the code we want only once and tell that code to repeat, just like we would expect it to. The result looks like the figure below.



Here again, we begin by pointing to the right of the screen, then moving 100 steps, waiting (for us to see), turning 90 degrees, and repeating four times. This, we know, makes a perfect square – we can see it! But how do we know mathematically that we have made a square? How many degrees does each corner of a square make? 90. And how many corners does it have? 4 – as we discussed. How many total degrees does this make? Count them – 90, 180, 270, 360. What do you know about 360 degrees? This forms a circle. It also forms any shape that ends where you began – just like a square. We will explore this in the next lesson when we talk about tessellations.

Materials

- Enough laptops for the students, groups up to 3 are appropriate.
- The Scratch software environment (see References section).
- A laptop and projector combo with Scratch for presenting to the students.

References http://scratch.mit.edu

Author: William M. Mongan

Date: 12/2/2007