1. Introduction

To learn about computers and how computers process digital versions of information (like digital images) you need to learn the basics of how computers work. Computers are simple machines that have a few, powerful types of action. These basic operations are loops, variables, conditional statements, and boolean expressions. They can also do arithmetic.

To introduce you to how to program computers, we begin with a language called Scratch invented at MIT to teach these important ideas in an easy-to-learn environment. We shall then build on these skills in our web-based slide show using other languages.

2. Getting Scratch

Scratch is free from http://www.scratch.mit.edu. There is a Windows version and a Mac OSX version. Just download the newest version from the site and install it on your machine. The labs in Halligan have it installed. The scratch website has galleries, and anyone can set up an account and publish and share scripts there.

3. Using Scratch

Here is what you see when you start up Scratch:

The animated world is called the Stage and is over on the right part of of the screen. The actors on this stage are called sprites. The instructions (the scripts) these actors follow go in the middle part of the screen. You, the playwright, put the instructions for each sprite in this section. The cast of characters appears in the lower right portion of the screen. Finally, the instructions you can use to write scripts can be found in the left panel. This palette of instructions is organized into categories shown at the top of the left side.

Let us look at some of these instructions, called statements.
The statements are shaped like puzzle pieces. It is impossible to assemble statements incorrectly. How do we make our sprite say something? In a music box, one puts a hole in a card in the right place and a horn blows. Here, we use the statement \texttt{say} \[]. How do we start the script? In one of the music boxes, we dropped in a penny. Here we click on the green arrow. The program has two parts: what starts the script, and what it does.

In the music box, one tells the machine to play the same note several times in row by putting a sequence of holes with gaps between them. In Scratch, we do much the same thing. Here are two examples:

If we wanted to play the sound 100 times, we could stack up 100 pairs of instructions, but there has to be an easier way. There is, and we shall learn about it soon. First, though, we worry about those cows running into the end of the field.

4. Conditional Instructions

Think about those cows hitting the end of the screen. It might be nice to put in an instruction that says something like "If the cow is at the edge of the screen, have it reverse direction." Consider that floor sweeper robot. When it hit a wall or a piece of furniture, it changed direction. We want to be able to add some sort of \textit{conditional instructions} to a script.

Scratch, like most computer languages, includes ways of testing if certain conditions are true. Here are some:
All conditions have this shape with pointy ends. We shall see how to use them in a few minutes. These tests are sometimes called Boolean Conditions, named after the mathematician George Boole who studied ways of working with statements that can be true or false. Being able to ask if a sprite is near a wall or if the mouse pointer is over a sprite means we can write programs that respond to user activity.

How do we do that? We use the if..then.. conditional controls. Here are two versions and a combination.

Notice how this structure reflects the two parts or three parts. If it is raining, take an umbrella. If the cow hits the wall, reverse direction, if you press the change return button on a candy machine, return the change.

Notice the else part. How does that work in English? Notice now the nested conditional control. Give an example of a situation in English that works that way.

Now we can combine Boolean expressions with conditional controls to write some short programs that take action based on a condition.

Note how the second program uses random numbers. These are essential for writing games the roll dice, shuffle cards, pick numbers. These are also essential for simulations of the real world.

Let’s modify hello5 so it plays a different sound if the randomly chosen number is 6 or greater.

5. Loops in Scratch

Computers can follow instructions, they can test conditions and perform different actions depending on the condition, and they can also repeat instructions. In a musical score, the 'repeat' symbol is the notation that instructs the musician to play a section of music again. In computer terms, a repeated set of instructions is called a loop.

Here are three important types of loops:

What sort of things in real life match each of these type of repetition?
Beginning with Scratch

Now we can combine loops with regular statements and also with conditional controls to build more sophisticated scripts. Here are a few:

Hello6.scratch

For each of these, let us predict what the program will do before we run it. Here is another loop combined with conditional execution: Hello8.scratch

6. Variables: Storing Values for Later

Consider a game in which the computer rolls two dice on the screen. The computer has to pick two random numbers between 1 and 6. Then, the computer has to do several things with that those numbers. The computer has to display the numbers for the user to see what was rolled. But, the computer also has to add up the numbers to see what the total is. Depending on the game, the computer may need to remember that total for later.

When you need to remember a word, or a number, or a name, what do you do? I usually write it down somewhere with a little label. I’ll put down a notation like: office_number = 7-3681. That way, I can refer back to the paper later and know what the number means.

Computers can also write down numbers for later use. Just as I made up a label "office_number" and then wrote down the actual value, the computer allows a programmer to make up labels and store a number with that label. Computer languages call these named storage containers variables. Variables, along with loops, and conditional control, are among the most important basic ideas in working with computers.

Computer variables are not a tricky concept. They are like mail boxes in a dorm or apartment building. Each box has a name on it, and each box can contain something. You can put any name you like on a mailbox and you can store anything in it. In computers, the names are regular words and the contents are numbers or words. For example, one can call a variable die1_roll and one can store in it a number like 2. One can then look in that box as often as one likes and see that the box contains the number 2. This box is called a variable, because later you can put a different value in the box, say 5. That is, the value stored in that named box can be varied.

Creating a Variable and Changing It
Creating a variable in Scratch is easy. On the Variables palette, one can make a variable and one can delete a variable. You just pick a name and then you decide whether you want the user to see the variable on the stage.

Let’s make one now and see what we can do with it. The two simple operations are changing the variable by a fixed amount or setting the variable to a new value.

Here is Hello9.scratch showing how we can use a variable to do two things with a number. We first pick a random number between 1 and 10. Then we say the number and after that we check its value to determine if it is less than 6. Since we are using the value in two places, we had to store it somewhere.

Using variables is how computers ‘write it down for later.’

7. Acknowledgements

This introduction to Scratch is based on introductory lectures given by Henry Leitner and used with his permission.