Blocks Programming

Michael Tempel
Logo Foundation
michaelt@media.mit.edu

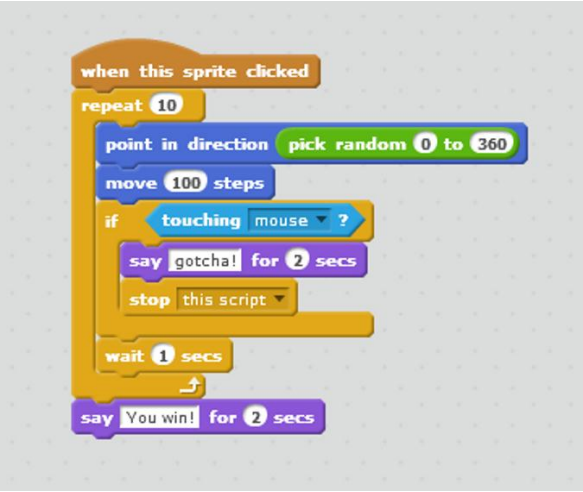
(A version of this article was published in CSTA Voice, Vol. 9 No. 1, March 2013,
http://csta.acm.org/Communications/sub/CSTAVoice_Files/CSTAVoice_03-13.pdf)

The popularity of Scratch over the past six years has encouraged the widespread use of a particular style of visual programming known as blocks programming. Programs are created by snapping together graphical pieces, like putting together a jigsaw puzzle.

The first blocks programming language was Logo Blocks. Developed in 1996 at the MIT Media Lab, this graphical version of Cricket Logo controlled programmable bricks – small microcontrollers that were used in robotics projects. Blocks programming is also found in Turtle Art, PICO Blocks, MIT App Inventor, StarLogo TNG, and Alice.

Here is a quick overview of how blocks programing works using Scratch as the example.

Scratch features sprites – screen objects that may be animated and act as characters in a story or game. For example, we may have a cat sprite and a mouse sprite. This script causes the cat to move randomly around the screen. It may catch the mouse.



The cat sprite points in a random direction and moves 100 steps. It repeats this 10 times unless one of the moves causes it to touch the mouse sprite. In this case, a talk bubble with the word "gotcha!" appears for two seconds and the script stops. If the mouse hasn't been caught after 10 repeats, the cat sprite says "You win!" and the script ends.

The blocks provide a visual representation of the program, which for many people is more understandable than text. A block's shape gives an indication of its purpose. The way in which the blocks are assembled shows the flow of the program.

Most blocks have a notch at the top and a tab at the bottom. These "stack blocks" may be put together, forming a sequence of instructions. The "repeat" block has a "C" shape, with the sequence of instructions to be repeated fitting into the "C". The "if" block has a similar shape, enclosing the instructions to be run if a condition is met.

Some of these blocks, such as the "move" block require inputs. We can click in the number field and type a number. For the "point in direction" block, rather than use a constant as the input, we have used a "reporter block" which returns a random number between 1 and 360. This block has a rounded rectangle shape so as to fit into the number field of the "point in direction" block.

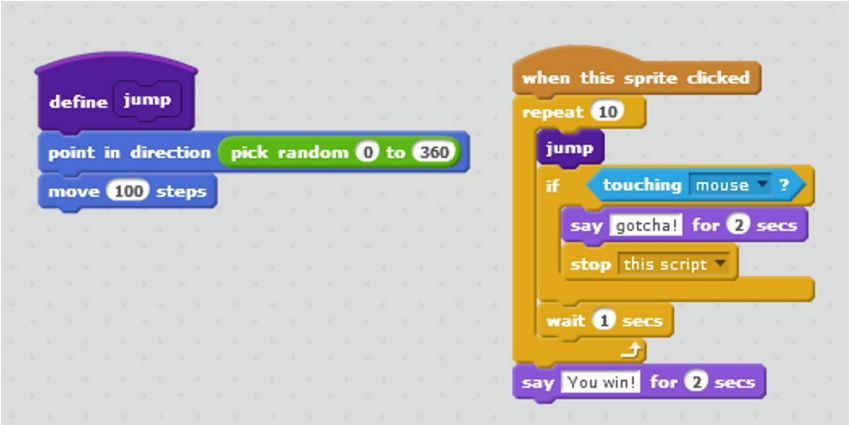
The "if" block needs a Boolean input, which is provided by the "touching?" block. It reports "true" if the cat touches the mouse and "false" if the two sprites are not touching. The hexagonal shape of this block allows it to fit properly into the "if" block.

A beneficial side effect of having the shapes of the blocks be related to their uses is that it is largely impossible to make syntax errors. If we try to put the "point in direction" block into the "if" block, it will not go. It's the wrong shape. Similarly, we cannot put the "touching?" block into the "point in direction" block. In this way, most syntax errors are precluded.

The blocks are also grouped by color. Blocks that move sprites are blue. Control block like "repeat" and "if" are yellow, and so on. Categorization by color provides an additional visual clue to what the various components of a program are doing.

There are limitations to blocks programming. Because of the graphical representation, a large program takes up a lot of screen real estate and can be hard to follow. But as with text-based programming languages, one can create procedures (functions) that allow a body of code to be collected and called by name. In blocks programming, this is done by making a new block[1].

Using our previous example, we can take the code that aims and moves the cat sprite and use it as the body of a procedure named "jump." The newly defined "jump" block can take the place of those two lines of code. This is a simple example, but procedures can be complex, and may include other user-created blocks as sub-procedures. The definition of a block may include the block being defined so as to create a recursive procedure.



In recent years there has been increased discussion of the need for computational thinking and fluency for everyone. At the same time, the number of students majoring in computer science has been dropping. Because of the ease and comfort in getting started, blocks programming makes computing accessible to a much wider audience. And some of those students will become sufficiently engaged with computational ideas to want to pursue computer science careers.

One can go quite far with blocks programming. SNAP, an extended version of Scratch, is being used to teach computer science at the high school and college level, including some of the pilot sites for the new AP Computer Science: Principles course.

Computer science majors will of course also learn conventional text-based programming languages. After an initial experience with computing through blocks programming, the transition will be smooth.

References

There are many blocks programming environments available. Here is a partial list:

1. Scratch software and resources are available at http://scratch.mit.edu. An additional web site for educators is ScratchEd at http://scratched.media.mit.edu/

2. The initial description of Logo Blocks is *LogoBlocks: A Graphical Programming Language for Interacting with the World* by Andrew Begel. It may be found at http://research.microsoft.com/en-us/um/people/abegel/mit/begel-aup.pdf.
   A current version of Logo Blocks that works with the Super Cricket is available at http://handyboard.com/cricket/software/
3. SNAP, also known as Build Your Own Blocks (BYOB) is an extended version of Scratch: http://snap.berkeley.edu/
4. PICO Blocks, which works with the PICO Cricket may be downloaded from http://www.picocricket.com/
5. TurtleArt is a blocks-based art and geometry environment: http://turtleart.org/
6. Scratch for Arduino is an extension of Scratch that interacts with Arduino microcontrollers: http://seaside.citilab.eu/scratch/arduino
7. StarLogo TNG is a blocks version of Star Logo: http://education.mit.edu/projects/starlogo-tng
8. Alice is a 3D animation environment: http://www.alice.org/
9. MIT App Inventor is used to create apps for Android devices: http://appinventor.mit.edu/

---

[1] This feature is in Scratch 2.0 and other blocks-programming environments.