SCRATCH: INTRODUCING KIDS TO PROGRAMMING

Kyle Snowden
ksnowden@email.arizona.edu

Dylan Fox
dylanfox@email.arizona.edu

April 7, 2020

ABSTRACT

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim 2justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt.

1 Introduction

```
main() {
}
```

2 History

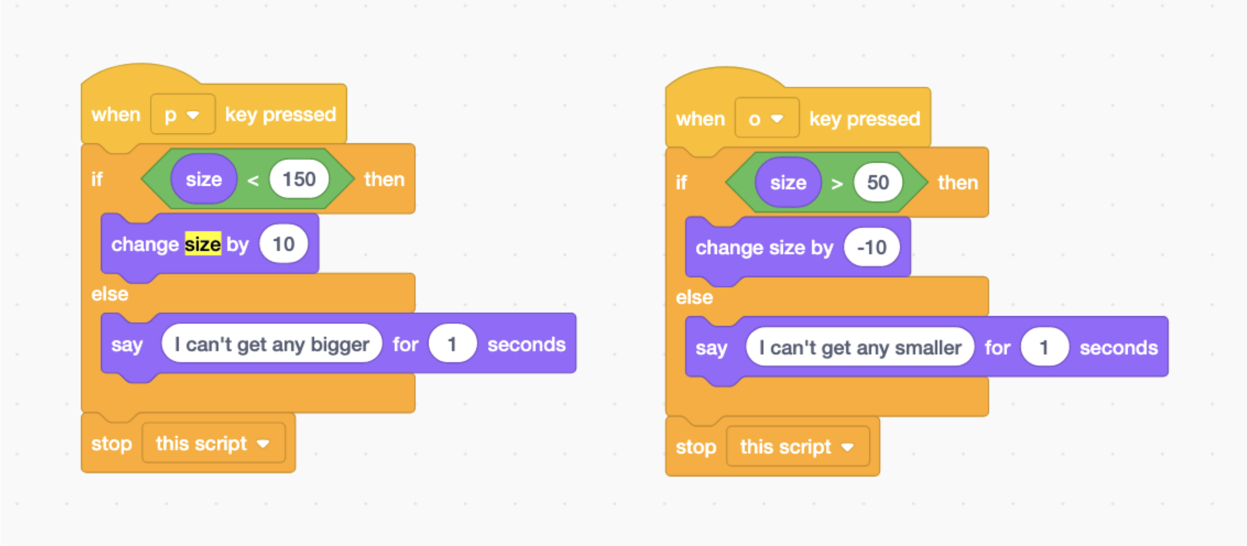
Scratch is not your traditional programming language, it's what's called a block-based visual programming language that was developed by the MIT Media Lab. It's primary purpose is to educate it's users of concepts and skills that can then be applied to other languages such as Java or C. The language first appeared in 2003 with the first desktop version of the language was developed, however it wasn't until 2007 when it was released to the public. The goal of the project was to teach young children to code in an easy fun and interactive way. [1]

Today Scratch is currently on version 3.0, released in 2019, replacing it's predecessor Scratch 2.0 which was released on May 9, 2013. Today Scratch is used in many places across the globe and has been translated into 70+ languages. It is very prevalent in classrooms in all age ranges, scratch was developed in close coordination with a young audience at "Computer Clubhouses to maximize it's ease of use and educational effectiveness.

Scratch aims to simplify creating animations, games, and interactive stories, and simulations. Scratch 3.0 has it's own self contained paint editor and sound editor allowing users to create assets all within one suite. Scratch targets kids within the age range of 8-16 years old, often giving the kids a brief glimpse into Computer Science for the first time.

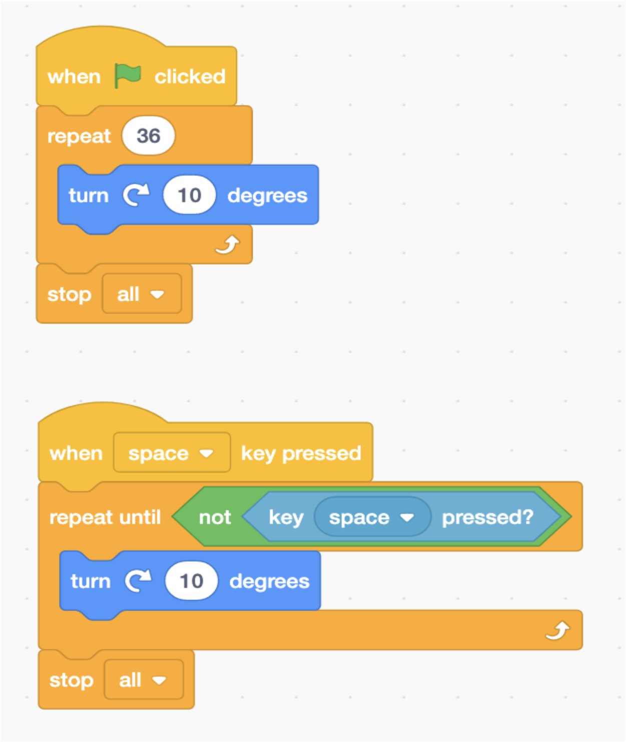
3 Control Structures

The first thing to realize when it comes to control structures is that Scratch is structured around visual programs heavy in user interaction. Since this is the focus, Scratch code is commonly organized into many small 'scripts'. Each of these scripts begin with an 'event hat block', which is essentially a handler that waits until a certain condition is met. Commonly these are key presses, mouse interactions, and introductions of new sprites or backgrounds to the screen. The following example highlights these hat blocks, as well as the commonly used 'if-then-else' block:



In this example, the scripts respond to key presses by increasing or decreasing the size of the sprite on the screen. The size of the sprite is capped and if it would be changed beyond these caps, instead there is a visual cue letting the user know the limit has been reached.

Another common family of control blocks are the 'forever' and 'repeat' blocks, which essentially function like while loops.

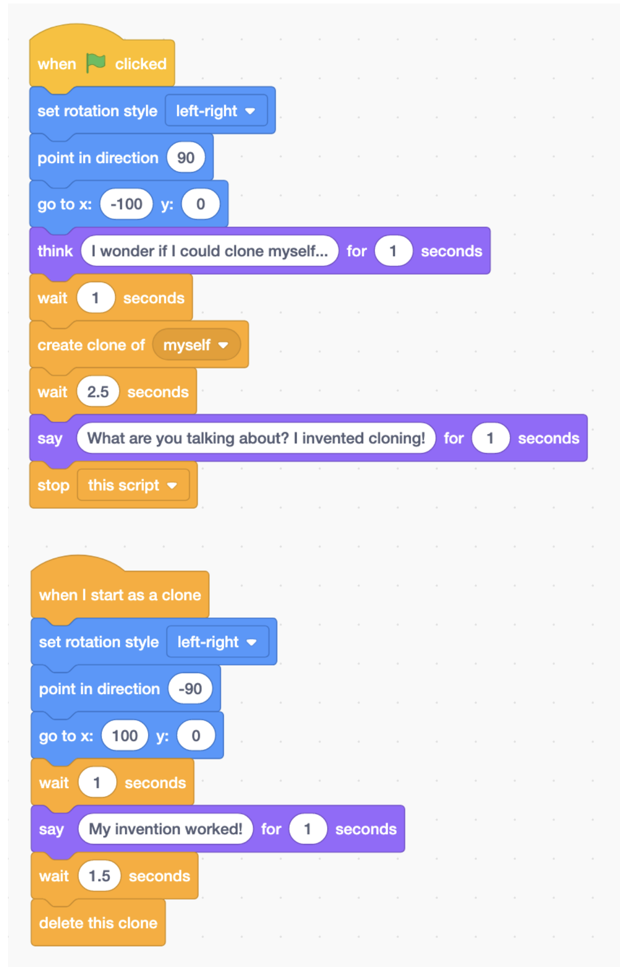


In the example to the left, both handlers begin with user input. The top script begins when user clicks on the flag—the typical way to start the program—at which point a 'repeat' block will turn the sprite around clockwise. In the bottom script, the sprite will rotate clockwise while the spacebar is pressed. It is important to note that since Scratch is meant to be very user-friendly, it often hides complexity. In both of these cases, the 'turn 10 degrees' command takes an amount of time to execute, creating a smooth rotation.

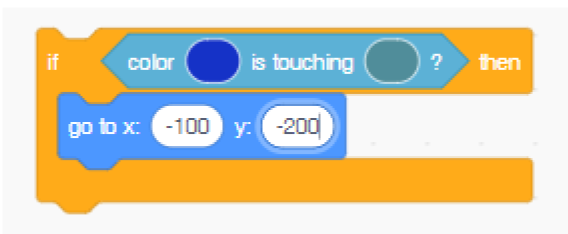
Another block in this family is the 'forever' loop that will not escape until the 'stop' block is called. This is functionally the same as the 'repeat until <boolean>' block seen the left.

Another interesting control structure is the sprite cloning capability of Scratch. Scratch has extensive support for 'sprites', which are bitmap graphics used along with 'backgrounds' in the construction of a scene. Sprites can be either static or dynamic, and there are unique features which relate to sprites. One of these features is showcased in the following code snippet, which clones a sprite.

In the program to the right, the program starts when the user clicks on the green flag. After some commands which position the sprite correctly (the commands in blue) the original sprite visually displays the text "I wonder if I could clone myself...". The real magic of cloning happens at the command 'create clone of myself', at which point the second block begins executing. In Scratch, 'cloning' copies only the visual aspect of a sprite, but executes a different script. Continuing this example, the clone says "My invention worked!", after which the original sprite responds, "What are you talking about? I invented cloning!" Both sprites are coordinated so that neither is speaking at the same time.



There are further control features which stem of Scratch's visual focus. One example of this is a sprite's ability to detect if it is touching another sprite, background, or a specific color. In the example below, the boolean expression 'color <color1> touching <color2>?' is used as part of an 'if-then' control block.



4 Data Types

5 Subprograms

6 Summary

References

[1] Scratchers. Scratch wiki. https://en.scratch-wiki.info/wiki/, 12 2008.