SCRATCHING THE SURFACE

Cobi Nguyen
cnguyen32@email.arizona.edu

Zachary Wiegand
zwiegand@email.arizona.edu

April 20, 2020

ABSTRACT

1 Introduction

2 History

Scratch is a visual block-based programming language with the goal of making programming more accessible to children. Scratch gets its name from DJs and their technique of combining different vinyl records called "scratching". Scratch is developed by the MIT Media Lab with the first prototype appearing in 2002 and officially launching on January 8, 2007. Currently on version 3.0 scratch is implemented using JavaScript while versions used Squeak and ActionScript. Currently Scratch is alive and well with over 40 million website visits, as seen from the analytical data on their website, and version 3.0, the newest version, coming out on January 2, 2019.

3 Control Structures

Scratch makes learning an accessing its control structures are fairly simple because Scratch was created to introduce people into the world of coding. On the left side of the web page where the different sections of code blocks can be found, there is a whole section dedicated to control structures. Within this section the developer can find different forms of iteration, conditions, wait statements, and even an interesting cloning mechanic.

In Scratch there are three given version of iteration for the programmer to use not including recursion. These three versions of iteration are the repeat loop, the repeat until, and the forever loop. While having different names they are equivalent to control structures that an experienced coder would have already came into contact with. The repeat loop is equivalent to a for loop so it repeats its code block a certain number of times. The repeat until is effectively a while loop which means it will repeat until a conditional is satisfied. Finally, the forever loop does exactly what it sounds like; the forever loop will repeat its code block indefinitely or until the code is stopped somewhere else.

This is also a great opportunity to mention options to stop code being run early. Scratch gives programmers a 'stop' function. The stop function can specify if **all** code needs to halt, that single block of code is to stop running, or if all the code exclusive to that sprite should stop. This function gives further control to the developer by allowing them to decide when a script stops running. Further control of when code should be ran comes from two provided wait functions. the first wait function acts like a sleep, it pauses the script at that statement for the specified number of seconds. The second wait function has a little more complexity. The programmer can pause the scrip in that sprite at the wait statement until the specified condition is met. This second wait statement can come into use if a programmer wants an object o pause until interacted with by the user or another object.

Within the control section the developer also has the standard conditional statements that an experienced coder would come to expect. Those conditionals are if statements and if else statements. One might notice there is no if else-if statements which means if a developer wants to create branching conditional checks they would have to create a somewhat robust version of a switch case. That being said Scratch also does not have a formal version of switch cases so a developer would have to use multiple if blocks in succession and make sure that if one of those if blocks gets ran that the others are also not ran.

A unique type of control structure for Scratch is three functions for cloning sprites. The first cloning function is 'create clone (sprite)'. This is what is called to create the clone of either the sprite calling the function, or create the clone of another sprite. The second function is 'when I start as a clone'. Say a clone has just been made of sprite A, in sprite A code following this second function will execute. After a clone has finished doing its job, the program may want the clone to go away, that is what the last clone function is for. The third clone function given to programmers is 'delete this clone'. This function simply removes that clone completely from the program.

A great example of cloning can be found within the creative program for milestone 2, '99 Bottles'. The way the program was designed the end goal was to have 99 soda bottle sprites on the screen that could run code when clicked on by the user. It would have been unreasonable to create 99 individual sprite objects manually. Therefore one single soda bottle sprite was made and then was recursively cloned to fill the screen with 99 bottles.

4 Data Types

Due to its nature of accessibility to new programmers Scratch does not have any data types that seem out of the ordinary to the experienced programmer. The three data types that Scratch has are numbers, strings, and booleans.

Numbers in Scratch refers to both integers and decimal values. Scratch does not see a difference between the two. There are ways to manipulate number variables within Scratch such as flooring, converting a number to the nearest whole number lower than the decimal, ceiling, converting a number to the nearest whole number larger than the decimal, and even taking the square root of your number variable.

String is a series of characters of letters, numbers, characters, and basically everything you can type. Strings are useful for holding any information that is readable for humans.

Booleans are a variable type that only allows two values, true and false. By using booleans a programmer can test conditions and then proceed to tell the program what to do based on the result of those tests.

Even though that the amount of different data types is fairly short the variables within Scratch also have different types. The user has access to global, local, and cloud variables. By default all variables are created as global and are able to be to accessed by any part of the program. Local variables are more personable in the way that only their owner is allowed to change them. However, other pieces of the program may be able to look at the local variable. Lastly, cloud variables are variables stored on the cloud which allows the variable to affect multiple instances of an open project and gets saved when the project is closed.

5 Subprograms

Subprograms within Computer Science are smaller blocks of code that perform a specific task within a larger project. Within Scratch these subprograms are easily represented by the code blocks given to the programmer. Each code block is basically its own subprogram, such as, the repeat block which repeats the code within itself as many times as the developer wants the code to repeat. These subprograms are often expanded upon because it takes a combination of these code blocks to form something recognizable as a subprogram.

The code blocks that Scratch gives its developers are easily categorized into 10 different sections. This effectively gives the user or programmer 10 types of subprograms to play with and join together. The 10 different categories of code blocks that Scratch has are Motion blocks, Events blocks, Looks blocks, Control blocks, Sound blocks, Sensing blocks, Pen blocks, Operator blocks, Variables blocks, List blocks, which are within Variable blocks, and My blocks.

Motion blocks affect anything to do with the movement of sprites. Since Scratch primarily relies on the use of sprites to perform actions within a project motion blocks become useful in the sense that they give the sprites the ability to move around the stage. These motion blocks seem to only effect these sprites on an x and y axis but a programmer is given the ability to rotate the sprite using these motion blocks.

Event blocks are used when the programmer wants another subprogram to run when a specific event happens within the project. An example being when the space bar is pressed on a user's keyboard it causes the sprite to jump. Event blocks are useful because they allow pieces of the program to run at specific moments and event blocks allow user input.

Look blocks relate to anything about the looks and appearance of the stage and sprites. With Look blocks the developer is able to simulate a sprite changing outfits or even having the backdrop the stage change to a different picture. Look blocks can even make a sprite look like it is talking using speech bubbles. While Look blocks do not seem to add much to the inner workings of a project it gives a developer an extra step in customizing how they want their program to look.

Control blocks work with the basic flow of a project. Control blocks give developers access to simple structures that are common in coding such as loops and if statements. This category of code blocks is aptly named because everything within this category controls how to program runs.

Sound blocks are fairly straightforward in the sense that they either play MIDI or saved sound files. However Sound blocks do give the developer a lot of customization on how to manipulate sounds. A developer is given the ability to change the pitch, making the sound higher or lower, the pan, making the sound seem like it is coming from the left or right, and the volume of a sound within the project.

Sensing blocks are similar to Event blocks in the way that code is ran when specific things happen. However, Sensing block are different by going off of sprite and stage conditions. An example of this is if two sprites touch each other it can be detected by a Sensing block. These sensing blocks are useful in stabilizing a project's flow because they consist of Booleans that work well with Control blocks.

Pen blocks are very interesting because they give sprites the ability to draw on the stage within a project. Pen blocks also have a large degree of customization allowing the developer to determine pen size and color, the ability to erase all marks, and even putting the pen down or up. This means that you can draw shapes and lines from a sprite to give a sense of a trail or even create a graph of any shape or size.

Operator blocks deal with mathematical and logical processes within a project. Within the Operator category a developer is given the ability to modify strings, do addition and subtraction, but even have a random number generated. Operator blocks are especially useful because of the type of manipulation of data that they give to the developer.

Variable blocks have two subcategories, Variables and Lists, they are grouped together because they both store and access data. Variable blocks are mostly talked about in the Data Types section of this paper, but to simply put them they are changeable values that are either a Number, String, or Boolean. List blocks allow the ability to store multiple pieces of information at once in a specific order. This gives the developer the ability to create something like an inventory in which a player has a series of items that they can use.

My blocks allow custom procedures for sprites. My blocks are particularly useful for running scripts without screen refresh. At the end of the day My blocks are highly customize in the sense that the limit of a My block is the limit of a developer's imagination.

6 Summary

References

[1] Al Sweigart. In *Scratch Programming Playground: Learn to Program by Making Cool Games*. No Starch Press, 2016.

[2] MIT Scratch Team. Scratch 3.0. `https://en.scratch-wiki.info/wiki/Scratch_3.0`. Accessed: 2020-3-26.

[3] MIT Scratch Team. Scratch for developers. `https://scratch.mit.edu/developers`. Accessed: 2020-3-26.

[1] [3] [2]