# Making a Game from Scratch, with Scratch!

## INTRODUCTION

Beginning to learn and familiarize yourself with coding at a young age is beneficial and important as coding is in high demand, it encourages and improves many skills such as problem-solving, logical thinking, creativity, and persistence, and it has many different applications in both jobs and creative projects. The perfect program to start with is Scratch. Scratch is a basic platform for aspiring programmers that is visual, centered on blocks, based on the coding script javascript, and developed by the MIT lab. Overall the way you should get into coding is by creating the projects you love, and for me, that is video games. With that, I will take a look at the process of making a game in Scratch!

## GETTING STARTED

The first step to creating a game is starting with a general idea or concept of what you want to create. This does not have to be a full-fledged plan at first, but it should be something that you can build off of when deciding different aspects of the game. Some general ideas may be an objective for the game, the main character, or a setting. It should be something that inspires you.

According to Juni Learning, a program that offers online classes for kids, “making a plan is the first and most important step towards making a game,” which should consist of figuring out a visual theme, what a player controls, and an objective (“How to Make a Game on Scratch Step-by-Step for Beginners”). This planning allows you to keep a goal in mind, and for a more organized process overall. An example of what your player does or controls, which should be planned out, is controlling a player’s movement or pressing objects as they fly by. Another important thing to decide is the objective of the game which could be getting to the end of all levels, collecting as many of an object as possible, or tricking all of your friends into believing you’re not the imposter. Lastly, it is helpful to decide a visual theme, which could determine different physics in the game. Moving underwater would be different than moving on land or in outer space, for example.

After brainstorming and choosing the game concept you are most excited to create (I recommend taking note of it in a google document as I did with my project), it is important to familiarize yourself with Scratch. With coding, there are no defined steps with a project you create. It

takes understanding the different blocks, being intentional about what you want to happen, being able to use outside resources, using logic, and practicing.

The only thing needed for scratch is a computer, good internet, and access to the web. The first thing you should do when you open up scratch.mit.edu, the Scratch official site, is press the “Join Scratch” button top right to create a Scratch account. Following this, you can go to the “Ideas” section (top middle button), click “Choose a Tutorial” and get started teaching yourself how to use the platform. If you just want to jump into it on your own, a suggestion is to go to the explore page to see a few people’s projects similar to the type of game you want to create. After clicking on a game, you can press the “see inside” button top right, to look at the code for the game. Try to play the game and figure out what the code is doing. You can see the coding of different sprites in the game (sprites are computer graphics that you can move via code including characters, objects, and more) by clicking the different sprites at the bottom right corner of the screen. At the top left of the screen are the three categories “code, costumes, and sounds.” You can see the different things that were coded with “costumes” and “sounds” and how they were coded with “code”.

When you are ready to start your project, press "create" near the top left of the screen. When you open it, you will see a cat on the screen with a white background. Unless you want the cat to be your character, go to where you see the cat in the bottom right of your screen, where all sprites in your game will be shown, and click the trash can. Once the cat is deleted, you should start creating or choosing your sprites, as you need things to be controlled before you start coding. These sprites should include characters, enemies, what the character is moving on, and anything else on the stage! To choose a character go to the bottom right screen and hover over the cat icon. There you will see the options to upload a sprite, surprise - a random sprite, paint - create your own, or choose a sprite. Choose the option according to if you want to make a character, use a character in their library of sprites, or upload a character from your files. I wouldn’t recommend using the surprise option, as a random character would most likely stray from the theme. Picking backgrounds is a similar idea, yet for this, hover over the image icon next to the cat.

## USING BLOCKS (BASIC EXAMPLE - MOVING LEFT)

After creating or choosing a character, you should familiarize yourself with **blocks and code your sprites**! Blocks are the “chunks” or “pieces” of instructions you are putting together to tell the computer what to do and can be found on the left. It is crucial to research and understand how to use

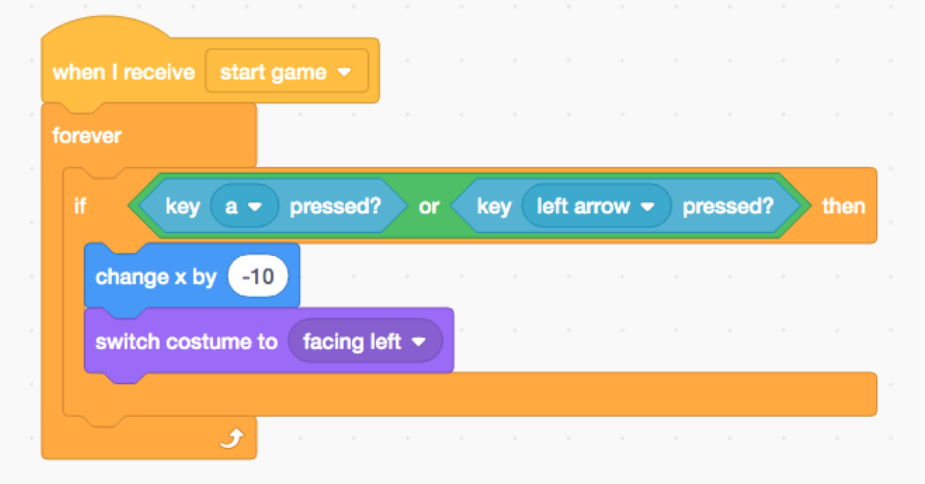
blocks, as they each play an important role in the code. I recommend this site which lists and explains what all 119 blocks do, defines the category of blocks and goes over the different block shapes: https://en.scratch-wiki.info/wiki/Blocks. Different blocks have different functions such as **variables**, which you can create, that hold number values and can be changed or referred to. Another category of blocks is **motion blocks** which determine a character's position and movement based on its x and y coordinates. Scratch relies on an x y coordinate system as if it is a graph. For example, there are "go to x: __ y: ___" blocks, in which you input x and y coordinates for your character to go to. Using the example of moving left, the motion block used would be "change x by __" where the blank is a negative number, such as -10, because moving negative spaces on the x-axis would be moving left.

It is necessary to code how the player will cue the **scripts** (sets of steps, in this case, blocks, for a computer to follow). For example, have the key "a" set in mind for when the character moves left, so you can code that when the key "a" is pressed, the character will move left. This leads to conditionals, which are how the movement is coded. Conditional blocks, under the "Control Blocks" category, have a diamond-shaped hole in them to be filled by sensing blocks. The script would be "If <key a is pressed> then [change x by -10]". With this, you have already used a control (if... then), motion (change x by __), and sensing block, (key a is pressed) however the code will not work, because an **event block** is needed. The event block tells the program when the whole script is cued. If you want the player to be able to move when the game starts, use the event block "When [Green Flag Icon] clicked" before the script. This is an important block, as it's how the first script in the code for the whole game needs to start, as the game starts when the green flag is clicked. However, if there is a title animation playing, for example, you might not want the movement to be controlled by the player right away. One thing you could use is a **messaging** block. As explained by the Create and Learn team, "Messaging blocks allow your sprites to communicate with each other in code. Messaging is used to start running code in other sprites" ("Advanced Coding with Scratch"). Once the sprite with the title animation is finished, it can broadcast the message "start game." The main character can then use this as a control block so the script would start with "When 'start game' is broadcasted."

The use of **operator blocks** comes in hand when you want to perform mathematical functions or use multiple sensing blocks at once. Currently, with the move left script, the character would only move left if the key "a" is pressed, but if you wanted it to make it so it would move when either the left arrow key or the "a" key was pressed, you would use the operator < > or < > where sensing blocks fill the diagonals. Thus you would have "If <key a> or <key left arrow> is pressed..."

Next, it is important to think of how different things in the game will affect another. Multiple things may change when a or the left arrow key is pressed. You may want the character to face a different direction when moving left, which can be used with a looks block if you had one costume facing left and another right. Within the if...then control block, you would put the looks block "switch costume to ___" then enter the specifically named costume, in this example "facing left."

Finally, make sure you have the forever block, a control block, surrounding the code after the event block which will ensure that the code continues to run instead of running only at the instant of receiving the "start game" message. With that, a basic script is finished with the use of event, control, looks, operator, sensing, and motion blocks! Here's how it would look in block form:




As you mess around more with coding, you will get more familiar with how things work, and be able to apply the different blocks to make your game do so much more than allowing your character to move left. The rest of the coding is up to you. You can also add music or sound effects with sound blocks, create your own blocks to save space in a script, and any other features you want to add!

## TROUBLESHOOTING

One important thing to do when making a project is to keep test running the project over and over because though something may work one time, it does not mean it will work every time. Try changing the way you play or the buttons you press each time, to test all aspects of the code. If an issue occurs, you will need to troubleshoot the issue. Troubleshooting involves identifying the

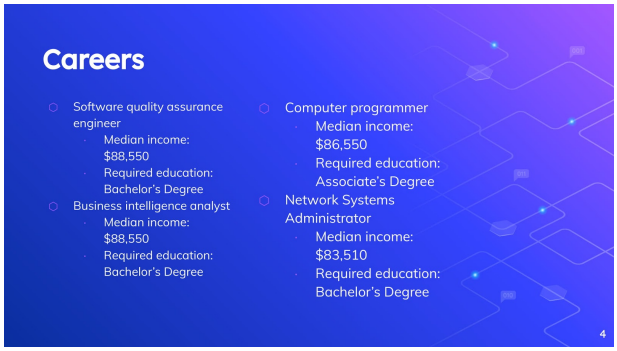
problem from the outside, figuring out the corresponding code on the inside, and tweaking the code to fix the issue. Figuring out how to tweak it may take multiple tries, but it is important to analyze all parts of the code, as it is sometimes not something you would expect to be the problem. One thing to check is the order of the blocks. Scratch uses **sequential coding** in which blocks are put together and run after another. If you want two things to occur simultaneously such as saying something while moving, use a **parallel script** when separate chunks of code are created to run simultaneously.

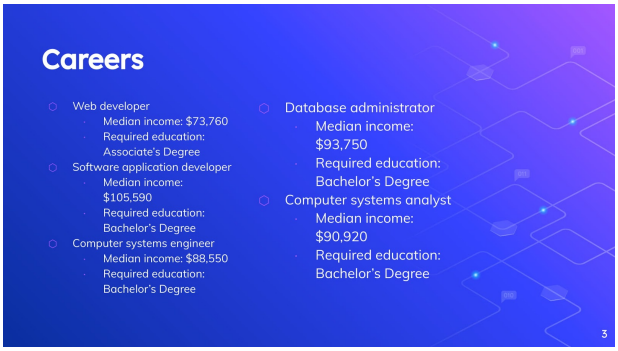
Along with this, using online resources are very helpful. There are also different scripts online that others have made that you can copy into your project if you get stuck. An issue that often arises is the program running slow. It is optimal to only have the one program running that is being used.

## CODING APPLICATIONS & IMPORTANCE

"You can find relevant uses for programming in almost every field you can think of," said Znala Willams, a Molecular, Cell, and Developmental Biology major who uses coding for research (Trang-Kwina). I had met Znala Williams upon preparation for a youth conference in which we presented on coding together. In this presentation, she described some of the applications of coding including those in the following pictures.

Coding Applications from a Slideshow on Coding Znala and I made




Prior to this presentation, I had interviewed Znala on coding in which I asked why learning to code was important and she replied, "I think it's really important for people to understand how to code, especially as the world relies more and more on technical innovations." She went on to say how it's important that there are more coders of coder, specifically, so that people of color can be better and properly represented in technology, as code isn't always inclusive towards people of color. Learning to code in general is valuable as it teaches the learner many tools, can be applied to a

number of different jobs and projects, and it is fun! Whether starting on Scratch, learning on Codecademy, or jumping into coding on Unity, I would encourage anyone reading this to try coding for themselves. Not just creating games, but making anything you want to make, because coding is a tool for you.

## Resources

1. Learn about Blocks in Scratch: https://en.scratch-wiki.info/wiki/Blocks
2. Get Started on Scratch: https://scratch.mit.edu/
3. Get Started on Codecademy: https://www.codecademy.com/