Scratch Lesson Plan

Scratch is a powerful tool that lets you learn the basics of coding by using easy, snap-together sections of code. It's completely free to use, and all the games made with scratch are posted online as a reference for beginner programmers. In order for you to use Scratch, you will need to make an account by clicking the "Join Scratch" button on the top right of the main page (https://scratch.mit.edu/). Any work done on Scratch without an account will not be saved.

In this example we will be constructing a basic game where the cat moves around the screen, collects a snack, and then goes to the carpet to take a nap. The three goals can be whichever objects you want, but for this lesson they will be called Cat, Snack, and Carpet.

Part One: Structure

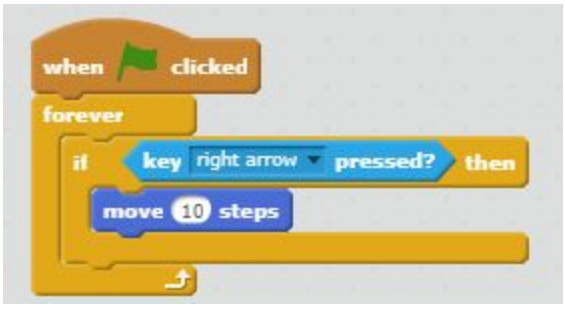
- On the top left of the screen, there is a **Preview window**. This is where you will play and test your game, as well as position your objects. There's a **green Flag and a red Stop** icon on the top right of this window that will start and stop your game. Try to remember to **stop your game before you make changes to your code.**
- Beneath the preview window is the **Sprite window**, where you will be creating and editing your characters, objects, and backgrounds.
- **Sprites** are the main object used to create a game. You'll already see one on your screen when you first open Scratch; a Cat. We'll be using this Cat, and his multiple costumes, to make a basic game.
- There are 3 tabs at the top of the window:
    - **Scripts**: Where you will edit your code. This is the main area of Scratch. You can select sections of code from the different tabs and click and drag them into the empty working area on the right. Different colours and shapes are used to show how the bits of code fit together, which we will go over in the lessons.
    - **Costumes**: Where you will edit your animations. Costumes are what Scratch calls animation frames. They're basically different outfits the sprite puts on to look like he is walking, jumping, flying, or anything else you want them to do.
    - **Sounds**: Scratch has a basic sound editor which you can use to give your game sound effects and background music. Feel free to play around with this after you complete the lessons.
- **You never need to worry about saving your progress in Scratch, since it automatically saves as you work.** Just make sure you're **logged in** to your Scratch account, and everything will be stored in your online library.

Part Two: Movement

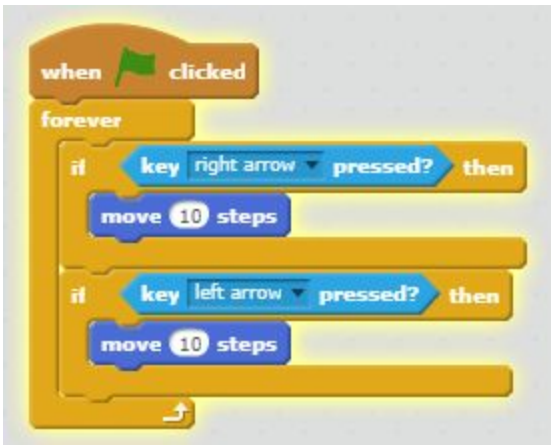
- The very first thing you need is an **initialization statement.** Make sure you have your **Cat** sprite selected, since he is the object that we want to move. Under the **events tab**, you'll find the statement pictured below. Everything that you want to happen when the game first starts should be placed under this event.
    - 

- Next, we want to get basic movement, starting with **moving to the right**. We want to create a statement that says: "If the right arrow key is pressed, move right." The "10 steps" determines how fast the character will move. We will keep it at 10 steps for this example.

- Something is weird, right? **Nothing happened!** This is because the code is starting, running once, and then stopping. We want it to keep running, and running, and running until we say that the game is over. So we will simply put all the code that we want to happen repeatedly inside a **Forever Loop.**



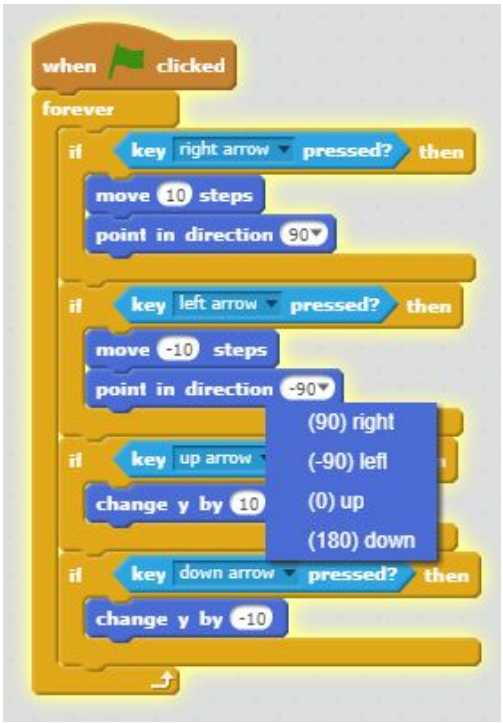
- Now the code should run forever, or until you tell it to stop. Now, let's do the same to move to the left. You can **copy the whole if statement** by right clicking and selecting duplicate.



- So everything is put together correctly, but he still moves right when you press the left arrow key. These move statements make your character move along the **X axis**, so "**move 10 steps" really means "move +10 steps".** So to make him move left we need him to **"move -10 steps".**

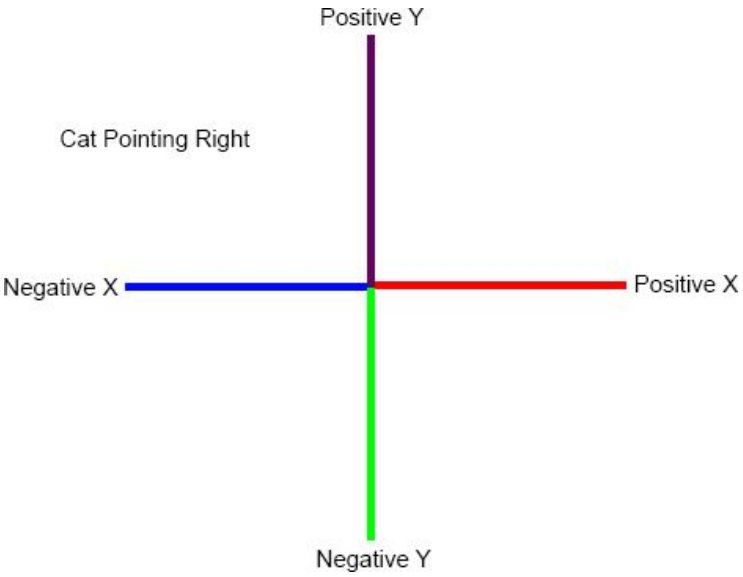
- Now we use a similar statement to make the Cat move up and down. **To go up**, we want to make him go up on the **Y axis instead of the X axis.** Copy your if statement again, once for up and once for down, and make them say "If (button pressed) change Y by (+10 or -10).

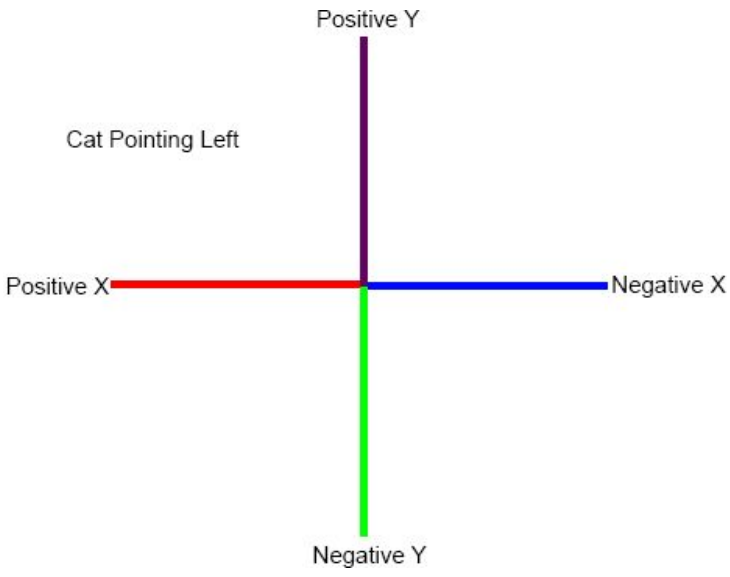
- Now he should be moving in all directions! Next, let's make him look in the direction the he's walking. We can do this using the **"point" functions** under the **Motion tab**. If you click the **dropdown menu on the point function**, it indicates which degree equals which direction. Go ahead and place a point function under your move left and move right if statements. In this game, the Cat will only ever face left and right, so there is no need to put a point function under the up and down movement functions.



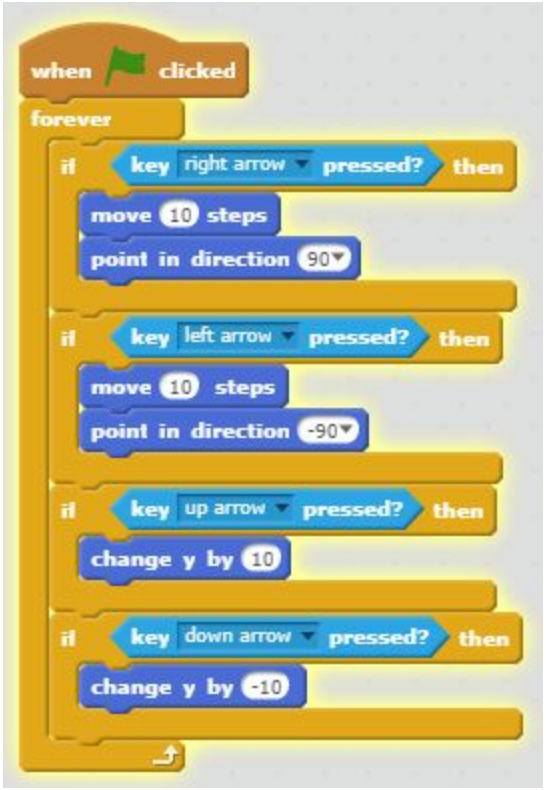
- Again, things aren't working as expected! Two things are not working. First, the **cat is turning upside-down** when certain buttons are pressed, and second, the cat goes right even when the left arrow is pressed. We'll address the upside down problem first. Simply press the little **"i" beside the character** you want to edit to enter the Sprite Settings, and change the rotation style to the little left/right arrow. This will make it so the character can only point left or right, and not 360 degrees.

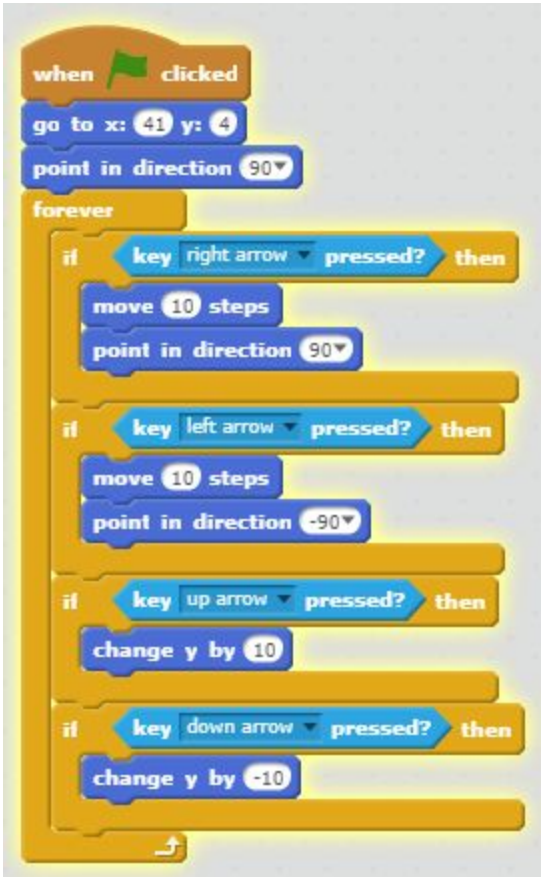
- The next issue is easy to fix, but difficult to understand. The reason the Cat is always moving right is because when we told him to point a different direction, we flipped the X-Axis that he was moving along.





- As you can see from the diagrams, when he is pointing right, if we want him to go right, he has to go towards Positive X. When the diagram is flipped (when the Cat is pointing left) if we want him to go right, he still has to go towards “Positive X”, because it’s now on the other side.



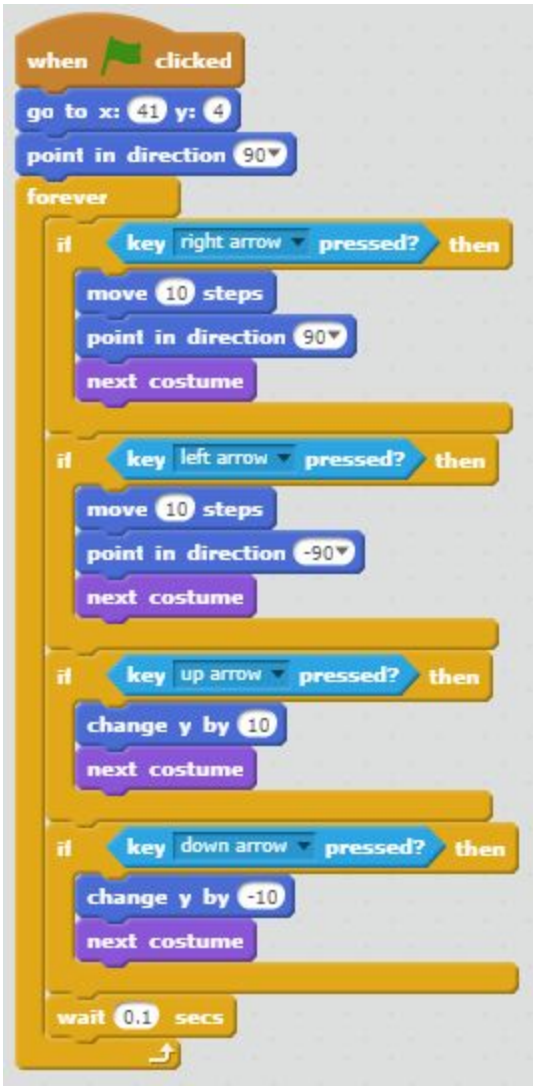
- During this whole process, every time you've started your code to test it, your **cat has been in a different position**. There's an easy way to fix this. Basically, we want to make a function that tells the sprite to reset to it's original position at the very beginning of the code. Simply place the cat where you want him to appear when the game starts, and then add a "go to x:__ y:__" function from under the Motion tab. The X and Y in the blanks are the current position of the Cat, but you can change them later if you ever need to alter the position. Since we only want this to happen once, make sure you put it outside of the forever loop, but after the game start code. You can also add a point function in the same location if you want him to be facing a certain way when the game starts.
    - 

- There! Now your Cat should:
    - Move in all four directions
    - Should change the direction he's facing when the corresponding buttons are pressed
    - Return to the starting position
- This is all the basic movement you need for a simple game, but we want to make the game appear more visually interesting, so next we want to add animations.

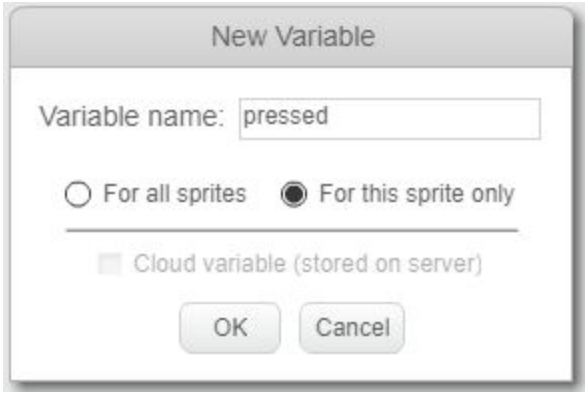
Part Three: Costume Changing

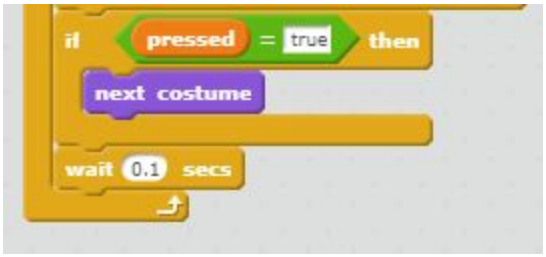
- Animations in Scratch are handled by switching a Sprite's **"Costumes"**. Under the **Costumes tab** above the code tabs, you can see all of a Sprite's costumes. Make sure you have the Cat Sprite selected on the bottom right Sprite menu. The default Cat sprite should come with two costumes, each with their feet in a different position. Basically, we **want to make a string of code that will tell the program to switch between the two costumes** whenever an arrow key is pressed to simulate the action of walking.
  - 

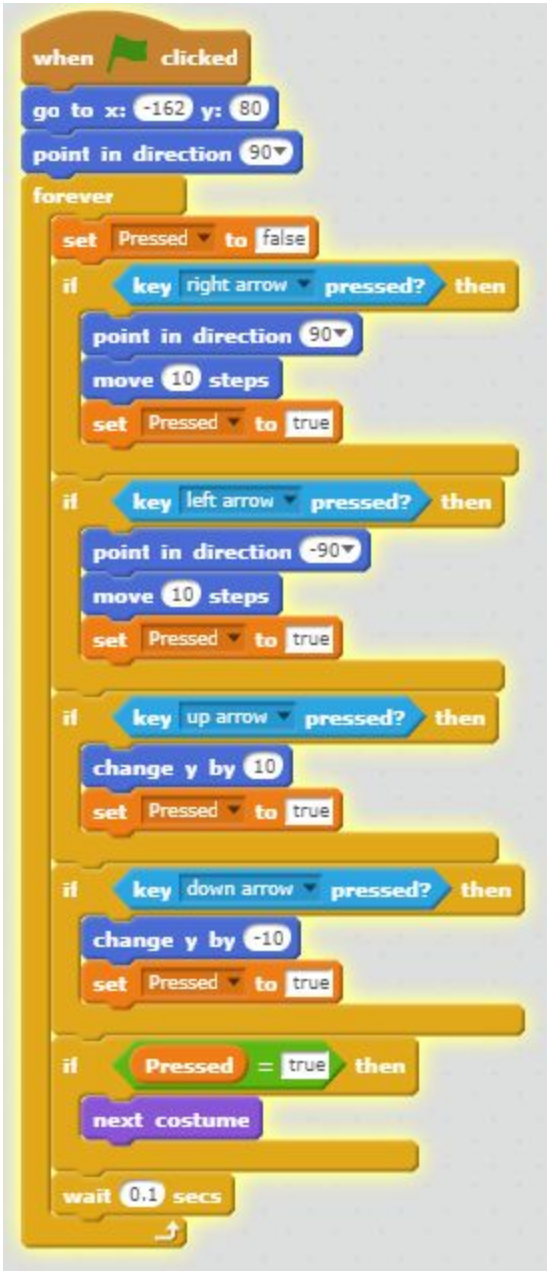
- Notice how fast his legs are moving? Doesn't look quite right, does it? This is happening because the code we've laid out is running as fast as the computer can run without any delay. This is easily resolved by adding a "wait function" at the end of the movement code, but inside the forever loop. This will tell the program to check if any buttons are being pressed, wait a moment, then check again. The result will be a steady walking pace for the Cat.

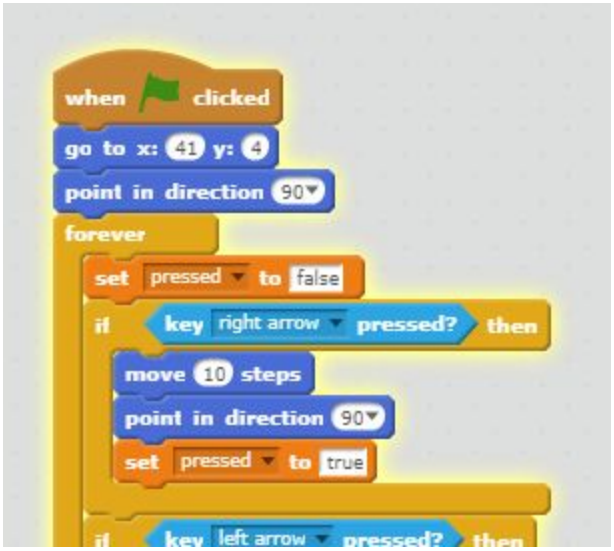


- The above code will work, but it won't work when the Cat moves diagonally. You could make a whole new set of functions for each combination of diagonals, but that will get messy. This is a good opportunity to introduce a variable. **Variables** clean up code, and allow a whole bunch of functions to be simplified into one string of code. Basically, we want to make a variable called "pressed" that detects when a variable is pressed, and changes the costume accordingly
- First, we have to **"make a variable"** which is called declaring a variable in programming terms. Go under the "Data" tab and click "Make a Variable". There are two types of variables, Local and Global. **Local** means it will only affect a certain set of code, in this case it's only the code under the Cat sprite. **Global** means that the variable can be used anywhere in the code, like in other scenes, sprites, and actions. We only want this pressed code to work for this one Cat sprite, so make a variable called "pressed" and check the "For this sprite only" circle.



- When you click okay, a few new options will appear under the “**Data” tab**. Now, we want to tell the “pressed” variable what we want it to do. We need to make a function that says “if pressed variable is true, change to the next costume”. We’ll put it near the bottom of the code, but still in the forever loop because we want it to be checked repeatedly. You can find the green “=” under the **“Operators” tab,** which has a bunch of functions that are used to compare bits of code. Make very sure that when you type “true” into the blank, it’s all lower case, and there are no extra spaces. If it’s not done, just right, the code won’t work.
    - 

- This bit of code won’t do anything yet. All we’ve done is tell it **what** to do, we haven’t told is **when** to do it. So, when do we want the costume to change? When an arrow key is being pressed. When is the arrow key being pressed? In the movement functions we made earlier! To do this, simply put a “set Pressed to true” function into each of your movement functions.
    - 

- Now this code is checking if a button is pressed, if a button is pressed it **sets the "pressed" variable to true**, and the code at the bottom is detecting that pressed is now true, so it's changing the costume.
- There's now one new problem. The cat keeps moving even when there are no arrow keys pressed. This is because there's code telling him to start moving, but **no code to tell him when to stop moving**. We just need to put in a quick bit of code that says "set pressed to false" that runs at the very beginning of the game.
  - 

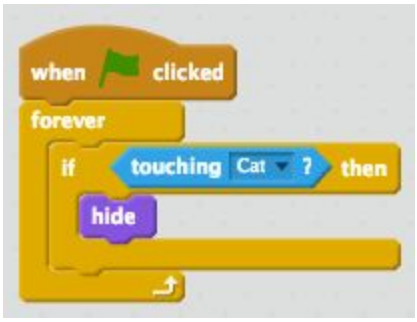
- That's all there is to animation in Scratch. You can use similar techniques to make all your sprites animate. Scratch has a built in costume editor if you're interested in making your own characters, but it's a bit awkward to use. You can always make something with photoshop, or whichever program you're comfortable with, and then upload the new costumes. Now it's time to give the Cat something to pick up.

Part Four: Object Detection

- Now it's time to create a Snack for the Cat by creating a new sprite. This can be done in a few different ways. All four ways are found on the menu (displayed below) that is found above your sprite selection menu.
  - 

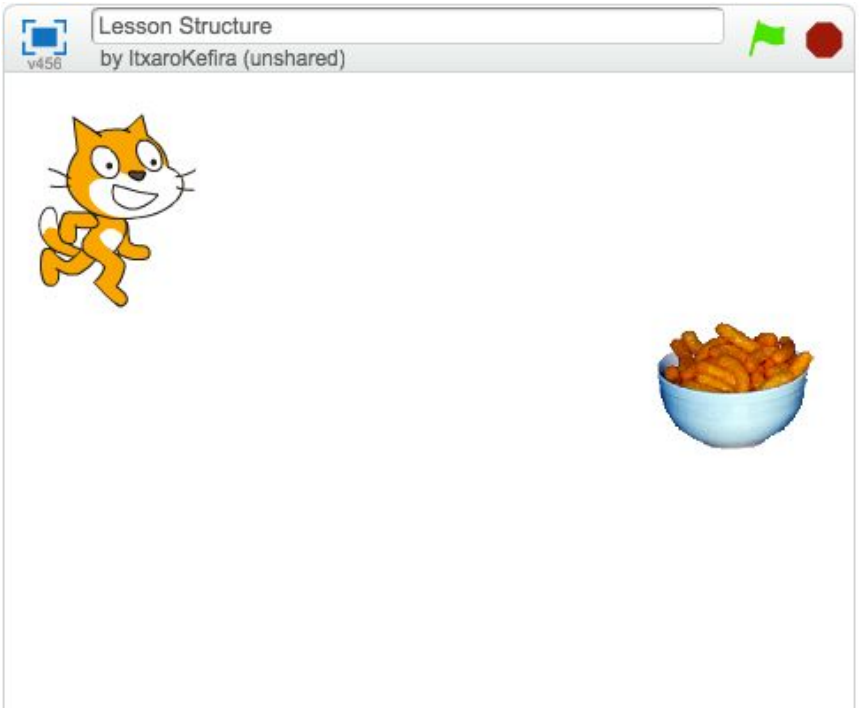
    - **Face:** Choose a sprite from the library that Scratch provides. This is nice because a few of the sprites already have costumes drawn
    - **Paintbrush:** Draw one using the built-in editor. This is handy for quick edits, but it's not as intuitive and easy to do as Photoshop so it's not recommended
    - **File:** Upload your own file with a custom made Sprite. This is the way to go if you want to create your own characters and animations.
    - **Camera:** Use the computer's camera to take a picture. We won't be using this function.

- We are going to use the pre-made sprites that Scratch provides for this assignment. Click the **Face icon** and select the Snack that the cat is going to pick up and press OK. Now this new sprite should appear in your sprite selection menu, so go ahead and select it. Notice that no code appears under the Snack, since all the code we've done so far was exclusively for the Cat. You can rearrange the snack, just like you did with the Cat, simply by clicking and dragging it around on the screen.
- What we want to happen here is "When the game starts, if touching Cat, hide". By hiding the sprite, it will look like the object has been **picked up**. The "touching Cat" function can be found under sensing.

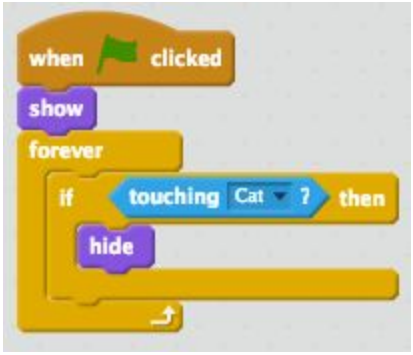


- Remember to put this all inside a forever loop so the Snack keeps checking if it's touching the Cat for the entire game, not just once.

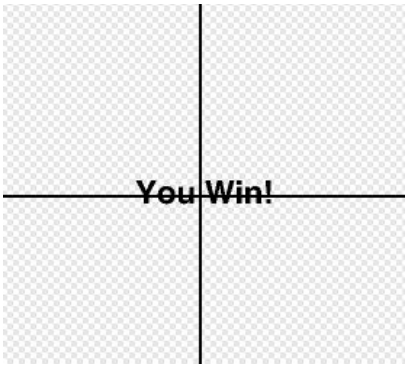
- If the Snack disappears right when the game starts, it may be because the Cat is already touching the Cat. Make sure your Snack is away from the Cat at the start of the game.



- Also, notice that the snack does not reappear when you restart the game. Just like the Cat's starting point, we told the Snack to hide, but we didn't tell it to show. Simply add a "show" function at the start of the game. Remember, we **only want it to show at the beginning of the game, not forever**, otherwise the snack won't hide when it touches the Cat.

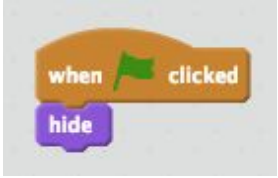


- Now that the Cat can grab his tasty snack, it's time to give him a place to lay down for a nap. Create another new sprite, using the same method as the Snack, and call it Carpet. Position it away from the Cat and the Snack.

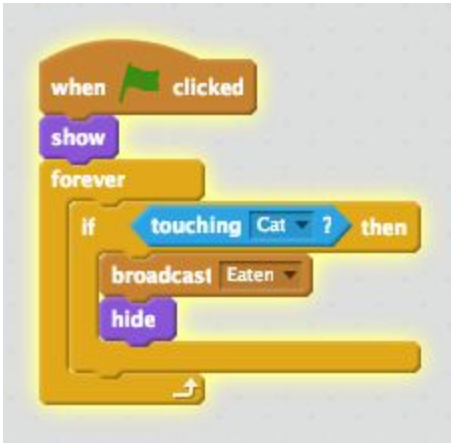
- We'll **create one more Sprite called You Win!** This time, use the Paintbrush option to create a custom sprite. Use the "T" text icon and write "You Win!". Then we want to get the image centered, so use the crosshairs tool on the top right of the editor, and click as close to the middle of your words as you can. This way, when you position this sprite in the game, it won't be awkwardly off to the side. Once that's all done, click back to the Scripts tab.
  - 
  - 

- Your screen should now look something like this:
  - 

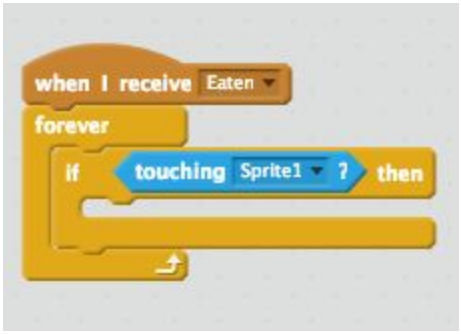
Part Five: Win Condition

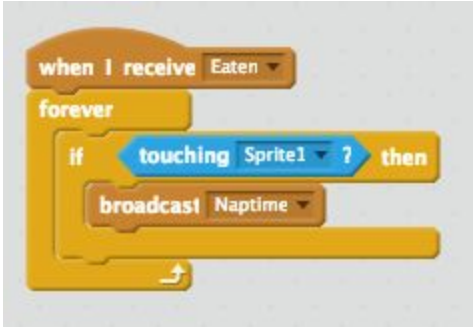
- There are two things that we want to happen with You Win!:
  - We don't want You Win! to show at the beginning of the game, only when the carpet is touched
  - We don't want You Win! to show unless the snacks have already been picked up
- First, let's tell You Win! to hide at the beginning of the game. Click on the You Win! sprite in the sprite window, and give it a simple string of code that says, "When game starts, hide". This will make it disappear each time the game starts until we want it to appear.
  - 

- Now we want to make the win condition appear when the carpet is touched. The tricky thing is, we don't want the You Win! sprite to show unless the snack is picked up **before** the carpet is touched. If you don't grab the snack, and just run to the carpet, you haven't truly won. To solve this, we are going to give the Snack some code that makes him "broadcast" that he's been touched.

- So basically the Cat will touch the snack, the snack will scream “Ahhh! I’m being eaten!” and the carpet will hear this and realize “Oh, the Snack has been eaten, it’s my turn!”. This is done using the **“broadcast” function** under the Events tab.
- Select the Snack and add drag a broadcast function into the “touching Cat” code. From the dropdown menu on the broadcast, **make a “new message” called “Eaten”.**
    - The snack code is now doing, “Am I touching cat? Yes? AH I’M BEING EATEN! Hide.”
    - 

- Now we want the Carpet to hear this message. Select the Carpet sprite, and give him an Event called “When I receive ___” and choose “Eaten” from the dropdown menu.
    - 

- Now we want to put in the **detection code** in the same way we did to check if the Cat was touching the Snack.
    - 

    - This means that instead of checking if the carpet is touching the cat when the game starts, it will only check once it hears that the Snack has been eaten.
- At this point, we have:
    - Snack checking if it’s been eaten
    - Snack yelling if it’s eaten
    - Carpet receiving the Eaten message
- Now we want to tell the **carpet to broadcast that it’s been touched**. Simply add another broadcast function into the code we were just working with, and call it **“Naptime”**.
    - 

    - This makes the Carpet yell “The cat is on me!”.

- The final thing we want to happen is for You Win! to appear. Select the You Win! sprite and give it a string of code that says **“When I hear that the Cat has touched the Carpet, appear”.**
    - 

- That’s all there is to it! Now your game should play as follows:
    - The cat can move with the arrow keys, and changes costumes to look like he’s walking
    - If the Cat eats the Snack, he can lay down on the Carpet for a nap.
    - You Win! will appear if the Cat touches the Carpet after eating the Snack.

Final Challenge:

- Make the Cat lay down on the rug at the end of the game as “You Win!” appears, but not before the snack is eaten. You’ve learned everything you need to do this in this lesson!
- Congratulations! You’ve made a basic game. Scratch can do a lot more, but these are the basics you need to code something. Feel free to look at other people’s projects on Scratch, and add things to your game to make it even more tricky!