Moving characters with keys

Move, walk, jump, double jump

In scratch you use coordinated to move your characters.

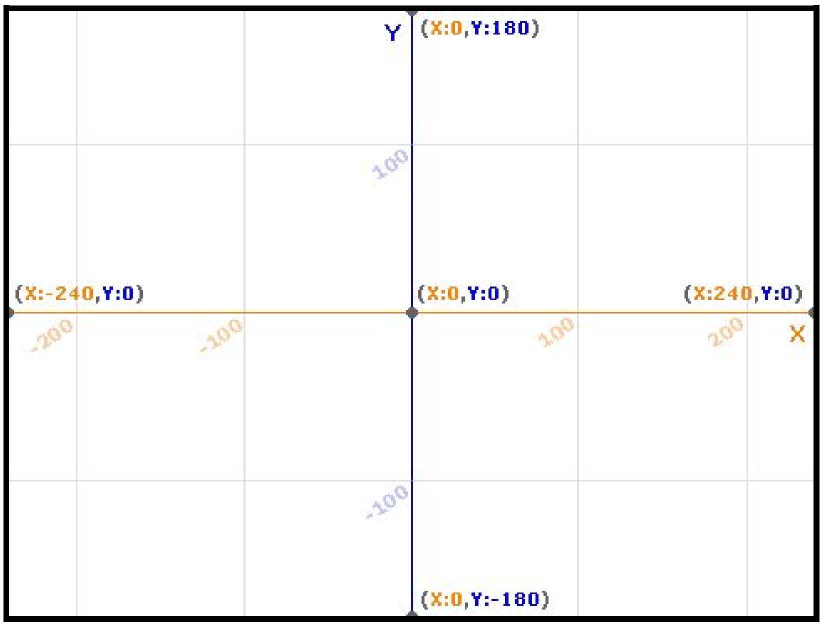
From side to side is called the **X Axis**, This is always the first number.

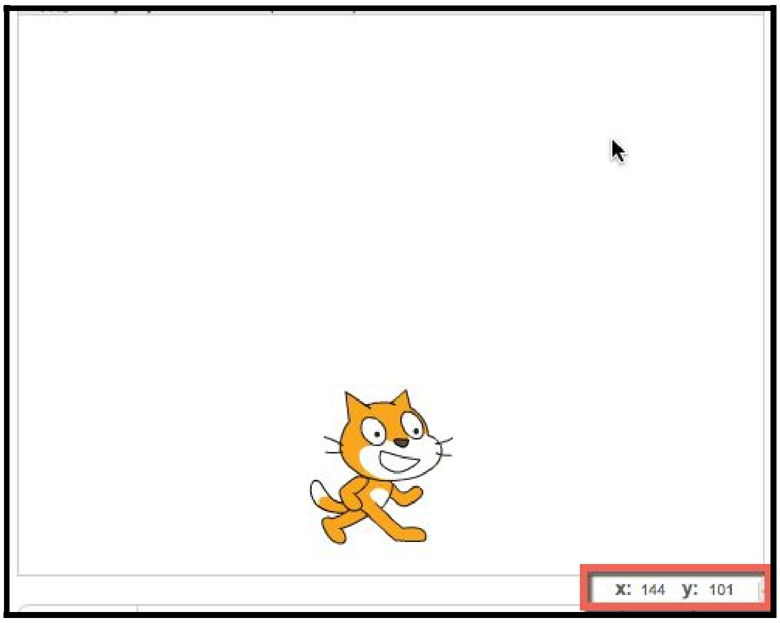
Up and down is called the **Y Axis**. This is the second number.

The Center of your page is called the **Origin**. It's X and Y number is **0**.

We write coordinates at two numbers, so the center is **0,0.**

In coordinates we also use negative numbers.
 Anything on either axis that is less than 0 becomes a negative number.





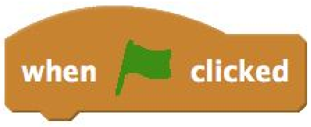
On the Y axis (up and down) *Up is positive, down is negative*.
On the X axis (side to side) *Right is positive, and left is negative*.

In Scratch we might need to find out the coordinates of a position on the stage.

Luckily we can find out my pointing the mouse at the spot, and looking at the numbers shown in the bottom right corner.

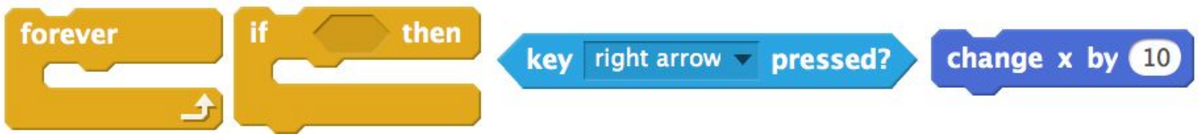
We're going to use these numbers to move our characters, and control all the elements in our game.

This code is going to be added to your game character, so make sure you have that sprite selected



Remember, we start all pieces of code with the **start block**.

Here are the blocks you will be using,



- The next thing we will need is a **forever loop**, you'll find this with the Control blocks. This makes sure the code is always running, and doesn't just run once. Put this straight after the start block.

- Inside this we will need two **if blocks,** If statements are also kept with the control blocks The **If blocks** will be **on top of each other**, NOT one inside the other.

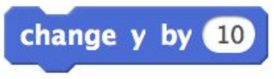
- Each If block will use a Sensing block called **Key pressed?** And you will choose the left and right arrow keys for each.

- And within each of these if blocks will be a Motion Block called **Change x by.** Remember right is positive and left is negative. So when the *right arrow* is pressed change x by **10**. And when the *Left Arrow* is pressed change x by **-10**.

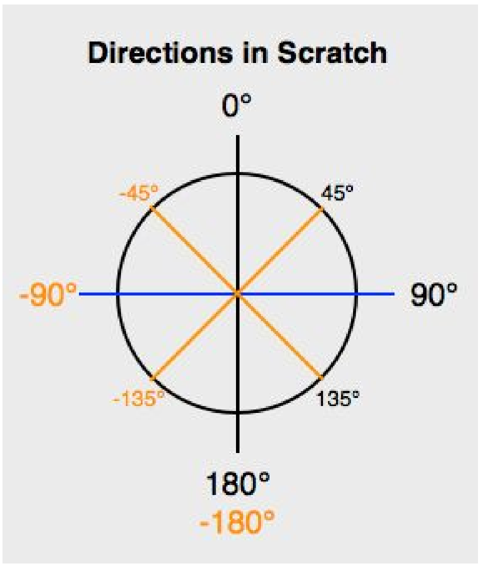
So now when you click the green flag on the stage, and press the left and right arrows, your sprite should move back and forth on the stage.

You have just mastered the basics of moving!

Depending on your style of game, you may also want to move up and down, to accomplish this take the same technique, and add blocks for *up* and *down* arrows, but this time changing the **Y axis** by 10 and -10.



the

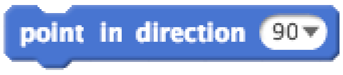


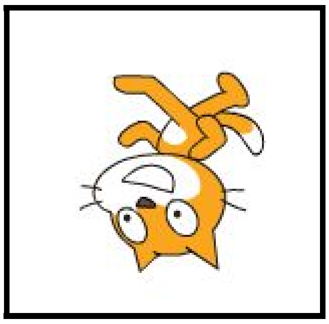
Changing Direction of the sprite

In scratch we use angles to control which way a sprite will move, or face.

There are 360 degrees in a circle.
By default, we are facing upwards.
90 degrees points right.
-90 degrees points left
180 degrees will point down.

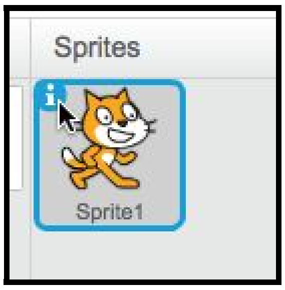
We want our sprite to change directions when we are moving left or right.
To do this we will use a motion block called **point in direction**

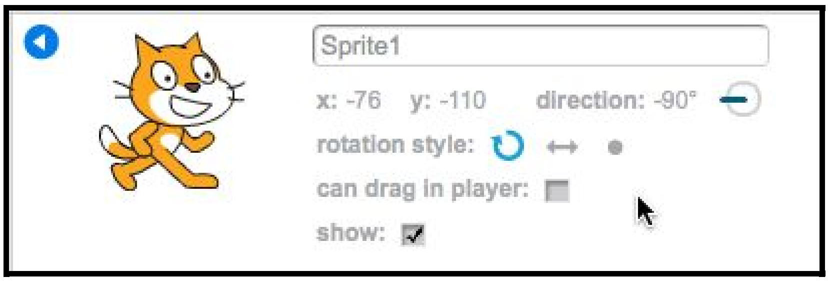




- Add this block to the *If loops*, before the ***change x by*** blocks.
- Set the value to 90 Degree when holding the right arrow, -90 when holding the left arrow.

But just adding the blocks will cause our sprite to flip upside down when they turn left. So we need to change a setting in the sprite info panel.





Sprite1
x: -76 y: -110 direction: -90°
rotation style:
can drag in player:
show:

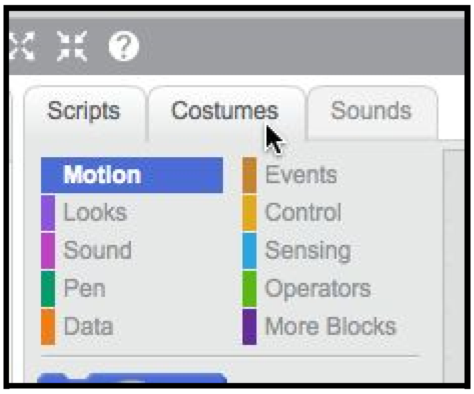
The Sprite info panel is opened from the blue i icon in the top left corner of your sprites.

It shows the X and Y coordinates of the sprite, and the angle direction it's currently pointing in.

- We need to switch our sprite **rotation style** so to the *left-right* style, By clicking the icon of the arrow with two ends.

Now when you move your cat left or right, it should face the correct direction.

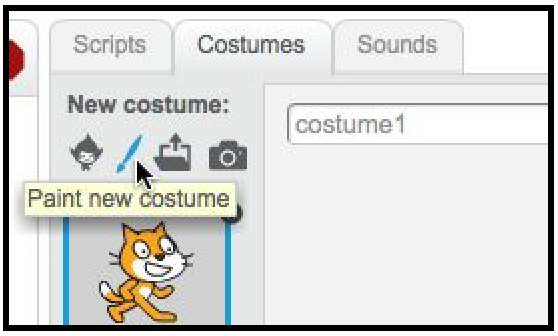
Creating a Walking Animation



In order to make our sprite look like it's walking we will use costumes. A costume is one of the many 'frames' or alternative appearances you might use for a sprite. One of the many ways you might use costumes is for **animation.**

If you are using the default scratch cat, he already comes with two costumes. If this is a character you created, you will need to create the other costumes.

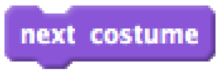
We can use the different frames to create what is known as a **walk cycle**.
This is the number of frames group of frames where a character takes a step, and then goes back to the first position, so that if its loops it looks like continuous walking.



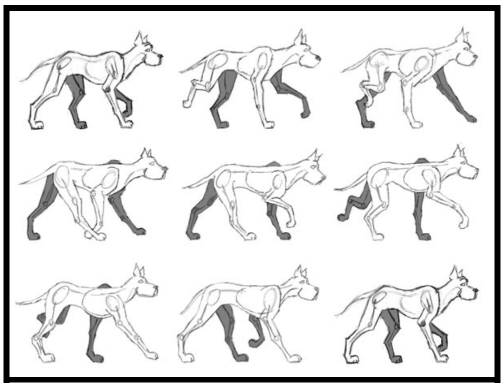
If you open the costumes tab in Scratch you can see the different costumes of the character. If this is your own drawing, you will need to make them.

You can choose a sprite, paint a sprite, upload a picture from somewhere else, or take a picture with the webcam. We can program the costumes using the purple Looks category of blocks.

- To use costumes for animation, find the **next costume** block and put it in the if loops for the left and right arrows. Along with the **change X by** and **point in direction** blocks.

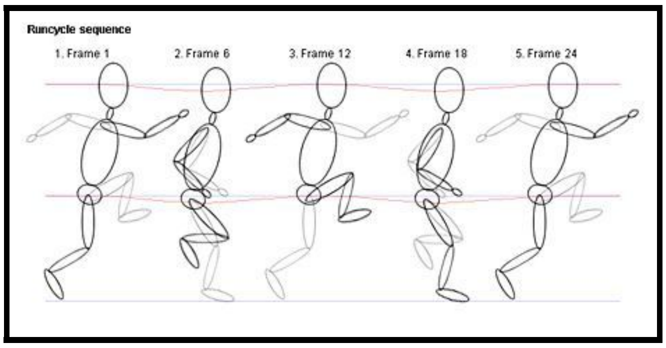


Now if we click on the green flag to start walking, and press our left and right arrow keys, it looks like the cat is running.



Tips
Walk Cycles take many different form, and can be as long and complex, or short and simple as you like. Different creatures will talk differently.

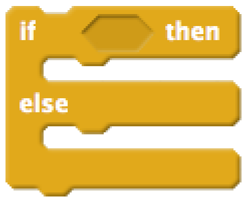
It's important to remember to end the cycle on the frame that would come before the first frame. So that it loops correctly.



Adding Gravity

We want our sprite to fall towards the bottom of the stage, and not go below it.
Remember that side to side is the X axis. And up and down is the Y axis.
So we want to make sure that the Sprite does not go below a certain point on the Y Axis.

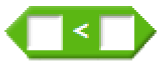
- Put your sprite on the bottom of the page. You can use the *sprite info panel* to find the Y value of this position.



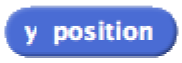
We will use Code to make sure our sprite can't dip below the floor.

- After your *if statements* for left and right arrows, add an **if else** block.

We will then use an Operator Block called **less than**.



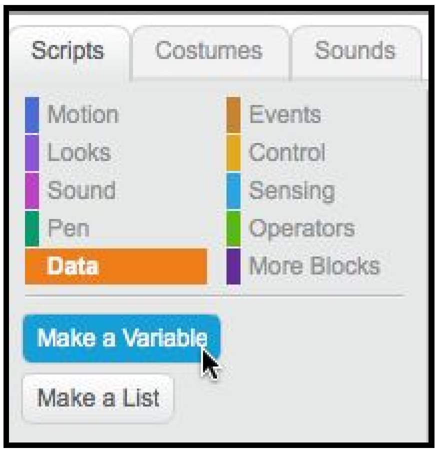
In Scratch, Hexagonal blocks (6 sides) are Boolean Blocks.
A Boolean checks if something is true or false.



We will be checking if the Y position of the sprite, is less than the number it will be when the character's feet are on the floor. Remember to find this Y value in your sprite information panel

- Take your **less than** block, and put it in the **if else statement**.

- Put the Y position in the first slot of the **less than** block, and the **Y position** of your sprite in the second.

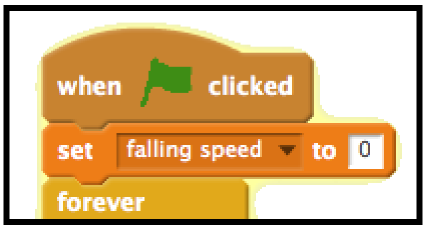


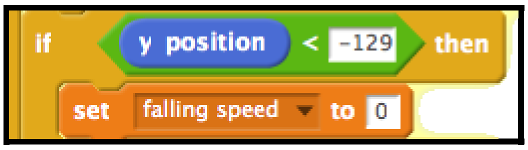
Now we are going to create a **variable** is a piece of information that Scratch holds on to, that can be changed with the code. An example would be the score in a game. This would be a number, that goes up as you hit your target.

You find Variables in the Data section of your blocks.

- Click the **Make a Variable** button, and name it '**falling speed**'. Leave 'for all sprites' clicked. And click OK.

- The first thing we will want to do is set our falling speed to 0. So take this clock and add it to the very top of your code, under the **when clicked** block. So that this will be set before any other code runs.
- We also want the falling speed to be set to 0 whenever the sprite hits the bottom of the page, so drag another '**set falling speed**' to the first part of your *if else*.
- Take a **set Y to** block, and put it under this falling speed block. Set the Y value to one pixel less than Y value you used before. In my example I used -129. So I will set my Y value to -130.

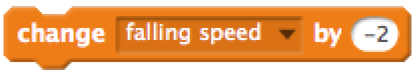




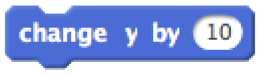
We are going to create code to make our character fall back down if they are above the ground, like gravity. Our code currently says if the Y position is less than -129 (or any value that has the character's feet on the bottom of the stage) then turn off the gravity, and stay at -130 - and so it will stop falling.

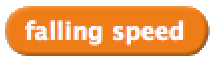
- Take a change falling speed by block, and add it to the *else* part of the **if else** statement. Change its value to **-2**.

Remember that **set to -** will change the falling speed to that number. But **Change by** will increase or decrease the speed by the number you state.



- Next, from the motion category, get a **change Y by** block, and put it inside the *else* block.
- We want to change Y by whatever the falling speed is. So, from the Data category, choose a '**falling speed**' block, and slot this into the **change by Y**





Now if we run our code, and move the sprite upwards, they should drop to the bottom of the stage, and stop. Looks like we have working gravity!

Creating a Jump

Now it's time to help this character to jump!

- Go to the brown Events category and select the **when space key pressed** block.

when space key pressed

The code that goes under this block will activate when we hit the spacebar, and allow our game character to jump .

set falling speed to 20

- From the Data section, get a **set falling speed** block. Add it to the **When Space Key** block.

Our falling speed is a negative number, that is linked to the Y axis. This means the Y value of the game character is above the floor, It will start getting lower until it reaches the floor.

change y by 10

- Take a blue motion block called **change y by** and attach it *under* the set falling speed block we had just added.

falling speed

- Then from the Data Section, get the **falling speed variable**, and put it inside the **change y by**.

Now the character will jump, but if we keep tapping the spacebar , it will just fly away, we need to place a limit on this jumping, or the game will be too easy.

- Go to your block of code for the jump, and break it off after the **when space key** block and slot in an **If condition.**

- We will use this to check that the character is on the ground in order to jump. Take an **equals block** from the Operators and slot it into the **if statement**, with the rest of your jumping code inside it.

- From the motion category, grab a **y position** bubble. And slot it into one end of the equals block. And in the other end, add the Y position your character is at, when it's on the ground.

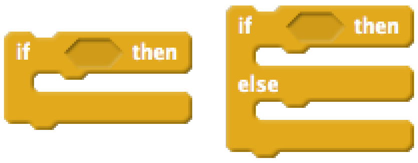
Now my character can only jump off the ground, instead of flying away.

The Double Jump

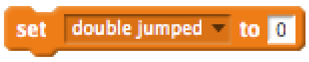
But in our game, we wanted our character to be able to do a double jump.
How can we find a middle ground between one jump, and floating away into the sky?

- First, we will create a new **variable** from the data section called '*double jumped*'.

- We will take the code we created before to make a jump happen, and change the **If block**, for an **if else block**, and slot the code we created before into the new if block.



- We will need to set our new doubled jumped value at the start of the program. So take a **set double jumped** to block, and put it *at the very top* of the code block, **under** the '**When clicked**' block.

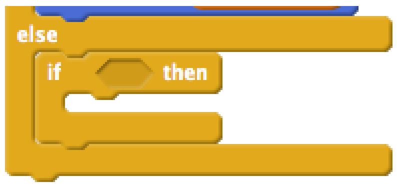


```
set double jumped to 0
```

We know that variables can contain numbers, but they can also contain text. When we use a boolean we set something to either true or false.

- Change the value of **set double jump** to **false,** by typing 'false' into the box'

- Take another If statement, and slot it into the else part, of the else if you have running when the spacebar is pressed.



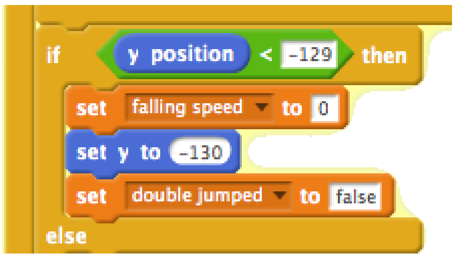
We are going to set our double jumped variable to be set to false if the character hasn't, and true, if they have.

- Grab an **equals block**, from the operators section, and slot it into this **if statement**. The take the variable for '**double jumped**' and slot it into one side of the equals.
- In the other side write the word 'false'.

If the character hasn't double jumped, we want them to get one more jump. We can copy code from before to do this.

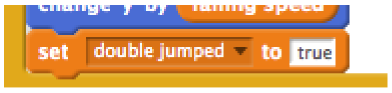
- We can right-click on code to make a duplicate. Do this by clicking on the first '**set falling speed**' in the **if else** block, and place the duplicate in the else part.

We want to make sure that the cat has to come back to the ground before it can do another double jump.



```
if y position < -129 then
    set falling speed to 0
    set y to -130
    set double jumped to false
else
```

- So in the main part of the code we will take another **set double jump to** block, and add it to the first part of the if statement at the bottom of the main code. The part where you had set the falling speed to 0 when the character is on the ground. After the character has done a double jump, we want to set **double jump variable** to **true**.



```
set double jumped to true
```

- So we take another **set double jump to** block, and slot it into the last if statement, of the jumping block and in the text box we type in ‘**true**’.

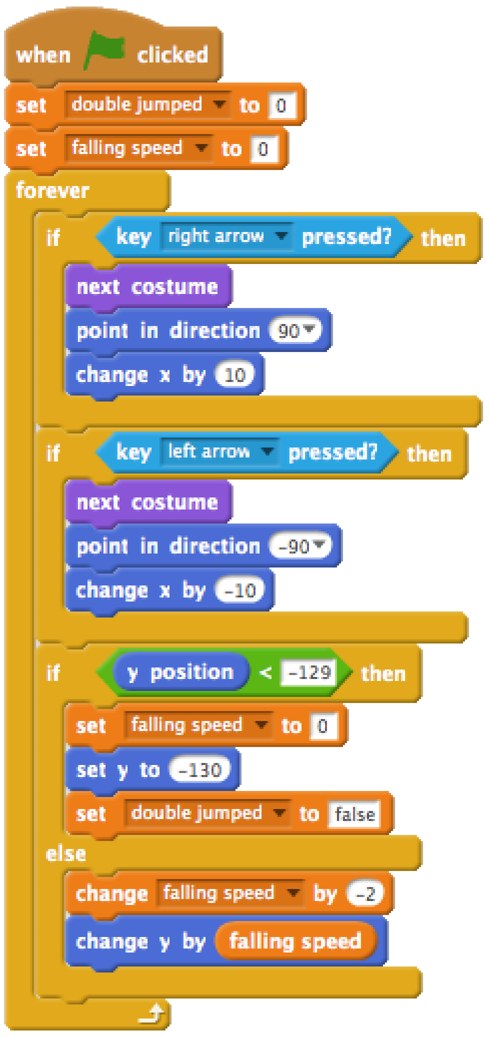
Now run your program, notice that the double jump variable will be ‘false’ until you press space bar twice, and you can’t jump any higher than this.

What kind of games will you use this technique in?

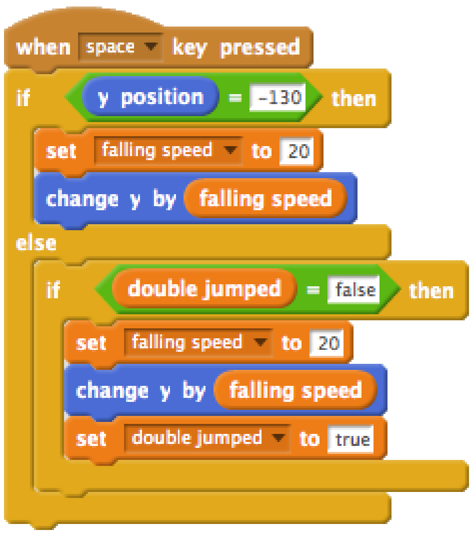
Find this Demo in video form on **InventwithScratch.com**
https://inventwithscratch.com/double-jump-demo/

The Code

No peeking!



```
when [flag] clicked
set [double jumped ▾] to [0]
set [falling speed ▾] to [0]
forever
    if <key [right arrow ▾] pressed?> then
        next costume
        point in direction (90 ▾)
        change x by (10)
    if <key [left arrow ▾] pressed?> then
        next costume
        point in direction (-90 ▾)
        change x by (-10)
    if <(y position) < [-129]> then
        set [falling speed ▾] to [0]
        set y to (-130)
        set [double jumped ▾] to [false]
    else
        change [falling speed ▾] by (-2)
        change y by (falling speed)
```



```
when [space ▾] key pressed
if <(y position) = [-130]> then
    set [falling speed ▾] to [20]
    change y by (falling speed)
else
    if <(double jumped) = [false]> then
        set [falling speed ▾] to [20]
        change y by (falling speed)
        set [double jumped ▾] to [true]
```