Tutorial

Cyclic Bouncing Animations in Scratch

Level: Grade 7–12 • **Topic:** App Development, Art & Design, Digital Storytelling, Game Design • **Subjects:** Computer Studies

*If you are looking for an accessible online version of this content, please visit the **Pinnguaq website*** (link: https://pinnguaq.com/learn/cyclic-bouncing-animations-in-scratch).

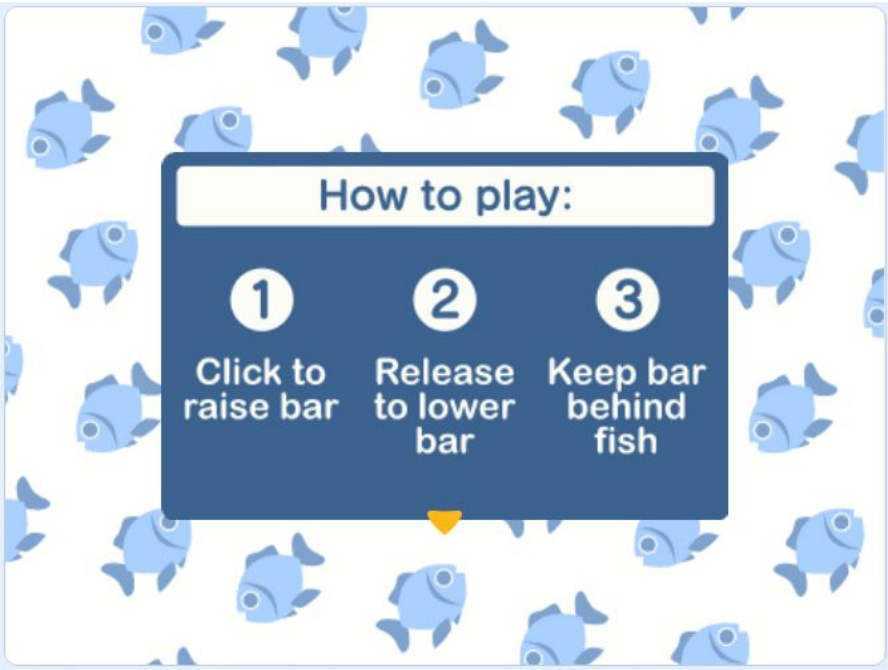
About the Author

Jonathan Weber

Jonathan is a digital steward on Pinnguaq's delivery team where he travels to communities to provide hands-on opportunities to develop digital skills, such as coding and digital art, for both students and adults. He has bachelor degrees in chemical engineering and computing technology, as well as a master's degree in education, and has previously worked as a software developer and analyst in government and higher education.

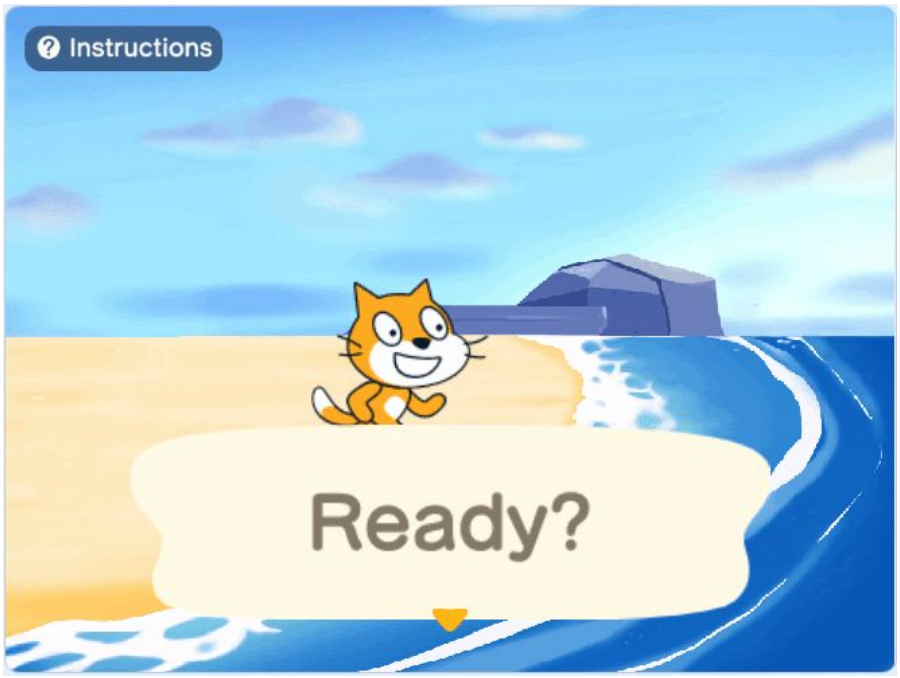
Introduction

In this tutorial, you will learn about how conditional statements and a counter variable can be used to create a bouncing arrow animation in Scratch, pictured below from my *Animal Valley: Extreme Bass Fishing* (link: https://scratch.mit.edu/projects/401411010/) Scratch game:



Small, cyclic animations—defined as animations that repeat over and over—are an easy way to draw the attention of a player or user to a certain place on the screen while also providing feedback to them that a program is waiting for their input. Feedback is an important aspect of a user's experience that helps them to understand what is going on in the program and what might be expected of them (*Note: if you're interested in user experience, check out our lesson Creating Positive User Experiences when Introducing Players to Your Game* (link: https://pinnguaq.com/learn/creating-positive-user-experiences-when-introducing-players-to-your-game)).

I used a small, bouncing arrow on the instructions page to indicate that a click is required for the game to continue to the next screen or state. That same arrow animation is used with the text bubbles to prompt the user to advance the text, as below:



Using the same animation in multiple places helps to orient a user to how they're expected to interact with the program that they're using. If different animations or icons were used, the user might expect different inputs to be required or a different effect to occur.

Using the same animation throughout a project also has the added benefit of us needing to develop a solution only once: after the problem is solved the first time, we can reuse that solution and code as needed and save ourselves the time and resources of developing a new solution, both in this project and in future projects (*Note: to learn how to use code between projects in Scratch, see my tutorial on Using the Backpack in Scratch* (link: https://pinnguaq.com/learn/using-the-backpack-in-scratch)).

Vocabulary

- **Cyclic animation** - An animation that follows a repeating cycle which allows it to continue indefinitely or as required. A "walk cycle" is a common example of a cyclic animation that is used to animate a character's walking motions.
- **Modulo (mathematics)** - Commonly denoted by "%" or "mod", modulo is an operation that gives the remainder of the division of one number by another number. For example, *6 mod 3 = 0*, since *6 ÷ 3 = 2* with no remainder; *13 mod 4 = 1*, since *13 ÷ 4 = 3* with a remainder of 1.

Computer Activity

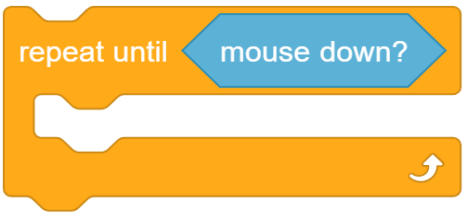
There were two main considerations for creating the cyclic animation used in *Animal Valley: Extreme Bass Fishing*:

1. Determining how the looping/repeating would occur
2. Figuring out how to create the up-and-down bouncing motion

Looping/repeating in Scratch

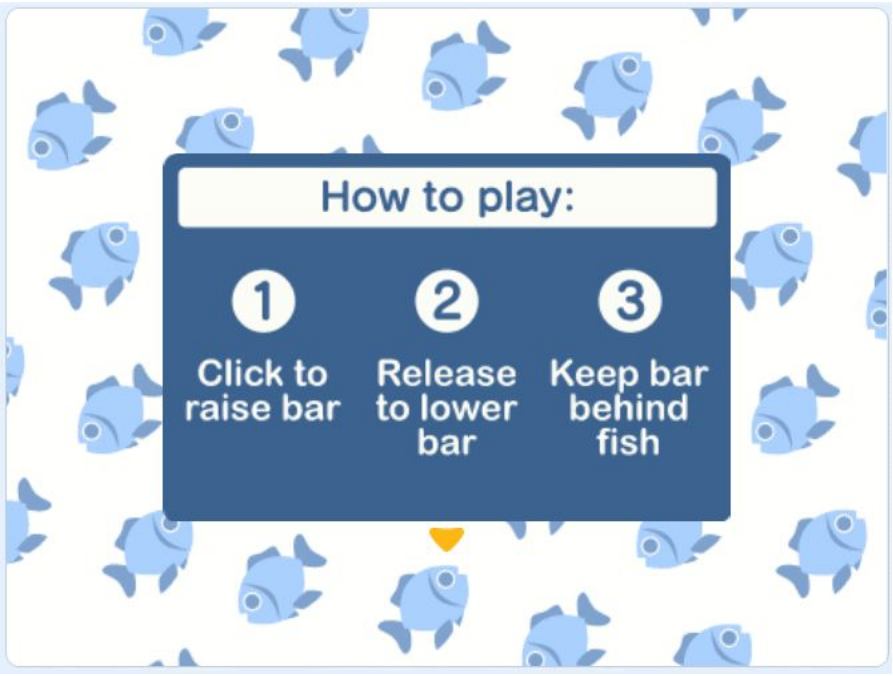
There are a number of built-in ways to repeat code in the "Control" category of blocks, such as the "**Repeat**" block, the "**Forever**" block, and the "**Repeat until**" block:

For *Animal Valley: Extreme Bass Fishing*, I had the extra requirement of using this particular animation to indicate to the user that their input was required so I needed the animation to repeat until the user clicked their mouse. This meant using the "Sensing" category block "**Mouse down?**" with the "**Repeat until**" block to create code that will run until the mouse is clicked:



Up-and-down bouncing

The bouncing animation is driven by a repeating up-and-down motion cycle. What that means is that the arrow first moves down for an amount of time and then changes direction to move up for the same amount of time that it moved down. The times must be equal so that the arrow doesn't end up drifting up or down, like so:



This means that we would need some sort of counter variable to keep track of how many or how far the arrow has moved down and then switch the direction of movement for the arrow so that it cycles in the same place. As an exercise, try and build the algorithm that I will describe below (hint: you will need to create a "counter" variable in the "Variables" category). In this example, I want the arrow to move down for 5 steps and then move up for 5 steps for a total animation length of 10 steps.

1. Set counter to 0
2. Repeat until the mouse is clicked:
    a. Check if the counter is less than 5 or exactly 5:
        i. If so, move the arrow down by 1
    b. Check if the counter is more than 5:
        i. If so, move the arrow up by 1
    c. Increase the counter by 1
    d. Check if the counter is greater than 10 or exactly 10:
        i. If so, reset the counter back to 0

I wrote out steps 2a and 2b separately for clarity, but in reality, they don't need to be separated into two conditional statements and instead can be done using an "**If … then … else**" block from the Control category.:

- Check if the counter is less than 5 or exactly 5:
    - If so (then), move the arrow down by 1

- If not (else), move the arrow up by 1

Keep in mind that Scratch doesn't have "Less than or equal to" or "Greater than or equal to" comparison operators, so when we're setting up our conditions, we can only use "Less than" or "Greater than". Luckily, though it's a bit less clear to read, it's a simple enough fix since:

Check if counter is less than 5 or exactly 5 (less than or equal to)

is equivalent to

Check if counter is less than 6

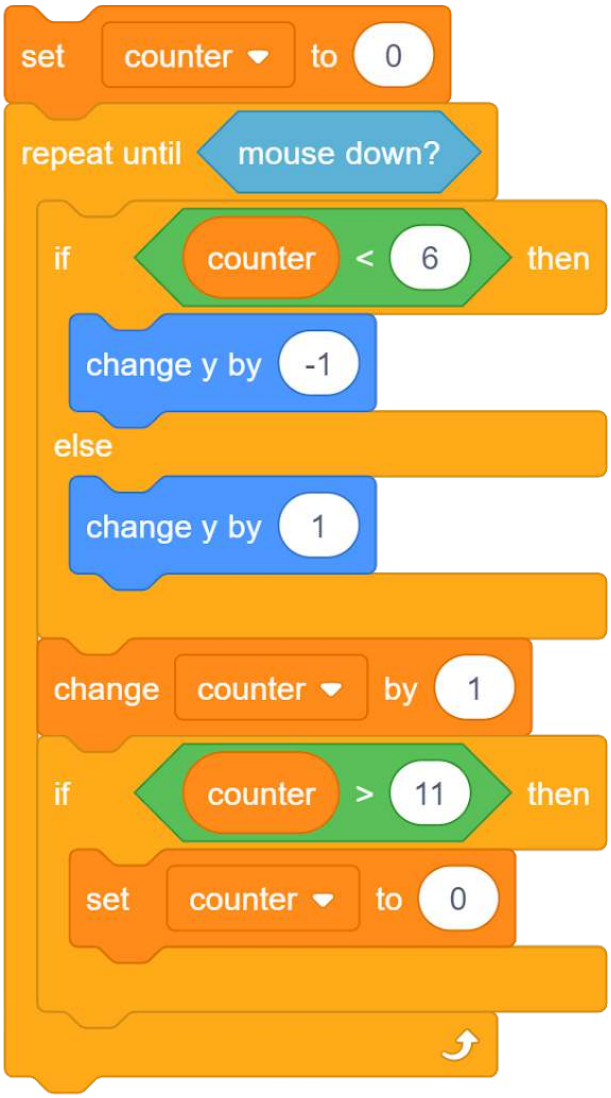
Likewise, for checking when to reset the counter:

Check if counter is greater than 10 or exactly 10 (greater than or equal to)

is equivalent to

Check if counter is greater than 11

This is the actual code that is used to control the bouncing arrow in *Animal Valley: Extreme Bass Fishing*:

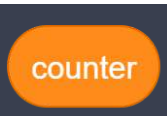
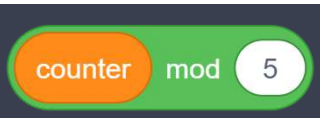


Using modulo instead of greater than and less than (optional alternative)

The technique I showed above isn't the only way of accomplishing the goal of a cycling, bouncing arrow. While using "greater than" and "less than" comparisons work perfectly fine, you can also use the modulo operator to accomplish the same effect. The modulo operator, denoted by "%" or "mod" where a "+" or "÷" would normally be, gives the remainder of the division of one number by another: *6 mod 3 = 0*, for example, since *6 ÷ 3 = 2* with no remainder; and *13 mod 4 = 1*, since *13 ÷ 4 = 3* with a remainder of 1.

Modulo is a useful operator because it will be 0 on every multiple of the divisor. In the example above, *6 mod 3 = 0*, but any multiple of 3 will also be 0. We can use this property to set up a condition where the direction of movement switches every time "*counter mod n = 0*" is true,

where *n* is half the length of the complete cycle (e.g. 5, in the code of the game). This can be better understood with the use of a table:

 counter	 counter mod 5
0	0
1	1
2	2
3	3
4	4
5	0
6	1
7	2
8	3
9	4
10	0
11	1
12	2

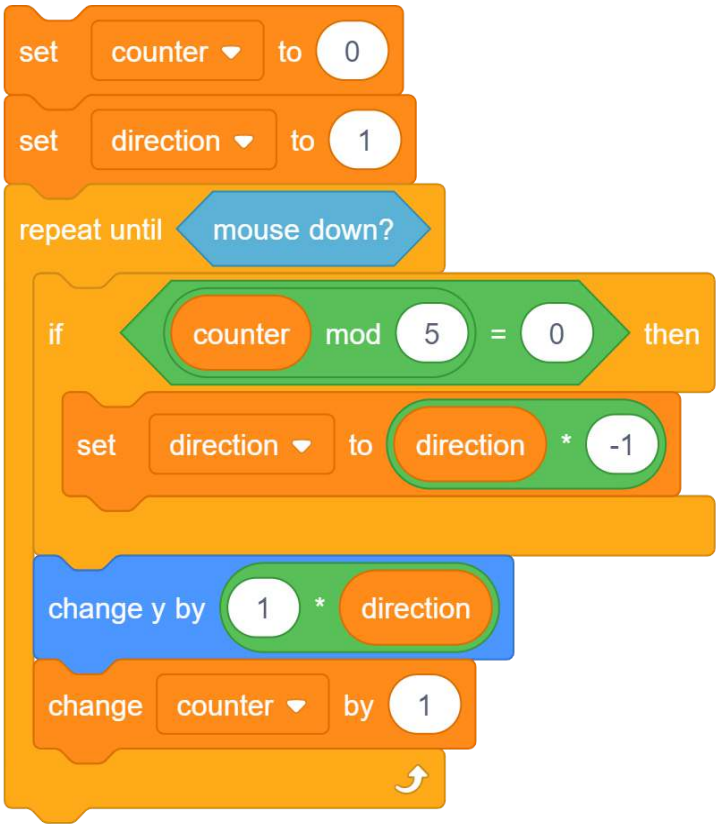
You can find the modulo block in the “Operators” category:

The algorithm remains mostly the same, but there are a few key differences:

1. Set counter to 0
2. Set direction to up
3. Repeat until the mouse is clicked:
    a. Check if the counter mod 5 is 0:
        i. If so, change the direction

b. Move the arrow by 1 in the current direction
c. Increase the counter by 1

And the code looks like:



There are a few notes about why it works the way that it is:

1. I am tracking direction using a new variable called “direction” which should be either 1 or -1.
2. The condition for the **“if”** block has been replaced by “**counter mod 5 = 0**”. When this is true, the new direction variable is multiplied by -1 which will make it positive if it is negative or negative if it is positive. This happens every time the counter is a multiple of 5.
3. In Scratch’s convention, since up is positive on the y-axis and down is negative on the y-axis, the block “**Change y by 1 * direction**” will move up if the direction variable is positive and down if the direction variable is negative.
4. There is no longer a need to reset the counter since it doesn’t matter how big the number is, only if it is a multiple of whatever we’re using in our **“if”** block.

Conclusion

This quick tutorial introduced the notion of cyclic animations as ways of providing feedback to users while also demonstrating two techniques that can be used to include cyclic animations in your Scratch projects. It also introduced the "modulo" operator which can be used in conjunction with a counter variable to support cyclic behaviours. These techniques will come in handy as your projects become larger and more complex because it is likely that users will require guidance, at least initially. While demonstrated in the context of a Scratch game the techniques covered in this tutorial can be recycled and adapted for use in other projects, including outside of games.

We want to see the awesome things you're creating! Take a photo or video and share your work with us by emailing media@pinnguaq.com or tagging **@pinnguaq** on Facebook, Twitter, or Instagram. Don't forget to include the hashtag *#LearnWithPinnguaq!* You can also upload your project to the **Pinnguaq Studio** (https://scratch.mit.edu/studios/26567463/).

Resources

- Scratch Wiki (https://en.scratch-wiki.info/)
- Scratch Wiki - () mod () block (https://en.scratch-wiki.info/wiki/()_Mod_()_(block))
- Animal Valley: Extreme Bass Fishing • Scratch game (https://scratch.mit.edu/projects/401411010/)
- Using the Backpack in Scratch • Tutorial (https://pinnguaq.com/learn/using-the-backpack-in-scratch)
- Creating Positive User Experiences when Introducing Players to Your Game • Lesson (https://pinnguaq.com/learn/creating-positive-user-experiences-when-introducing-players-to-your-game)