Flipping Out With Scratch

Tossing Coins and Playing with Probability in Scratch*

Things You Will Need

- Coins for students to toss
- Something for groups of students to write on
- Something for groups of students to write with
- Devices to access Scratch (scratch.mit.edu)

Unplugged Coding

Coding without a computer is called unplugged coding. It is a really good way to begin a coding challenge to orient students to the problem physically or representationally before moving to coding the computer.

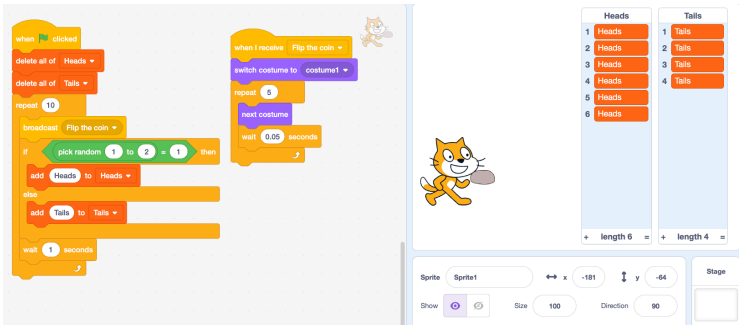
- Ask students to all toss a coin together
- Hands up if you tossed a head (or tails, you choose, it doesn't matter)
- Look around, tell them you are looking to see if half of them had tossed a head
- Why are you looking to see if half tossed heads?
- Get to the idea of tossing a head as having a probability of one half
- Keep going and toss coins many times waiting for the half to happen
- How long are we going to have to do this? It's even more fun if you have an odd number of students tossing and they finally realize it will never happen - join them at this point making an even number of coins being tossed
- If only there were a way to have a computer do the boring, repetitive work for us …

Plugging in and Coding

Students may be given the program *Coin Flipper 101* to start them off.

https://scratch.mit.edu/projects/1074477276/editor/

Starter code like this is valuable to get students started in on a problem without having to figure out how to code a program from scratch.



Have them explore and see how many times they need to run the program until they get a result of five heads and five tails. Not everyone will have this happen at the same time. Build the need to

modify the program to do the things we need it to so we can find out what we want to know about coin flipping probabilities and ***why it seems that we don't seem to get half heads half tails each time. Like what the heck is going on? We've been flipping coins since grade one and they always told us that flipping a head has a probability of one half.***

Starting to Stretch

Once you start getting student questions about how they can investigate why half and half doesn't seem to happen even half the time, encourage students to modify their code (or even start fresh) to answer their questions.

Encourage them to share their programs with each other and for others to use their programs to continue their investigations.

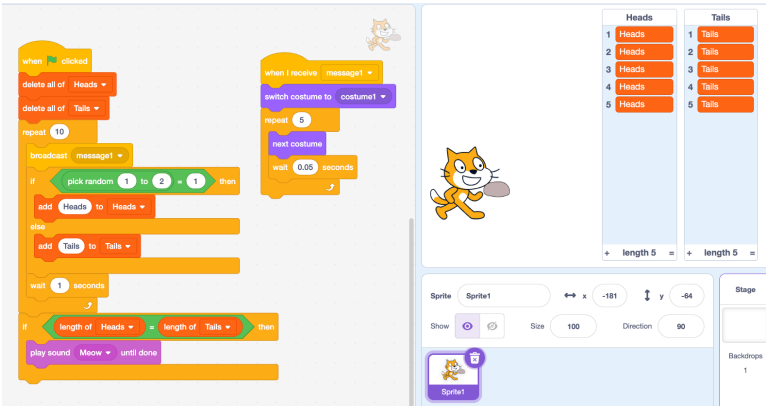
Below are some questions students asked and the solutions they coded to help them answer them.

Example Student Question 1

It's getting boring waiting for half and half to happen. Can we make it alert us when that happens?

Here is one solution we came up with to alert us to a successful result of half heads and half tails. How did we do it?

https://scratch.mit.edu/projects/1074480504/editor/

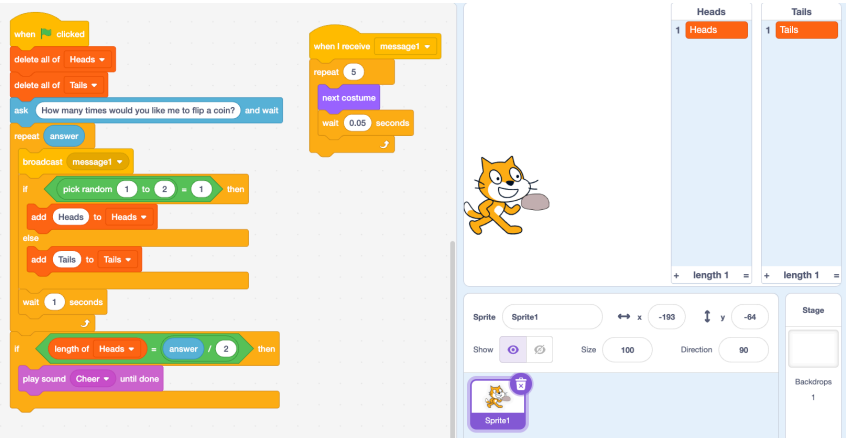


Example Student Question 2

Do we always have to flip a coin ten times? What about more times? Less times? A LOT of times.

We added a simple block to allow for user input. Now we can try different numbers quickly.

https://scratch.mit.edu/projects/441859232/editor/



Other Questions That Might Come Up

There are going to be a lot of different questions that can come up that can be coded for investigation. As the teacher, you can go around and prompt and poke your students into areas of inquiry that they might not have thought of.

It is also completely okay for students to spend time modifying and coding a program to investigate things that we as teachers know not to be a thing. Let them find out by themselves. That time is not wasted it is a valuable real life lesson that not everything works every time.

If we take ten tosses as one trial, how could we make the program run multiple trials for us? And keep track of them?

Can we have the program keep track of the experimental probabilities as they come about. How about getting it to stop once it matches our hypothetical calculated probability?

*Low Floor High Ceiling

The idea of low floor high ceiling mathematics activities has become much more common as our teachers learn how to incorporate the teaching of rich mathematics with student learning. The idea is that you start off with a nice, accessible bit of low floor mathematics that leads to ever more complex and wonderful ideas and high ceiling concepts. Coding can put low floor high ceiling activities into overdrive because we can program a computer to do all the heavy mathematical lifting for us.

Flipping a coin is a very nice low floor, but the interesting mathematics involved with flipping a coin many times gets tied down with the burdensome chore of keeping track of results, and honestly, it is quite boring. Enter the computer and our ability to code a program to do that burdensome

boring part for us. Once we are freed from these mundane tasks, we can investigate probability situations that we couldn't if we were flipping our digital binary decision makers ourselves. (Digital - finger. Binary decision - heads or tails. Heads we go, tails we stay. Flip a coin.)

A Higher Floor

You may have heard the phrase, "It'll average out," refering to the law of large numbers where if you do something probabilistic enough times, the theoretical and experimental probabilities will coincide. Well … theoretically that might be the case, but in practice?

A game to play with your students:

Everyone stands up and tosses a coin.

Heads keep standing, tails sit down.

Everyone still standing had a probabiity of being still standing of one half.

Those still standing tosses their coins.

Heads keep standing, tails sit down.

Everyone still standing had a probabiity of being still standing after two tosses of one quarter or one half times one half.

Keep going until there are but a few left. Keep calculating the probability of them still standing after n-many rounds. 1/2^n.

Then ask the class what the probability of these particular students to be still standing after the next coin flip. It's only 1/2.

The probability is only ever 1/2 for any one flip. It is 1/2^n for n flips. Fun, eh? What do large numbers have to do with this?

Probability is Just Organized Counting

You can make a nice chart of the number of ways to get certain results as you continue to flip a coin.

One toss of the coin.

One way to a head or one way to a tail. Two possibilities, each outcome is one of them. 1/2 H and 1/2 T.

A second toss of the coin.

One way to a HH, one way to a TT, two ways to a toss and get one H and one T (HT or TH) Four possibilities total. One way for 2H, 1/4. One way for 2T, 1/4. Two ways to get one T and one head, HT or TH, 2/4.

Continuing on we get this really nice triangle that we in the West call Pascal's Triangle. In the East it is known as Yang Hui's Triangle. Yang Hui

```
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5   10  10  5   1
    1   6   15  20  15  6   1
  1   7   21  35  35  21  7   1
```

has been dead a considerably longer time than Pascal, so it's probably not Pascal's. Honestly it might not be Yang Hui's either, but he did popularize it way before Pascal did.

Using this triangle, we can calculate the probabilities of getting an even number of heads and tails as an outcome. Each row is another coin toss.

Watch out, it becomes less likely to toss exactly half heads and half tails the more times in a row you flip your coin.

Is it breaking that law of large numbers?

Is that law really a law like the law of gravity?

There is so much more to get curious about and play with even if we do not have the mathematical understanding of it yet. That is the beauty of low floor high ceiling mathematics done with coding.

This is just the beginning of what you can do with the low floor of flipping a coin.

More Mathetical Links to Coin Flipping

- Colouring in Yang Hui's Triangle leads to a type of fractal called a Sierpinski gasket or triangle. You colour in all the even numbers on the triangle and those multiples of two make triangles within the big triangle. Guess what happens when you colour in all the multiples of three, or four, or more? There are programs out there to do this for you, but this is definitely an activity best done unplugged - it makes times tables interesting.
    - https://fractalfoundation.org/resources/fractivities/pascals-sierpinski-triangle/
- Here is just a beginning of an exploration of the patterns in Yang Hui's triangle for littler mathematicians.
    - https://fractalfoundation.org/resources/fractivities/pascals-sierpinski-triangle/
- More number patterns in the triangle like where the Fibonacci numbers are hidden.
    - https://byjus.com/maths/pascals-triangle/

While we are at it, the Fibonacci numbers are not really Fibonacci's any more than the triangle is Pascal's. Pingala, an Indian mathematician laid out this sequence of numbers in about 200 BCE. Similar to Pascal and Yang Hui, Pingala is way deader than Fibonacci so Pingala has a better claim, but again, we don't know if Pingal discovered the sequence or just popularized it.

- How to draw a Sierpinski triangle. Very satisfying to do. Amazing way to practice measuring and dividing by two.
    - https://youtube.com/watch?v=j6WYSdHuWgY
- How to generate the Sierpinski triangle out of seemingly random chaos.
    - https://youtube.com/watch?v=kbKtFN71Lfs
    - My Scratch code that simulates the chaos game.

    - https://scratch.mit.edu/projects/158116518/editor/
- My student's code that simulates the chaos game but with more math.
    - https://scratch.mit.edu/projects/923957908/editor/