2
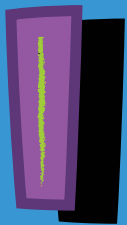# RAINBOW LINES IN SPACE!

n this chapter, you'll create a cool-looking animation: a rainbow *V* that flies through space and leaves colorful trails behind. This program was inspired by the *demoscene*, a subculture of elite programmers who made amazing graphical programs starting in the 1980s.

Demosceners made beautiful, elaborate programs called *demos* that showed off their artistic, musical, and programming skills. But most amazing of all, these programs were *tiny*—just a few kilobytes! The program we'll write isn't quite as small, but it's dramatic and colorful, and it uses only a few lines of Scratch code.

Before you start coding, look at the following figure to see what the final program will look like. Then go to *https://www .nostarch.com/scratchplayground/* to play the animation.
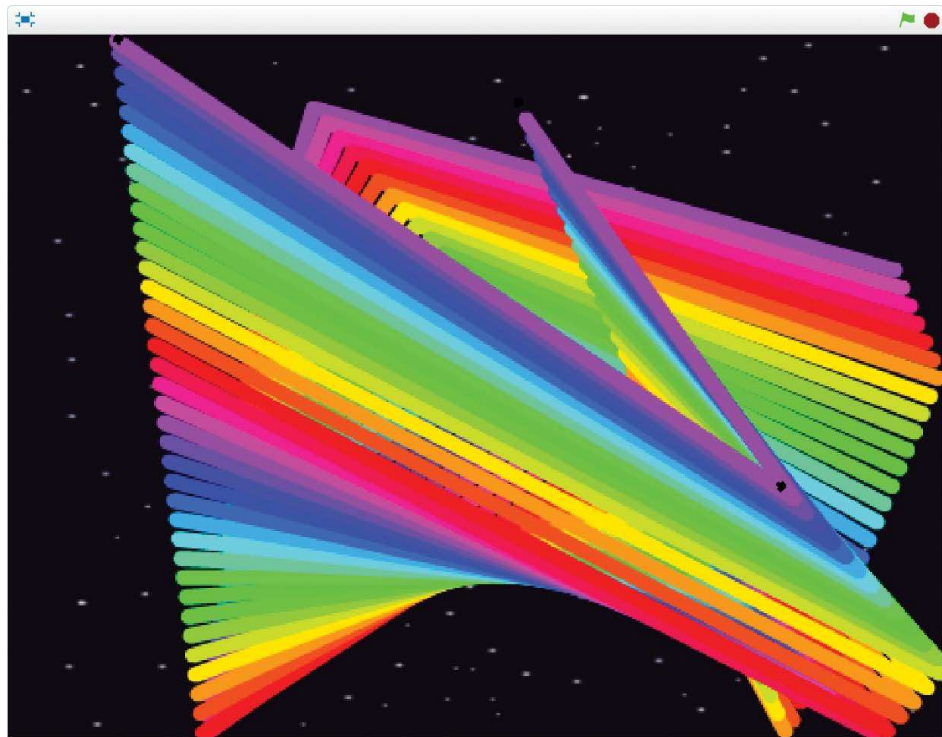
Just like demosceners, you can make beautiful programs. Let's create our own graphics demo in Scratch!

SKETCH OUT THE DESIGN

The first step of turning an idea into a Scratch program is to sketch out what you want it to look like. Planning your program helps you figure out the sprites you want and how they'll behave. I recommend drawing your ideas on paper so you can cross them out if you don't like them and write down notes and reminders.

It's also best to keep the project simple. If you start by making a complicated game like *Minecraft* or *Zelda*, you'll quickly realize that it's a lot of work. It's more rewarding to finish a small project than to deal with the frustration of an unending, unfinished, and unplayable game.

When you've completed your simple game, you can then build on top of it to make it more complex, which is the idea behind *iterative development*. First, you make the program work. Then, you make the program better. You can always add cool elements to the basic program after you finish it. Or, if your program becomes too complicated, you can return to your sketches and figure out what you want to remove.

Don't worry about making the sketch look nice. It's more important to have a solid plan for the main parts of the program. In my sketch, I have three parts: A, B, and C. We will work on these parts one at a time until we've built the full program.

A Create the space backdrop.

B Create three bouncing dots.

C Draw the rainbow lines.
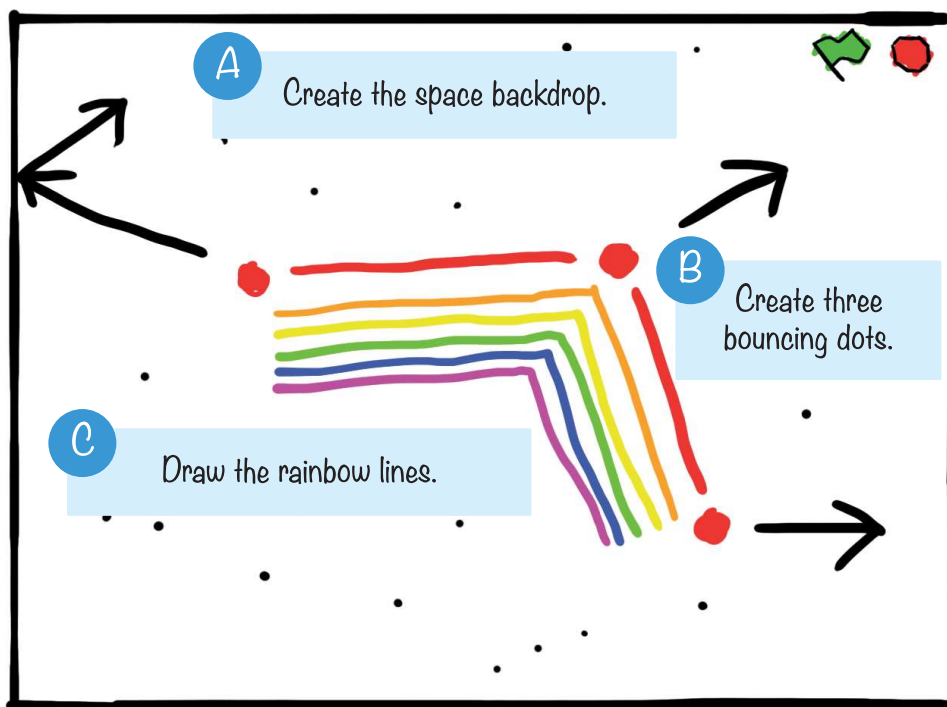
After you've completed a sketch of what you want your program to do, you can start programming! Go to *https://scratch.mit.edu/*, sign up for an account on the site, and log in (having an account lets you save your programs on the Scratch website). After you've logged in, click the **Create** button at the

top of your screen to start making your own Scratch project. Then click the text field in the top left to change the name of the project from *Untitled* to *Rainbow Lines*. Let's begin by tackling Part A of the sketch.
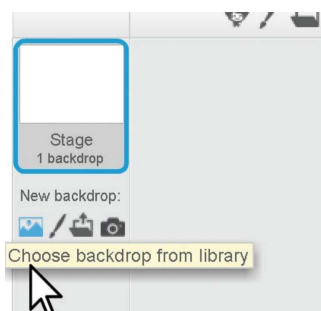
Ⓐ CREATE THE SPACE BACKDROP

First, let's clean up sprites we won't use and set a background.

### 1. Clean Up and Set the Stage

Every time you create a new Scratch project, you'll see an orange cat sprite on a blank, white Stage. We don't need the cat sprite for our program, so right-click the `Sprite1` cat in the Sprite List, and select **delete** to remove the cat from the Stage and the Sprite List.
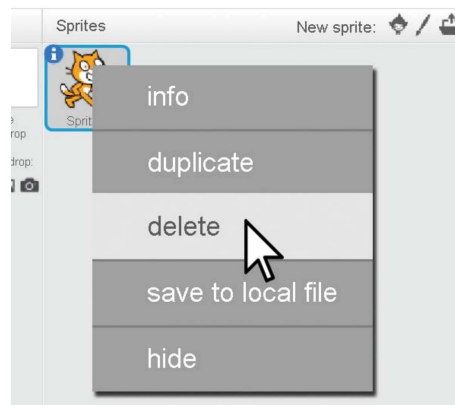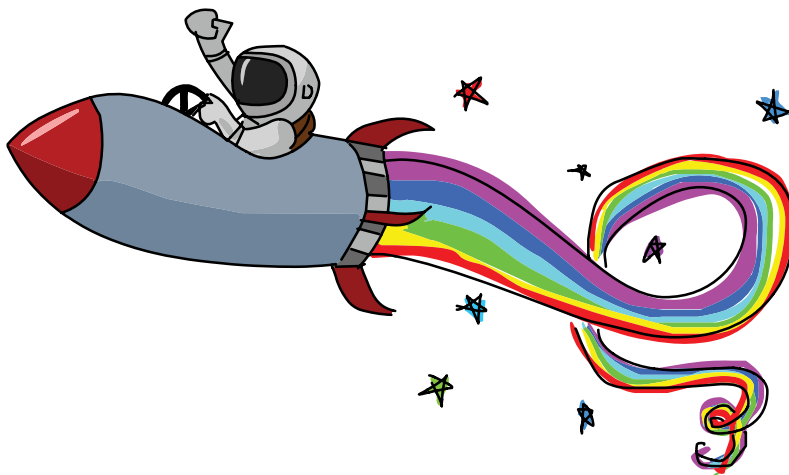

Click the **Choose backdrop from library** button (which looks like a landscape painting) under New backdrop.

The Backdrop Library window will open and display all the backdrops in alphabetical order. Select the **stars** backdrop and click **OK**.
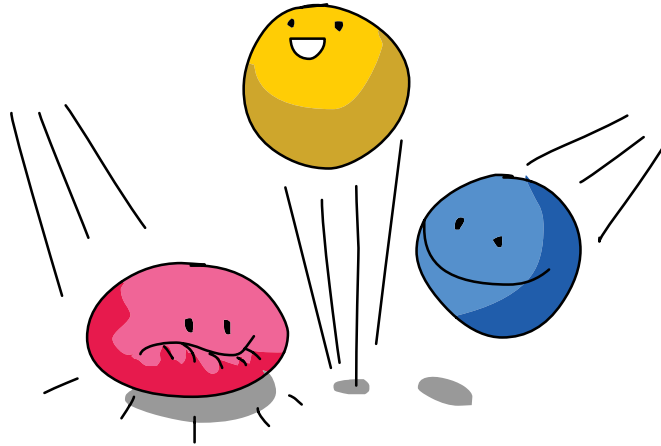
1 Make sure All is selected, or the stars backdrop won't show up.

2 Select the stars backdrop.

3 Click OK to finish.
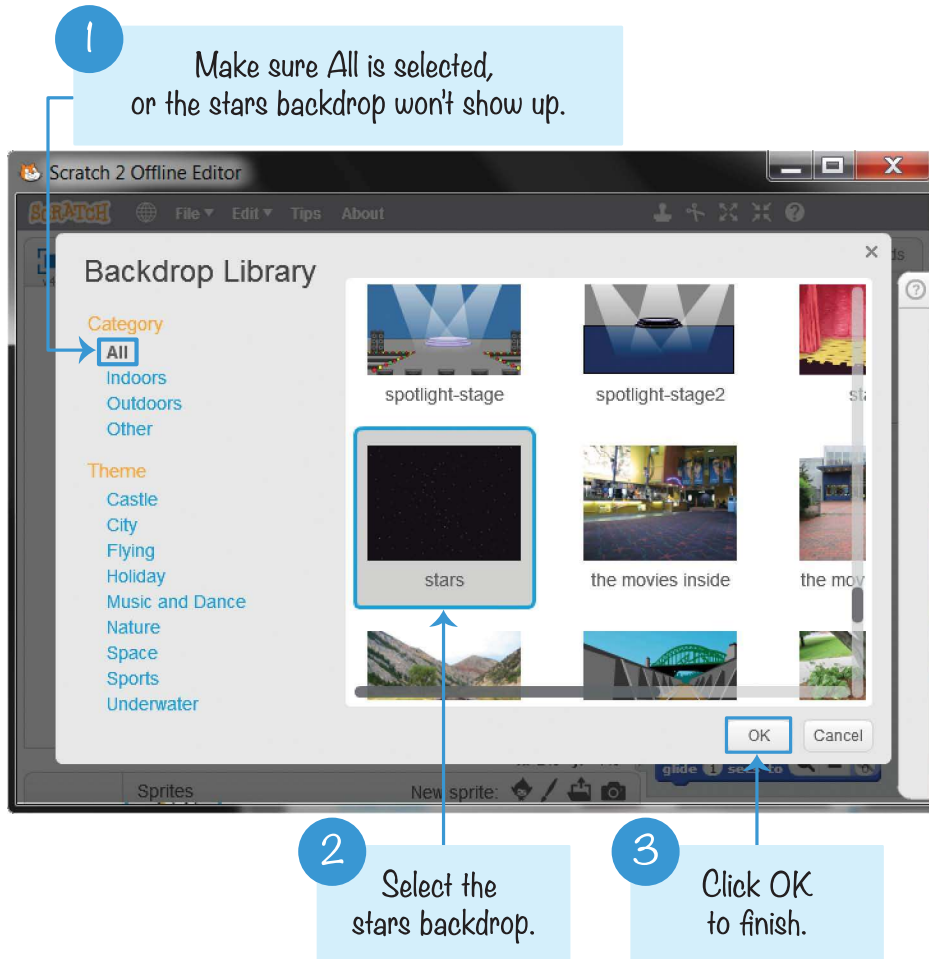
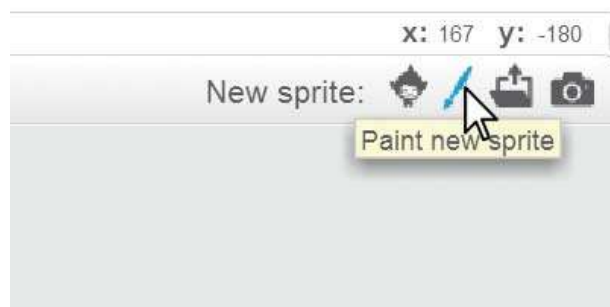Now the Stage's backdrop looks like outer space!

B CREATE THREE BOUNCING DOTS

Next, we'll add three new sprites that represent the three points of the flying *V.*


2. Paint the Dot

Click the **Paint new sprite** button (which looks like a paintbrush) next to New sprite.

A new sprite named `Sprite1` is created in the Sprite List. Clicking this button also switches to the Costumes tab, which contains the Paint Editor. Use the Brush tool to draw a small red dot near the crosshairs in the Paint Editor. It might help to zoom in by clicking the Zoom button (which looks like a magnifying glass) in the Paint Editor.
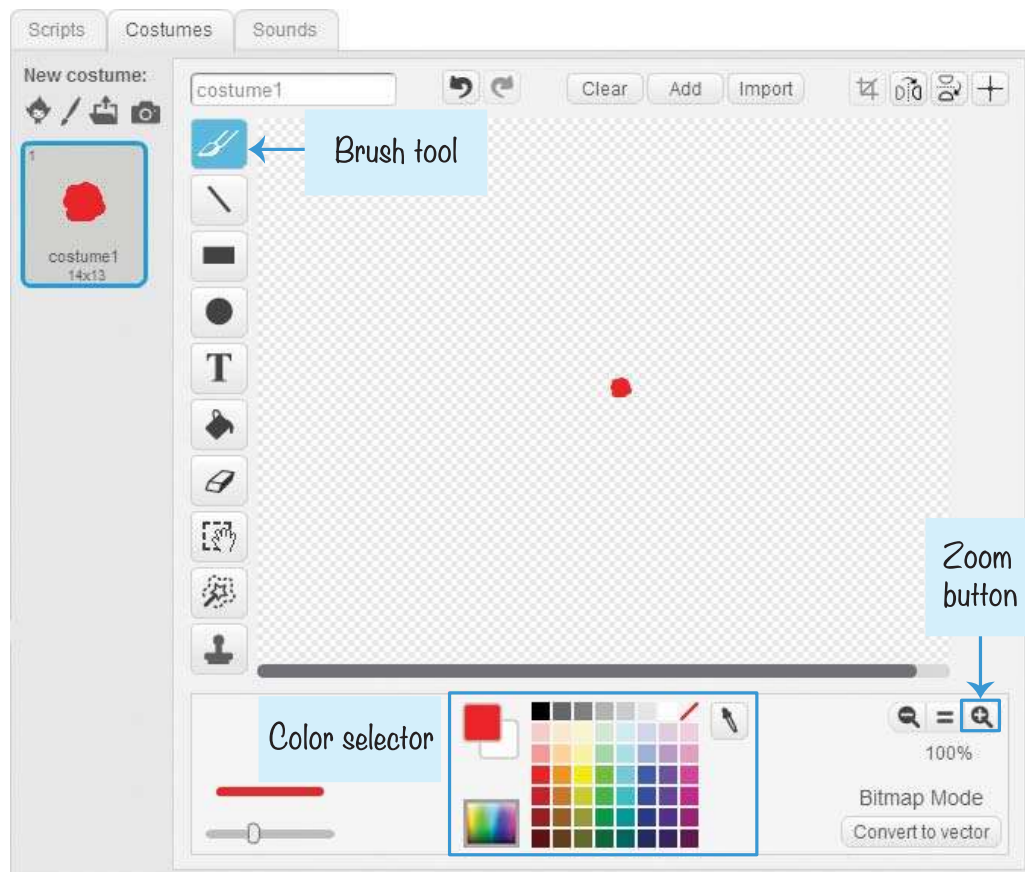
Click the ⓘ button for the `Sprite1` dot sprite to open its Info Area. (You can also open the Info Area by right-clicking the sprite and selecting **info**.)

Change the sprite's name from Sprite1 to Dot 1. Then click the ◀ button to close the Info Area and display the Sprite List again.

Type the new name here.
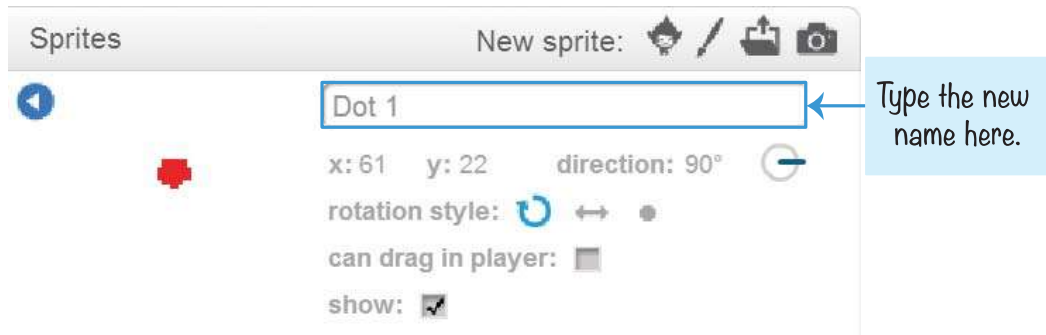
3. Add Code for the Dot 1 Sprite

Now we can start programming. Click the **Scripts** tab to make the Scripts Area visible. Add the following code to the Scripts Area. You can find these blocks in the *Events* (brown), *Motion* (dark blue), *Operators* (green), and *Control* (yellow) categories. If you have trouble understanding how to drag these blocks, view the animation at *https://www.nostarch.com/scratchplayground/*.
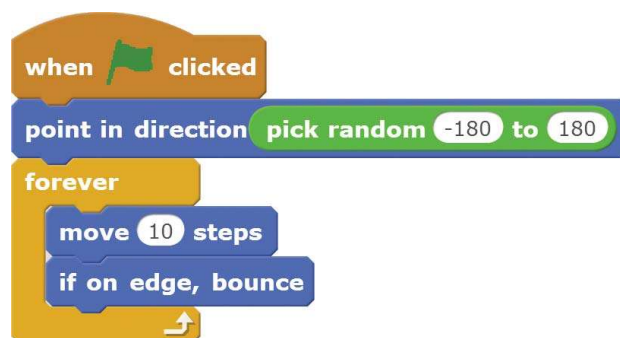

When you click the green flag, the Dot 1 sprite points in a random direction between −180 and 180 degrees. Then, the **forever** loop moves the sprite forward 10 steps and makes the sprite bounce when it hits the edge of the Stage. The sprite will continue to do this forever.

Notice that the Dot 1 sprite isn't drawing any rainbow lines yet. We'll do that later when we've created more sprites.

EXPLORE: DIRECTION AND DEGREES

Words like *up* or *right* are perfectly understood as directions by humans like you and me. But the computer needs a number to indicate an exact direction. All sprites in Scratch have their own direction number. The direction numbers are between −180 and 180 degrees. Pointing at 0 degrees is facing up. Pointing at 90 degrees is facing to the right. The following figure shows several directions and their degrees. Notice that the degrees increase in the clockwise direction and decrease in the counterclockwise direction. Also, notice that −180 and 180 degrees point in the same direction: down.
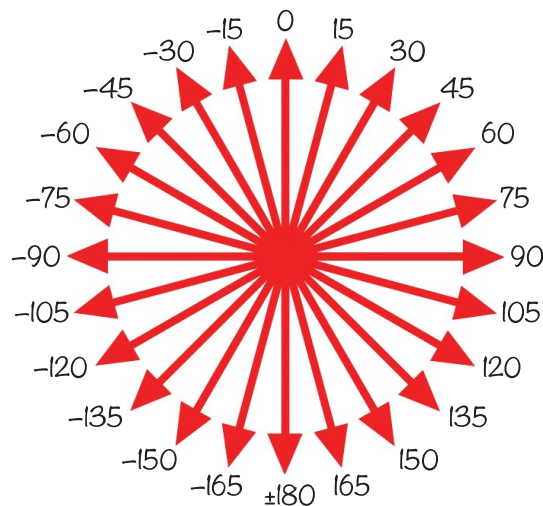

The **pick random -180 to 180** block chooses a random number to use as the direction. Then the **point in direction** block points the sprite in that direction. This means that the sprite could be pointed in any possible direction.

Let's write a new script that demonstrates how degrees work. Open a new tab by pressing CTRL-T in your web browser, and go to *https://scratch.mit.edu/* to open a new Scratch editor. You can edit multiple Scratch programs at the same time.

(continued)

In the Scripts Area for the cat sprite named `Sprite1`, add the following code using blocks from the *Events* (brown), *Control* (yellow), *Motion* (dark blue), and *Looks* (purple) categories. Keep in mind that we're writing a totally separate program from the *Rainbow Lines* program!
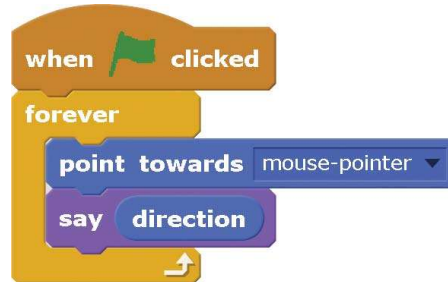
When you run this program, the cat sprite points toward the mouse. The cat will say the direction in which it's pointing.
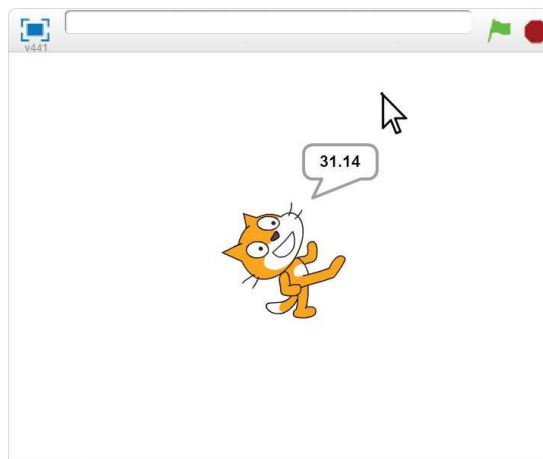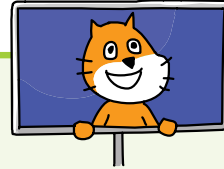

Notice that the direction number changes as the cat's direction changes.
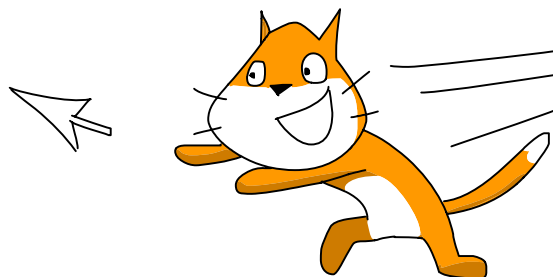
4. Duplicate the Dot 1 Sprite

Right-click the `Dot 1` sprite in the Sprite List and select **duplicate**. Do this twice so that you make two duplicates: `Dot 2` and `Dot 3`. (Scratch automatically names the sprites with increasing numbers.)

SAVE POINT

Click the green flag to test the code so far. Check that all three dots are moving and bouncing around the Stage. When you duplicated the sprites, you also duplicated the sprite code. Click the red stop sign and save your program.

C DRAW THE RAINBOW LINES

Now that we've created all the bouncing dots, we can create a fourth dot sprite to draw the rainbow lines. We'll write a program that makes this drawing dot move quickly between the three bouncing dot sprites, drawing a line as it moves. This process will repeat three times, and then after 10 seconds the screen will clear.

5. Add the Code for the Drawing Dot Sprite

Right-click one of the bouncing dot sprites and select **duplicate**. Because this is a duplicate of the bouncing dots, it has some code we need to delete. In its Scripts Area, delete all the code blocks by right-clicking the **when green flag clicked** block and then selecting **delete**. You can also delete blocks by dragging them to the Blocks Area in the middle of the editor, where they'll disappear.

Click the ℹ button and rename this sprite Drawing Dot.
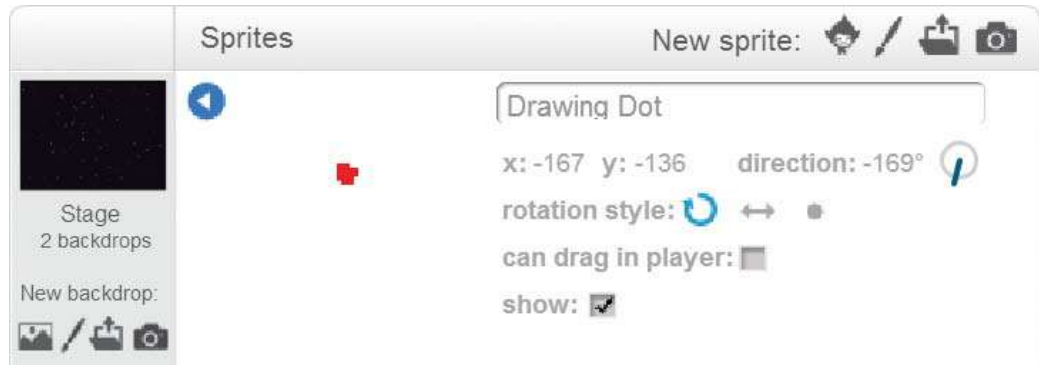
Add the following two scripts to the Drawing Dot sprite. You can find these blocks in the *Events* (brown), *Pen* (turquoise), *Control* (yellow), and *Motion* (dark blue) categories.
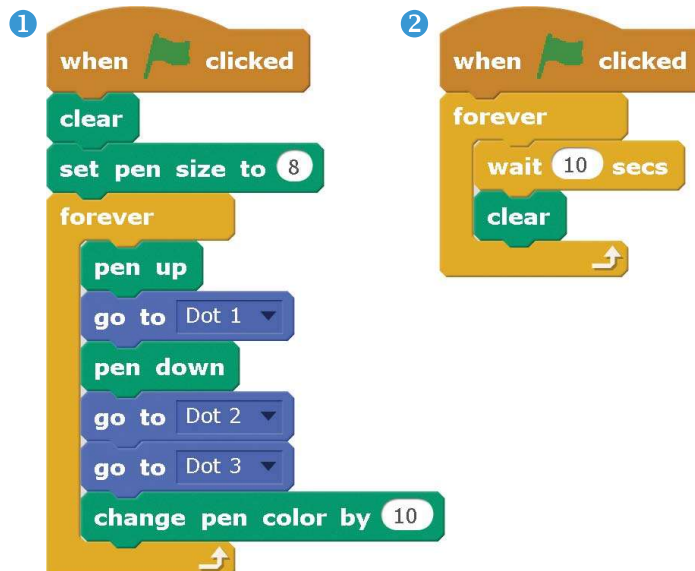
In script ❶, make sure you use the **go to** block, not the **go to x y** block. Also, be sure to change the **go to** blocks so they aren't going to the mouse-pointer. To do so, click the black triangle on the block and select a sprite from the menu.
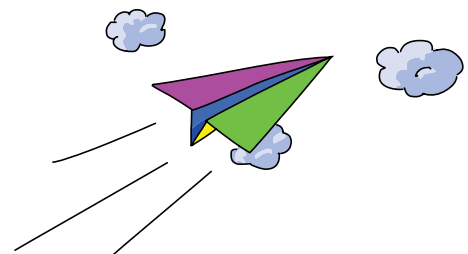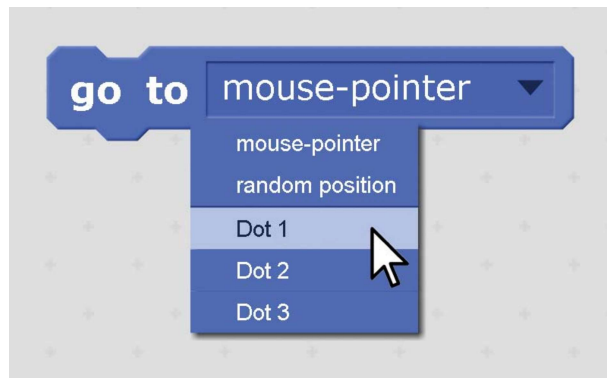
Before you run the code, let's talk through how it works. When you click the green flag in script ❶, the Drawing Dot sprite runs the **clear** block to clear away any pen drawing already on the Stage. Then the script runs the **pen down** block: as the sprite moves around, it draws a line on the Stage.

To better understand what the **pen down** block does, imagine holding down a marker on a piece of paper while you walk around it: a line would be drawn following you! The drawing dot goes to Dot 1, puts its pen down, goes to Dot 2, and then goes to Dot 3. Next, the **change pen color by 10** block changes the color of the pen slightly. (You can increase this number to make the colors change faster.) At the same time, Dot 1, Dot 2, and Dot 3 sprites continue to move around on their own. So the *V* line that the Drawing Dot sprite draws also moves around.

Script ❷ is a lot simpler to understand. This code waits 10 seconds and then clears the screen of any marks made by the *Pen* blocks. Now the Stage won't become overcrowded with rainbow lines.
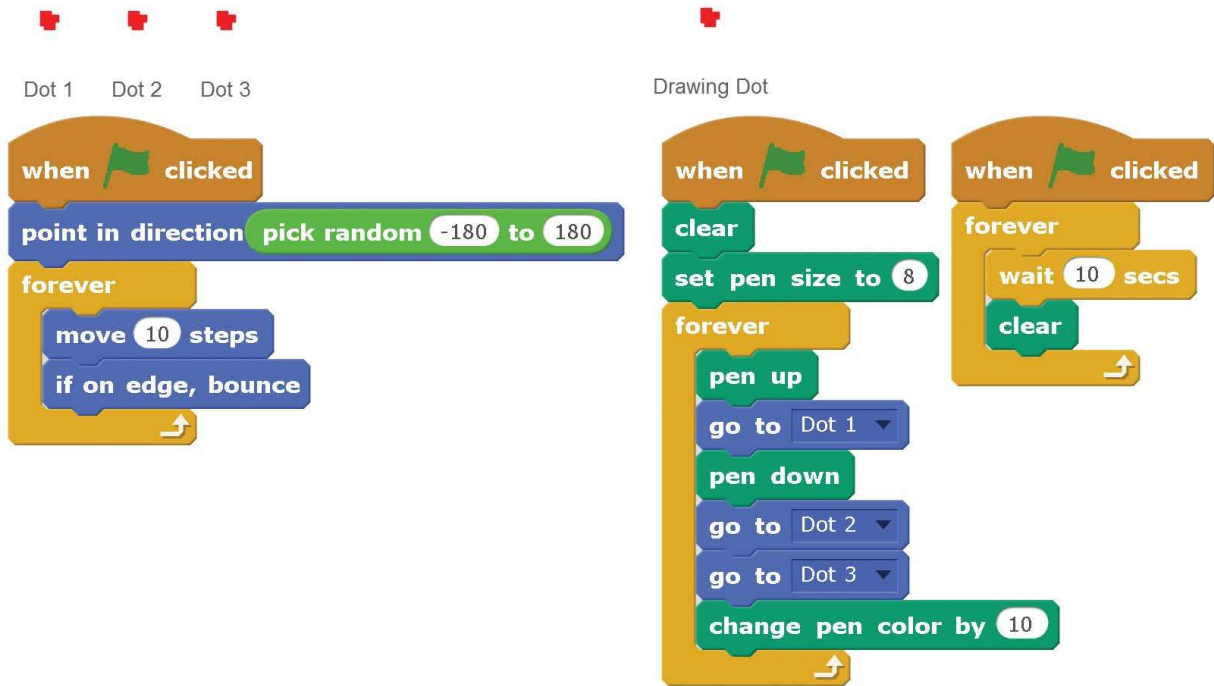
SAVE POINT

Click the green flag to test the code so far. You should see a flying rainbow *V* move across the Stage. Then, every 10 seconds, the rainbows clear. Click the red stop sign and save your program!
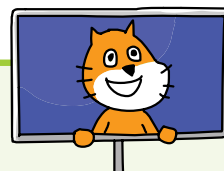
THE COMPLETE PROGRAM

The final code for the entire program is shown here. Notice that the code for Dot 1, Dot 2, and Dot 3 sprites is identical. If your program isn't working correctly, check your code against this code:

Dot 1   Dot 2   Dot 3

```
when [green flag] clicked
point in direction (pick random (-180) to (180))
forever
    move (10) steps
    if on edge, bounce
```

Drawing Dot

```
when [green flag] clicked
clear
set pen size to (8)
forever
    pen up
    go to [Dot 1 ▼]
    pen down
    go to [Dot 2 ▼]
    go to [Dot 3 ▼]
    change pen color by (10)
```

```
when [green flag] clicked
forever
    wait (10) secs
    clear
```

TURBO MODE

If you hold SHIFT while clicking the green flag, you can start the program in Turbo Mode. The computer is usually able to run code blocks quickly, but a program that draws sprites to the screen slows down the computer. In Turbo Mode, instead of drawing the screen after each code block, Scratch only
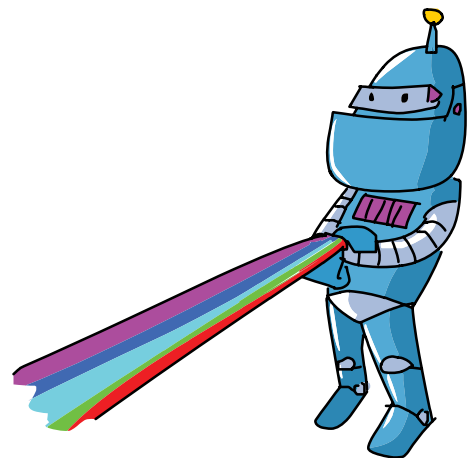
draws to the screen after several code blocks. Human eyes won't notice the skipped drawings, and the program will look like it's running faster.

SHIFT-click the green flag to run the *Rainbow Lines* program in Turbo Mode. Almost instantly, the screen fills up! To end Turbo Mode, SHIFT-click the green flag again.
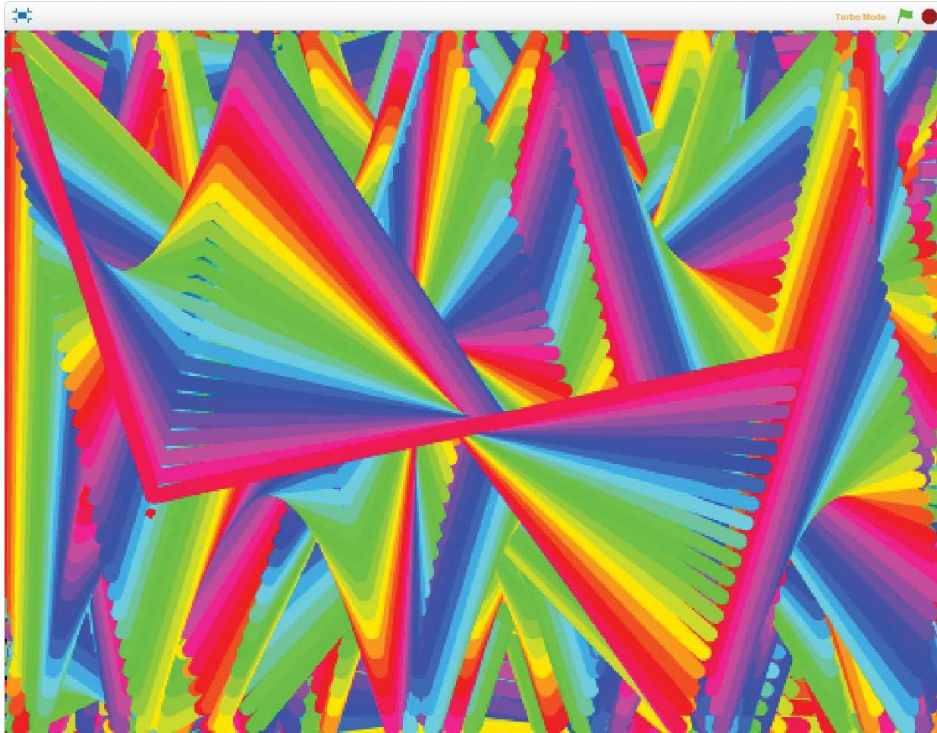
VERSION 2.0: RAINBOW TRIANGLES

Your *Rainbow Lines* program is complete as it is, but you can take this program to the next level. You can always think of new ideas to add to your programs after you have the basics working.

For version 2.0 of the program, let's connect Dot 3 and Dot 1 so that, instead of a flying rainbow *V*, the program has a flying rainbow triangle. Modify the code for the Drawing Dot sprite to match the following, but leave the rest of the code the same.
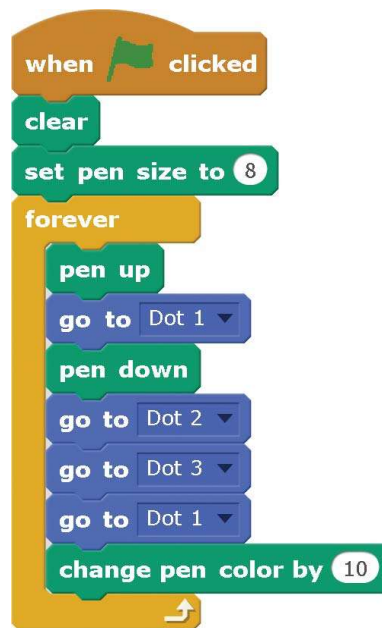
The new **go to Dot 1** block draws a line from Dot 3 to Dot 1, forming a triangle.
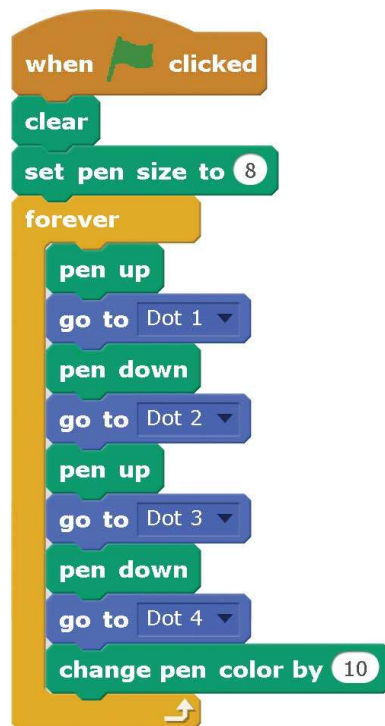

SAVE POINT

Click the green flag to test the code so far. Make sure you see a flying rainbow triangle moving around the Stage. Then click the red stop sign and save your program.

VERSION 3.0: TWO RAINBOW LINES

For version 3.0, let's create two separate flying lines instead of a single line.

Right-click the Dot 3 sprite and select **duplicate** to make a Dot 4 sprite. Then modify the Drawing Dot sprite's code to match the following.

```
when 🏳 clicked
clear
set pen size to 8
forever
    pen up
    go to Dot 1 ▾
    pen down
    go to Dot 2 ▾
    pen up
    go to Dot 3 ▾
    pen down
    go to Dot 4 ▾
    change pen color by 10
```

The new code puts the pen down while moving between Dot 1 and Dot 2. Then it lifts the pen up, goes to Dot 3, and puts the pen down again to draw a line from Dot 3 to Dot 4.

SAVE POINT

Click the green flag to test the code so far. Make sure you see two flying rainbow lines. Then click the red stop sign and save your program.
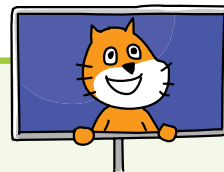
VERSION 4.0: YOU DECIDE!

You can start making your own changes to create new demo-scene programs. In this chapter's program you drew lines, but if you learn the mathematics behind Bézier curves, you can make beautiful curves. Another mathematical concept that produces beautiful images is fractals, as shown in the

following figure. Go to the Scratch website and search for "bézier," "fractals," or "demoscene" to get ideas for your own programs. You can also look at the code blocks for every Scratch program by clicking the **See inside** button on the program's web page. Check out some example projects at *https://www.nostarch.com/scratchplayground/.*
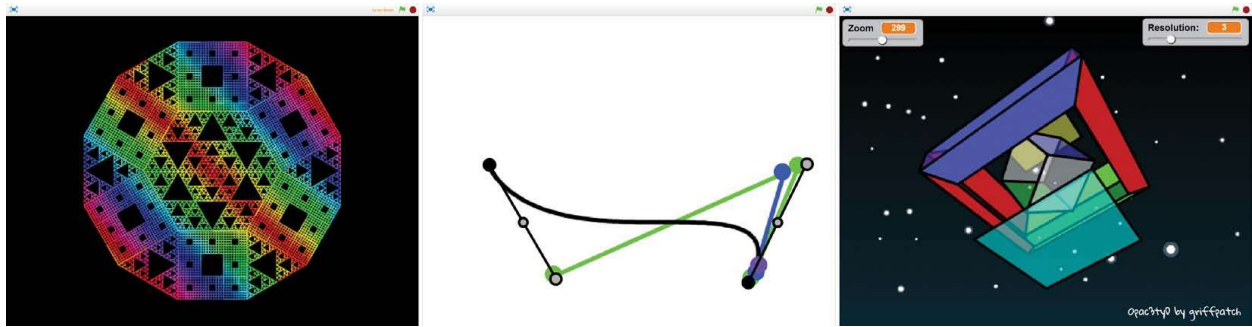
SUMMARY

In this chapter, you built a project that

▶ Has custom sprites that you created (even if they are just dots)

▶ Uses the **pick random** block to point a sprite in a random direction

▶ Makes sprites move and bounce off the edges of the Stage

▶ Duplicates sprites and their code

▶ Uses the *Pen* blocks to draw rainbow lines

This project is a demo that users can watch but can't control. In Chapter 3, you'll make a maze game that lets players interact with the program by using the keyboard instead of just watching. This will be the first real game project in this book!