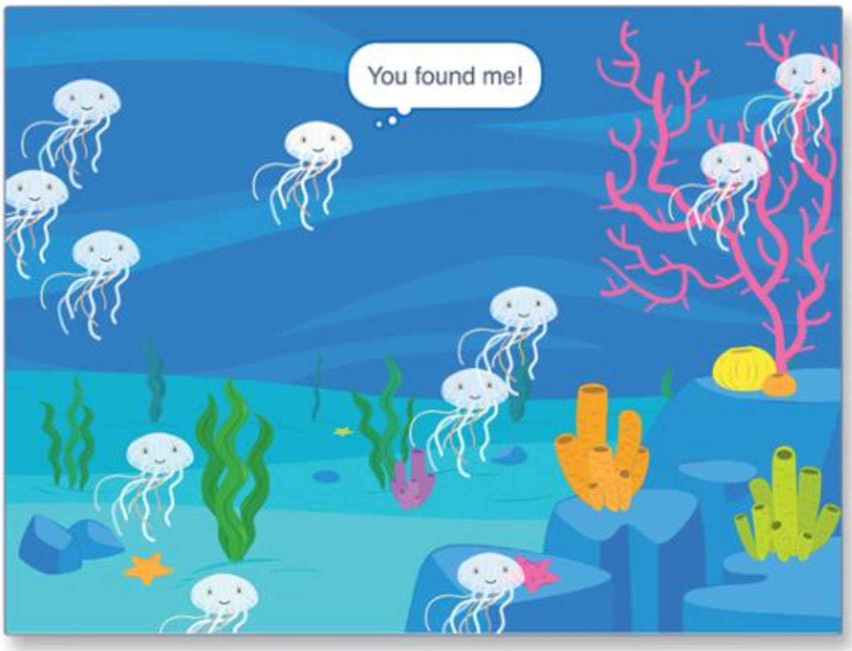
PROJECT 4

JELLYFISH JUMBLE



OH NO, THE JELLYFISH ARE ALL JUMBLED UP — CAN YOU FIND THE REAL MR. JELLY? In this project, you code a funny search-and-find puzzle featuring jellyfish on the ocean floor. First, you make a real Mr. Jelly. Then, by using *repetition* (doing something over and over), you make lots of *clones,* or copies, to create fakers. The fakers look just like Mr. Jelly — except, unlike Mr. Jelly, all the fakers are a little bit see-through! The fakers scatter to *random* places in the water — different places the computer creates for every puzzle — trying to confuse the player as she searches for Mr. Jelly. The player wins by finding and clicking the real Mr. Jelly. When Mr. Jelly is clicked, he announces, "You found me!" As you design and code your program, you add animation and sound effects to make the puzzle even more fun.

BRAINSTORM

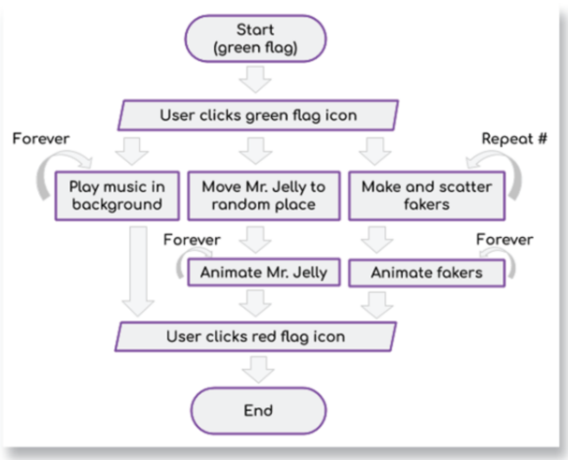
In this project, you work in Scratch to create your own search-and-find puzzle for viewing on a computer or tablet. Your scene uses assets from the Scratch libraries. You can create any setting and characters you want! This project uses the *ghost* effect to make the faker jellyfish varying degrees of see-through, but you may choose to use the *color* effect to create fakers of slightly different colors.

FLOWCHARTS

Plan your Jellyfish Jumble program by drawing flowcharts to show how the program will run. The flowcharts don't need to have every step; include just the main parts.

When I think about my program, I know that I want to make jellyfish who live in the ocean. To do this, I need to add a Jellyfish sprite (`Mr. Jelly`) and another Jellyfish sprite (`Fake`, which I will clone to make additional fakers). Then I will write code that executes when the user clicks the green flag icon.

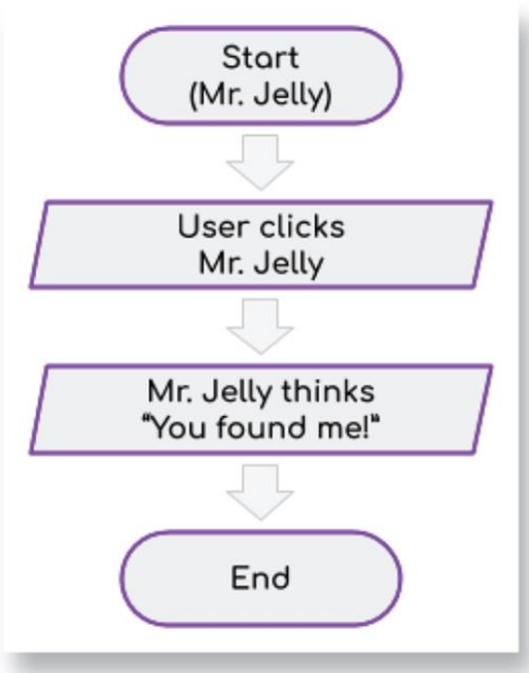
For this project, I need two flowcharts. One flowchart sets up the puzzle when the user clicks the green flag. The setup has different parts happening at the same time (in *parallel*).



Here are the parts of the green flag flowchart:

- **In the background:** Loop ocean sounds until the program stops.
- **Mr. Jelly sprite:** Send the real Jellyfish sprite, Mr. Jelly, to a random location onscreen and animate him.
- **Fake jelly:** Fake makes and scatters more fakers onscreen. The fakers animate.

The second flowchart is for searching for Mr. Jelly and clicking him when you find him. If Mr. Jelly is clicked by a user, he lets the user know that he's been found. No message appears if the user clicks a faker. Here are the parts of the Mr. Jelly clicked flowchart.



Now, get started coding your Jellyfish Jumble!

START A NEW PROJECT

Begin creating your Jellyfish Jumble program by starting a new project:

1. **Open Scratch at https://scratch.mit.edu.**
2. **On the Scratch home page, select Create.**

   Or if you're already working in Scratch, choose File ⇒ New Project from the menu bar. A new project opens.
3. **Name your program by typing a name in the project name field at the top of the Scratch interface.**

   I named my program `Jellyfish Jumble`.
4. **Cut (delete) Scratch Cat from the project by clicking or tapping the X in the Scratch Cat icon.**

   You can find the icon in the sprite area in the lower-right corner of the Scratch dashboard.

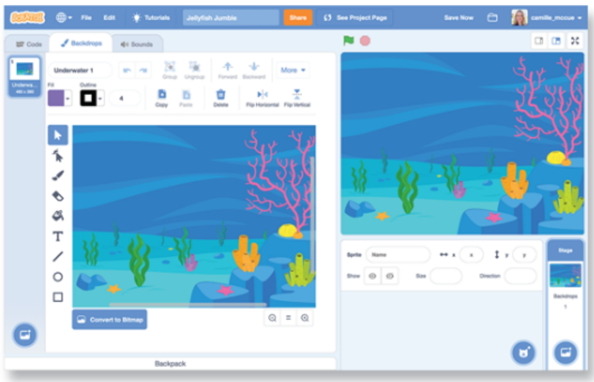
ADD A BACKDROP

The *backdrop* is the background color or image that fills the screen of your game. Add a backdrop as follows:

1. **At the stage, click the choose a backdrop icon.**

   The backdrop library appears on the Choose a Backdrop screen.
2. **From the list of backdrops, click or tap the backdrop you want.**

   Your backdrop appears on the stage. I chose the Underwater 1 backdrop for my program.



ADD A JELLYFISH SPRITE AND CUSTOM COSTUMES

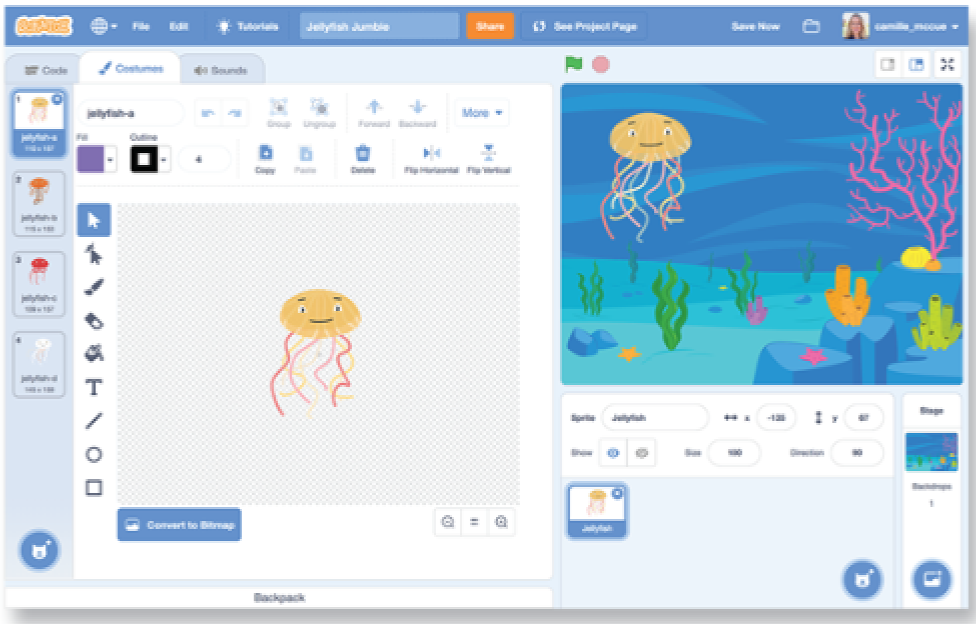
Your puzzle program has Jellyfish sprites, and each has costumes that you can change. Changing costumes over time animates the jellyfish, making them appear to swish their tentacles in the water.

Add a Jellyfish sprite to your scene and create its costumes by following these steps:

1. **In the sprite area of the Scratch interface, click the choose a sprite icon.**

   The sprite library appears on the Choose a Sprite screen.
2. **In the list of sprites, click or tap the sprite you want.**

   I picked Jellyfish. Your sprite appears on the stage.
3. **Click the Costumes tab to open the costume editor. Here, you can view all the costumes that come with the sprite.**



The tan costume is currently selected, but I will use only the white costume, which is named jellyfish-d by default. (I think it looks most like a jellyfish that once floated past me in the surf at Padre Island in Texas!)

Cut the costumes you don't want to use by clicking the X in the corner of each costume icon.

You should cut the tan, orange, and red costumes. These are named jellyfish-a, jellyfish-b, and jellyfish-c. This leaves only the white jellyfish costume.

Duplicate the white costume by Control-clicking (Mac) or right-clicking (Windows) its icon and selecting Duplicate from the pop-up menu.

You now have two white jellyfish costumes.

In the costume editor, on one of the costumes — it doesn't matter which one — click or tap the Ungroup button.

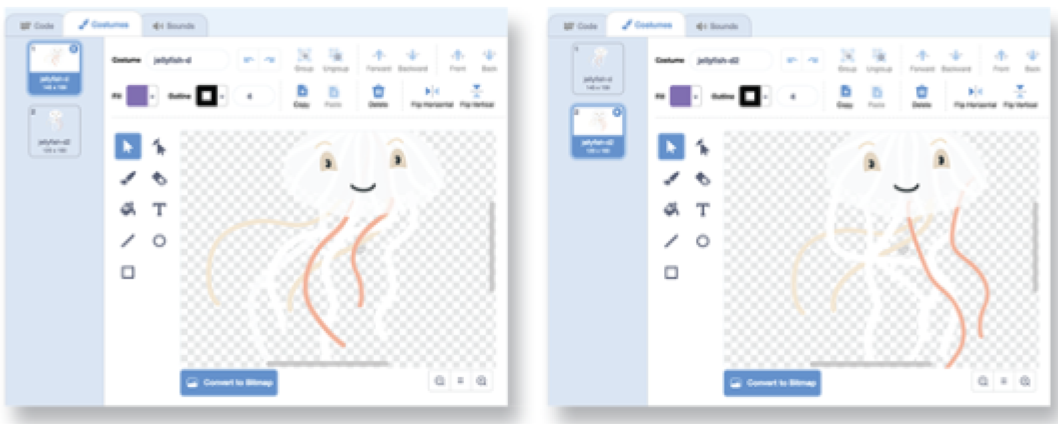
This ungroups the different body parts of the jellyfish so you can edit them individually.

Click a tentacle to select it and then edit its appearance and size.

Drag the tentacle to a slightly different position, drag a sizing dot to shrink or increase its size, or turn its rotation handle (or do all of these).

Repeat Step 7 to change the appearance of a few more tentacles.

The goal is to make the tentacles look a little different on the two costumes.



MAKE A MR. JELLY SPRITE AND A FAKE SPRITE

Now that you have a Jellyfish sprite that looks the way you want, you can use it to create the jellyfish characters for your puzzle. You will now make and name two sprites: Mr. Jelly and Fake. (Later, Fake will be cloned to make lots of fakers, also known as *distractors* — objects that make it hard for the user to find the real Mr. Jelly.)

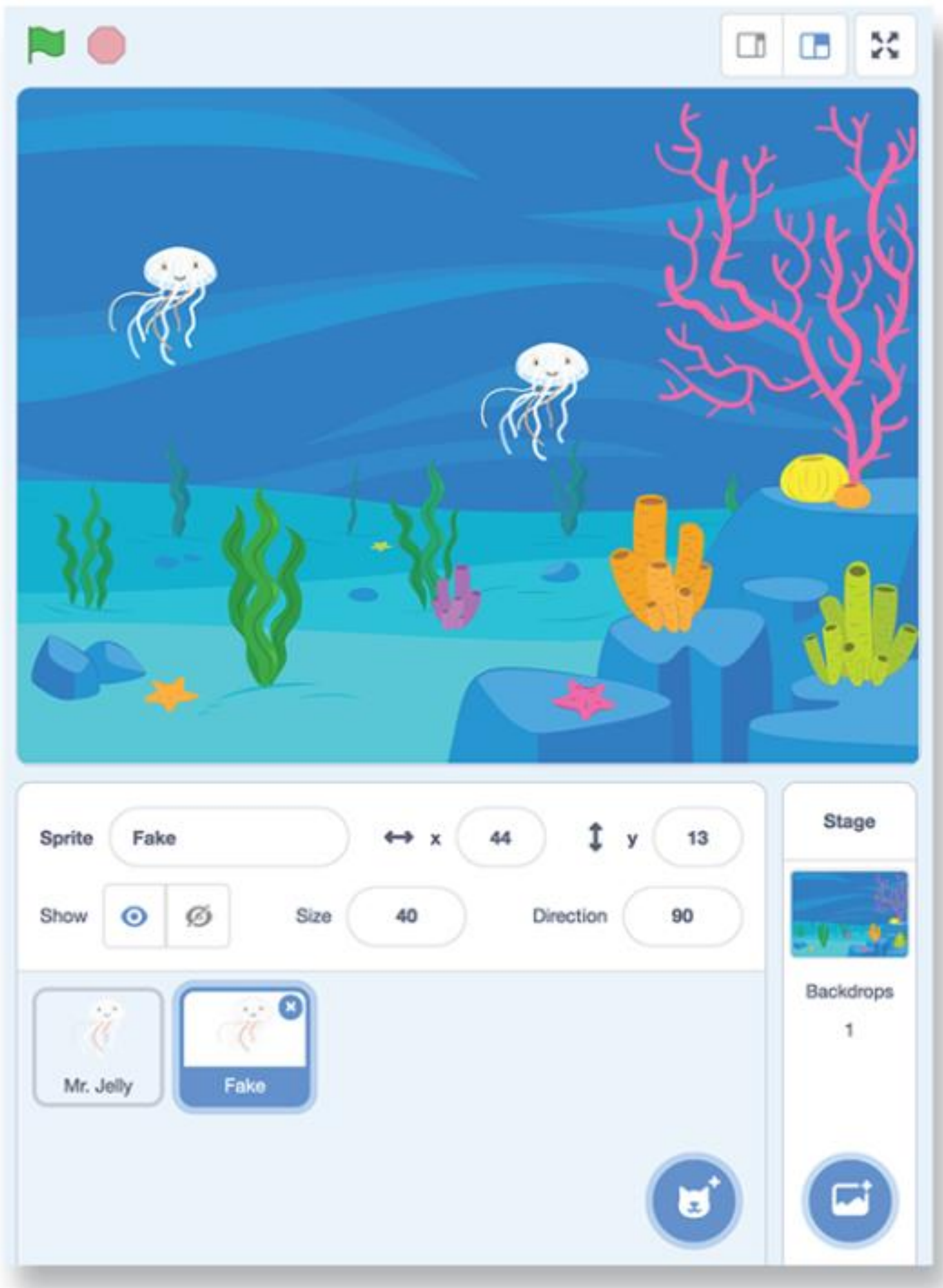
1. **Continue working in the sprite area of the Scratch interface. Be sure you have the Jellyfish sprite selected.**
2. **Resize the Jellyfish sprite by typing a new number in the Size field above the sprite.**

   The default (starting) size of a sprite is `100`. I changed my jellyfish to size `40`.
3. **Type a new name in the Sprite field.**

   The default (starting) name is `Jellyfish`. I changed the name to `Mr.  Jelly`.
4. **In the sprite area, Control-click (Mac) or right-click (Windows) Mr. Jelly and select Duplicate from the pop-up menu. Change the name of the duplicate Jellyfish sprite to `Fake`.**

   Fake is an exact copy of Mr. Jelly — all the attributes, including costumes and size, are the same. (Don't worry about making Fake look different from Mr. Jelly — only the fakers need to look different because eventually you will hide Fake.) Both Jellyfish sprites, Mr. Jelly and Fake, now appear on the stage. An icon for each sprite also appears in the sprite area of the Scratch dashboard. The figure shows both Jellyfish sprites, sized and on the Underwater backdrop.

REMEMBER *If you add a sprite and then decide you don't want it, cut it by clicking or tapping the X in its icon.*



Sprite: Fake
x: 44
y: 13
Show
Size: 40
Direction: 90

Your user interface is now complete! The Jellyfish Jumble user interface has a stage with a backdrop and two Jellyfish sprites. You can click and drag the sprites to any position you like on the main stage — but the code you write later will move them to random positions anyway!

TECHNICAL STUFF *Scratch doesn't care what you name assets, variables, or new code blocks. But most other coding languages have rules about naming — they use camelCase naming. Like a camel, the name starts low (with a lowercase letter) and goes high (with an uppercase letter) each time you start a new word. So if you are naming a sprite, you could call it something like `mrJelly` or `fakeMrJelly`.*

CODE THE GREEN FLAG BLOCKS

The green flag code sets up the puzzle. You will write green flag code for the backdrop and both sprites.

BACKGROUND

The backdrop plays music that loops until the user clicks the stop icon or finds Mr. Jelly. Here's how to code it:

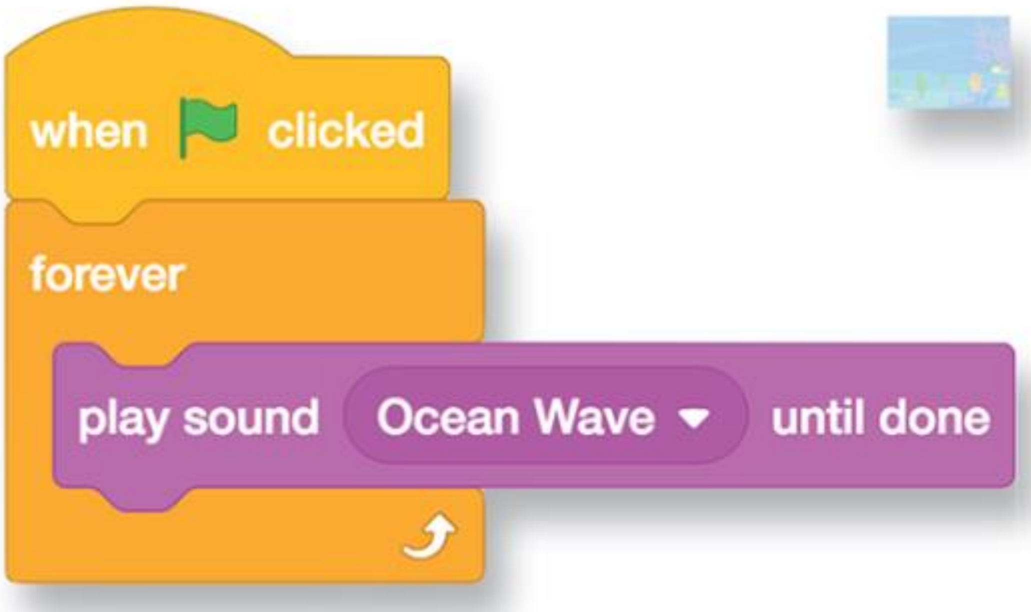
1. **Select the backdrop by clicking its icon on the stage.**
2. **On the Code tab of the Scratch interface, select the Events category. Drag a `when green flag clicked` event command to the code workspace.**

   The green flag command starts the code block.
3. **From the Control category, drag a `forever` loop command to the workspace and attach it to the previous command.**

   Any command placed inside the loop will repeat forever.
4. **Select the Sound icon. Drag the `play sound until done` command to the code workspace and attach it to the previous command.**
5. **Click or tap the down arrow on the `play sound until done` command tile and select the sound you want for the background.**

   I chose the `Ocean Wave` sound, which comes with the Underwater 1 backdrop.

   The code block for the backdrop is now complete!



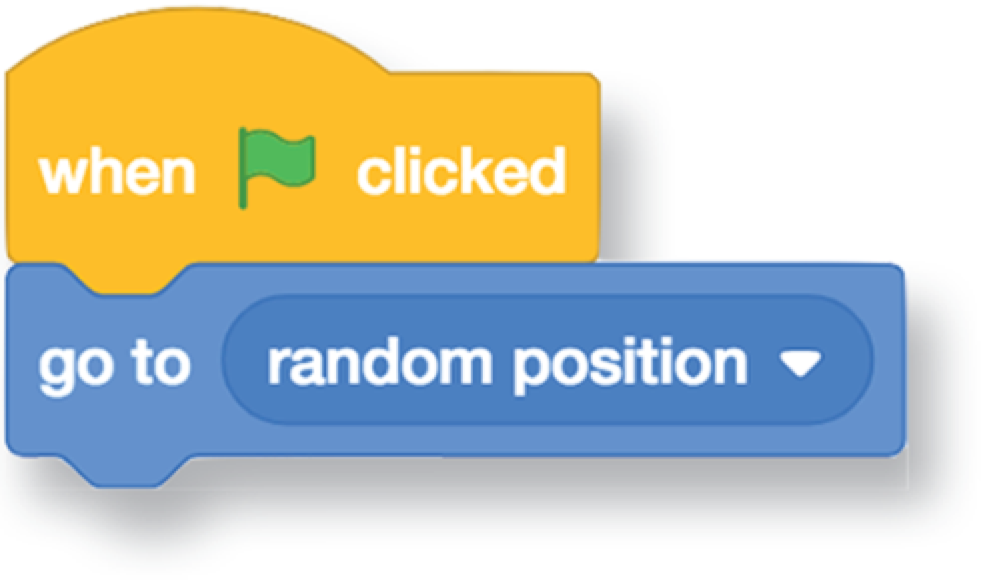
```
when 🏴 clicked
forever
    play sound Ocean Wave ▾ until done
```

MR. JELLY

When a user clicks or taps the green flag, Mr. Jelly should go to a random location and animate. Write the code like this:

1. **Select Mr. Jelly by clicking his icon in the sprite area.**
2. **On the Code tab of the Scratch interface, select the Events category. Drag a `when green flag clicked` event command to the code workspace.**

   The green flag command starts the code block.
3. **From the Motion category, drag a `go to random position` command to the workspace and attach it to the previous command.**



Now you will add a `forever` loop to give Mr. Jelly an animated swish. The swish will happen every second, when he changes between his two costumes.

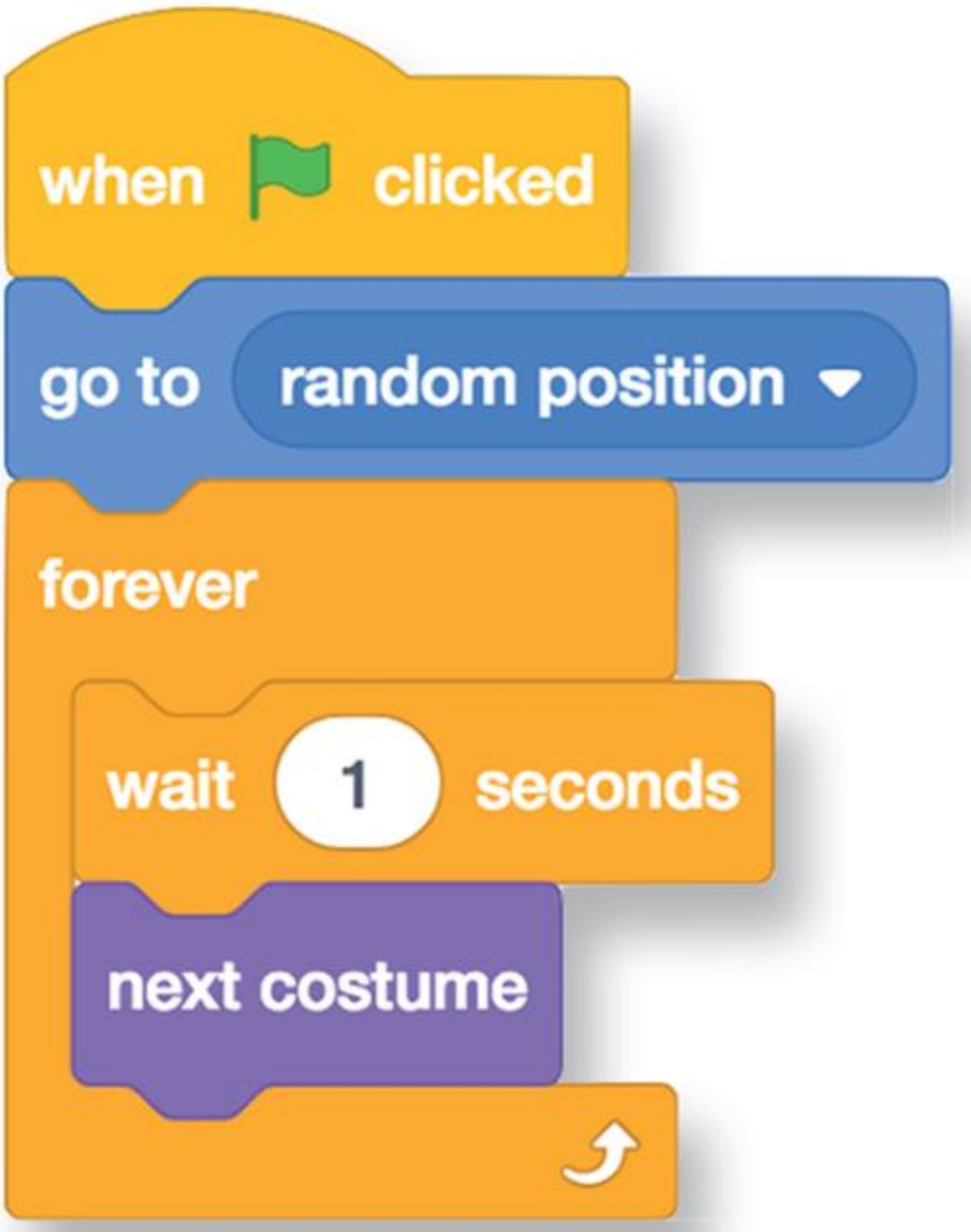
1. **From the Control category, drag a `forever` loop command to the workspace and attach it to the previous command.**

   Any command placed inside the loop will repeat forever.

2. **From the Control category, drag a `wait 1 seconds` command to the code workspace and attach it inside the `forever` loop.**

3. **From the Looks category, drag a `next costume` command to the code workspace and attach it to the previous command, inside the `forever` loop.**

   Here is the final green flag code block for Mr. Jelly.



```
when green flag clicked
go to (random position ▾)
forever
    wait (1) seconds
    next costume
```

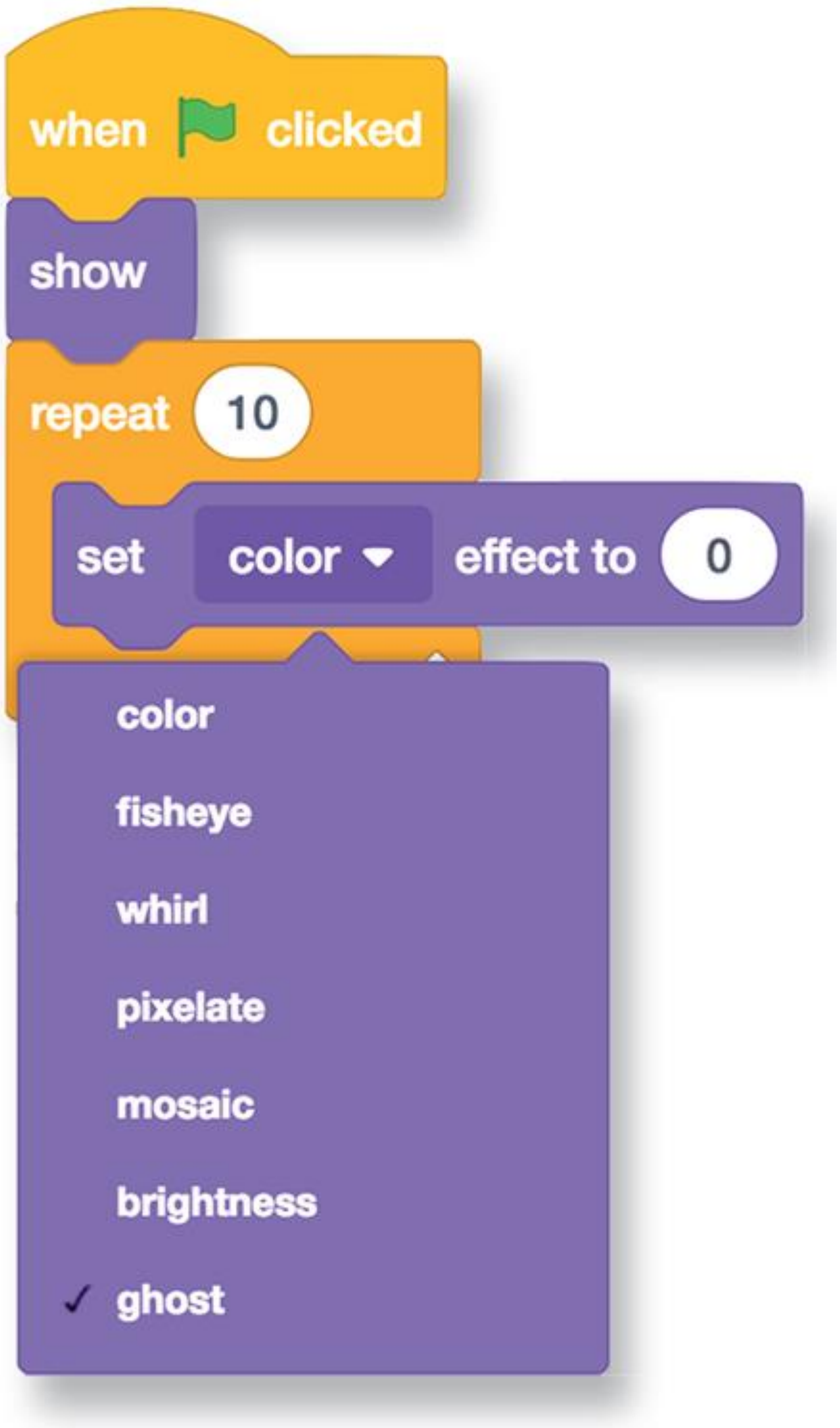
FAKE AND THE FAKERS

When a user clicks or taps the green flag, Fake goes into action. Fake uses a ghost effect to appear a little see-through, and then he clones to put a copy of himself in the water. Last, Fake moves to a new, random location in the water and repeats the ghosting and cloning. By doing ten `repeat` loops, Fake fills up the ocean with fakers. Write the code like this:

1. **Select Fake.**
2. **On the Code tab of the Scratch interface, select the Events category. Drag a `when green flag clicked` event command to the code workspace.**
3. **Select the Looks category. Drag a `show` command to the code workspace.**

   The fake jellyfish must show (appear) before you can clone it. This matters only if it is hidden at some other time (which it is — as you see in a few steps).
4. **From the Control category, drag a `repeat 10` loop command to the workspace and attach it to the previous command.**

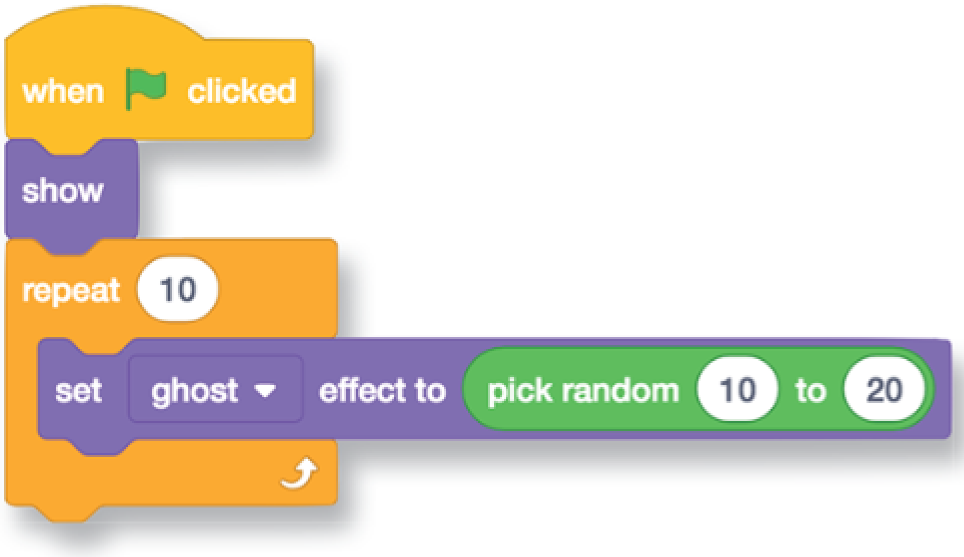
   Any commands placed inside the loop will be executed ten times. After the `repeat` loop, the code block will execute the next command in sequence.
5. **Select the Looks category. Drag a `set color effect to 0` command to the code workspace, placing it inside the repeat loop. Press the down arrow on the command tile to reveal the pop-up menu of effect options.**



6. **In the effects menu, change** `color` **to** `ghost`**.**

7. **From the Operations category, drag a `pick random` loop command to the workspace and place it inside the `set ghost effect` command, replacing the `0`. Type numbers to change the range of the random command.**

   You can change the random range to anything you want. I used `pick random 10 to 20`. Bigger numbers make the sprite more see-through. Here is the newly added ghost effect.


```
when green flag clicked
show
repeat 10
    set ghost effect to pick random 10 to 20
```

8. **From the Control category, drag a `create clone of myself` command to the workspace and attach it to the previous command, inside the `repeat` loop.**

   This clones Fake, who is wearing a random ghost effect. The clone is created at the same position Fake is located. (The clone is "stacked on top" of Fake.)

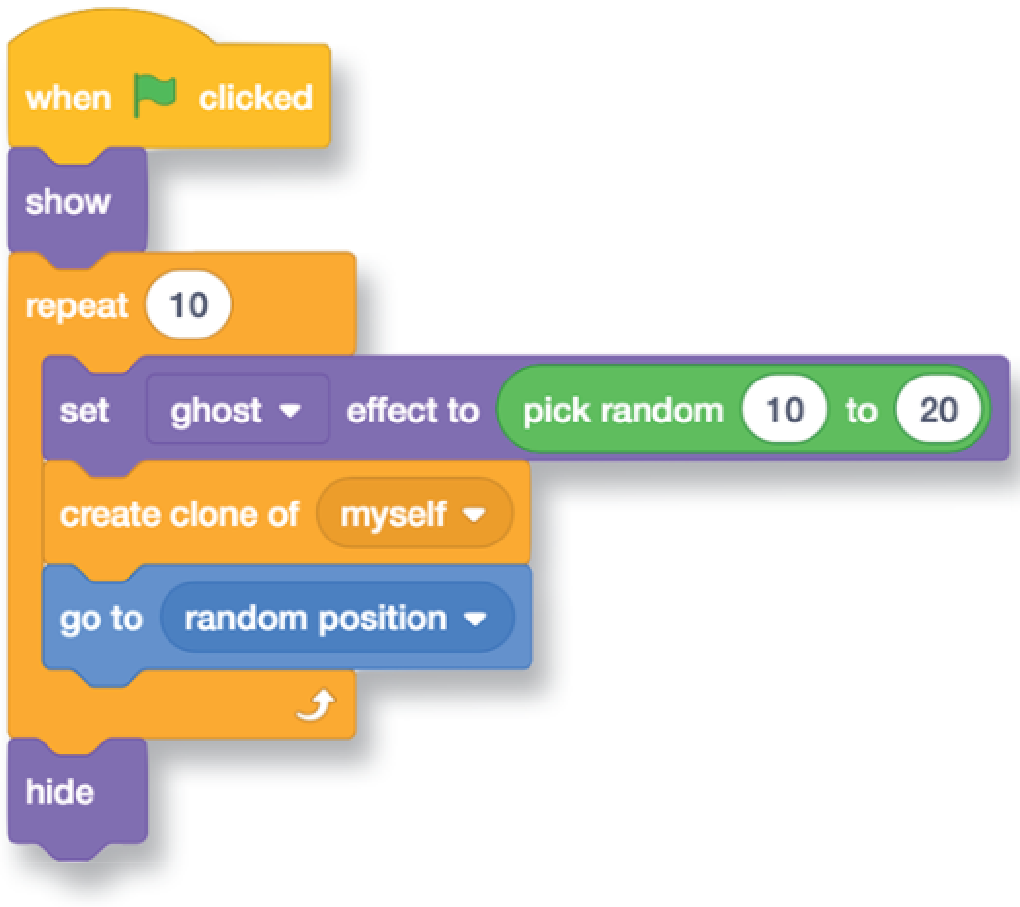
9. **From the Motion category, drag a `go to random position` command to the workspace and attach it to the previous command, inside the `repeat` loop.**

   This sends Fake to a new, random position. Fake leaves behind the clone he just made. The `repeat` loop is now complete.

10. **From the Looks category, drag a `hide` command to the workspace and attach it to the previous command.**

    The `hide` command hides Fake after the `repeat` loop completes. (By hiding Fake, you don't have to write extra commands to animate him later.)

You're all finished writing the green flag code for Fake. Now you just need to write a bit more code to finish the entire project.



```
when 🏳 clicked
show
repeat 10
    set ghost ▾ effect to pick random 10 to 20
    create clone of myself ▾
    go to random position ▾
hide
```

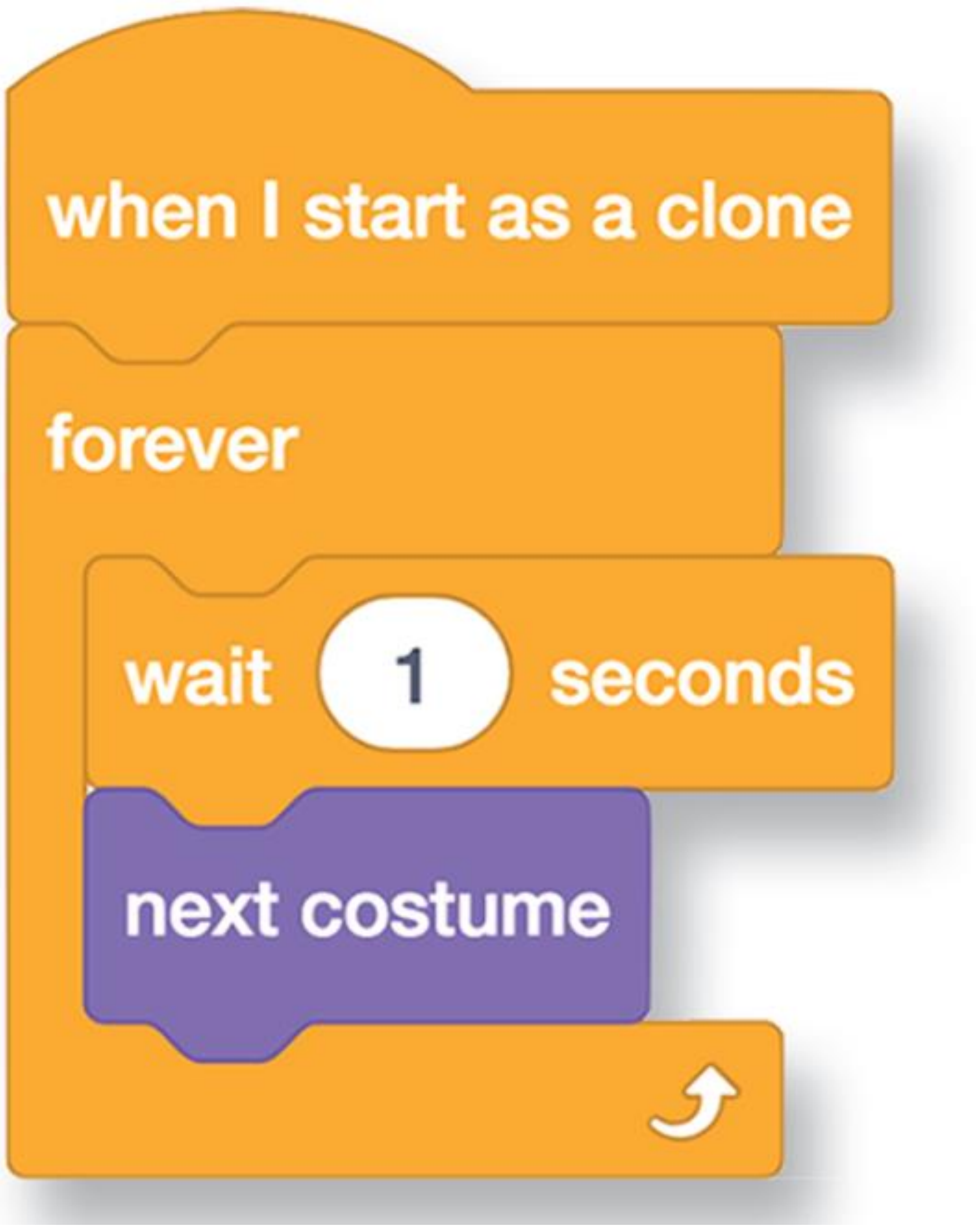
CODE THE FAKER CLONES TO SWISH

If you test your code so far, you'll see (after you get any bugs out!) that the puzzle setup mostly works. But you probably noticed that the only sprite swishing is Mr. Jelly. You need to animate the clones so that they swish, too. To do this, you have to write a separate code block for just the clones. Each clone will execute a `forever` loop in which it waits a second and then changes its costume, over and over. Here's how to code it:

1. **Continue working on Fake. From the Control category, drag a `when I start as a clone` header to the workspace to start a new code block.**
2. **From the Control category, drag a `forever` loop command to the workspace and attach it to the previous command.**
3. **From the Control category, drag a `wait 1 seconds` command to the code workspace and attach it inside the `forever` loop.**

4. **From the Looks category, drag a `next costume` command to the code workspace and attach it to the previous command, inside the `forever` loop.**

The `when I start as a clone` code block for Fake is finished.



```
when I start as a clone
forever
    wait 1 seconds
    next costume
```

Instead of showing and hiding Fake, you could add this same `forever` block of code to Fake to make him swish. Note that you have to do one or the other — either show/hide the sprite or add the animation loop to the end of the sprite's code. If you don't, the sprite will show but it won't swish like the clones.

CODE MR. JELLY TO KNOW HE'S BEEN FOUND

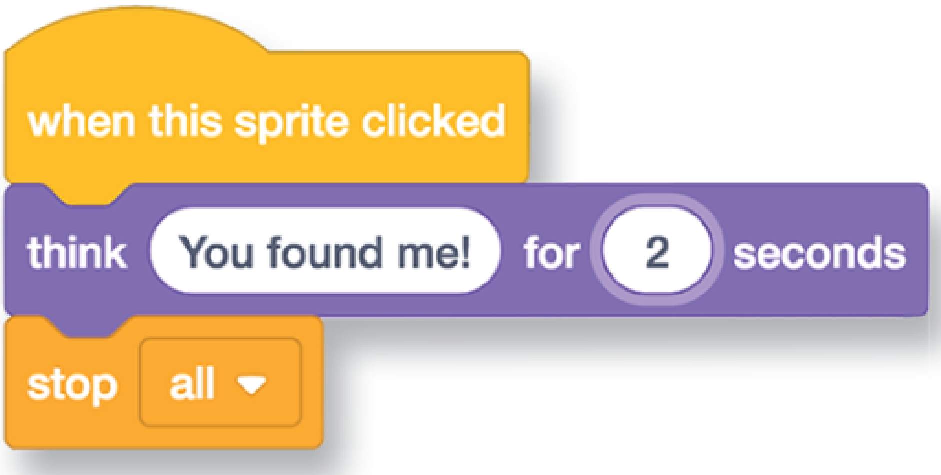
Test your code again. You should see that the puzzle sets up correctly and that all the jellies onscreen (real and fake) are swishing. But if you find and click Mr. Jelly, nothing happens. You need to write a final block of code in which Mr. Jelly responds to being found and clicked (or tapped) by the user:

1. **Continue working on Mr. Jelly. From the Control category, drag a `when this sprite clicked` header to the workspace to start a new code block.**
2. **From the Looks category, drag a `think Hmm... for 2 seconds` command to the code workspace and attach it to the code block header. Change `Hmm...` to `You found me!`**

   You can leave the `2 seconds` as-is.
3. **From the Control category, drag a `stop all` loop command to the workspace and attach it to the previous command.**

   After Mr. Jelly thinks aloud that he has been found, the entire program stops. This has the same effect as the user clicking the red stop icon.

   The `when this sprite clicked` code block for Mr. Jelly is finished.



```
when this sprite clicked
think You found me! for 2 seconds
stop all
```

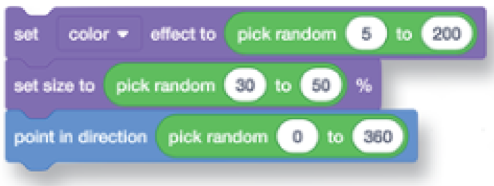
FUN WITH CODE *Building your program in chunks and testing as you go is called coding iteratively. When you iterate, you develop your program in stages, the way real coders do!*

ENHANCE YOUR SCENE

Consider enhancing your puzzle with new features:

- **User-selected difficulty:** Instead of making ten fakers, the users can input the number they want. Use an `ask` command to ask the users how many fakers they want. Then, instead of using `repeat 10` to create clones, replace the `10` with the user's `answer`. The `ask` and `answer` commands are in the Sensing category.
- **New sound:** Add new sounds from the sound library or that you record. For instance, you can make Mr. Jelly scream or make another funny sound when he's found. Just add a `play sound until` done command to Mr. Jelly's `when this sprite clicked` code block.
- **New sprite and sound:** Create an entirely different puzzle just by changing the characters and the setting!
- **New effect:** Instead of using a white jelly and the ghost effect, try using a colorful jelly and the color change effect!
- **Random sizing and random direction:** Instead of having all your sprites appear the same size and pointing in the same direction, use the `random` command to vary the sprite and clone attributes. You learn more about direction in Project 6.

Here is an example code snippet with a new effect, random sizing, and random direction that you can use to enhance your program. Note that a new scene and sprite are featured, too!



```
set [color ▾] effect to (pick random (5) to (200))
set size to (pick random (30) to (50)) %
point in direction (pick random (0) to (360))
```

SAVE, TEST, AND DEBUG YOUR PROGRAM

As you work, Scratch automatically saves your program in the cloud, so you don't have to take any special actions to save your work. Test your program by clicking Mr. Jelly and also clicking some fakers to see the program's response. Fix any bugs to ensure that the entire program works the way you want it to. (For help with debugging, see the section in Project 1 on debugging Scratch programs.)

SHARE YOUR PROGRAM WITH THE WORLD

After your program operates perfectly, it's time to share it! Set the status of your program to Share, and then add to your project page a description of your program and directions on how to run it. See Project 7 for details on sharing your programs.

BIG IDEAS IN THE PROJECT

Cloning: *Cloning* means using an object (such as a sprite) to make identical copies of the object. The copies are called *clones*. The clones *inherits* (receives) all the attributes of the *parent* (original) object. These attributes can be changed later.

Event-driven programming: In an *event-driven program,* events affect how the program runs. An event can be a user action such as a button click. In Jellyfish Jumble, events are triggered by the user clicking the green flag, the program making faker clones, and the user clicking Mr. Jelly.

Iterate: *Iterate* means to code, test, and debug the code in stages, building up the completed program.

Parallel processing: When the computer program runs different blocks of code at the same time (in *parallel*), two or more things happen in your program at once. In Jellyfish Jumble, the green flag starts the execution of code blocks on the backdrop and both sprites, all at the same time. Other processes happen in parallel, too. For example, the background music continues looping forever while all the jellyfish swish.

Program sequence: Computer code runs one command after another, in order.

Random: In programming, a *random* number has different values each time you run your code. For toys, games, and puzzles, users want to have some variety as they play. Randomness creates that variety. In Jellyfish Jumble, you send the Jellyfish sprites to random places in the water. You use randomness also to ghost all the fakers by different amounts. When coding randomness, you set the range of values (usually, low to high) that your random number can have.

Repetition: Repetition, or *looping,* of computer code lets you make sections of your code run many times, without having to write that code over and over. You can create loops in different ways. In this project, you use a `forever` loop so that the animation keeps going and music keeps playing until the program stops. You also use a `repeat` loop to make a section of code repeat a known number of times. You use more repetition and different types of loops later in this book.

User interface (UI): The *user interface* is what users see onscreen and the way they use your program. The user interface in this project is a puzzle that shows a setting and funny characters. The users must click or tap the screen to play.