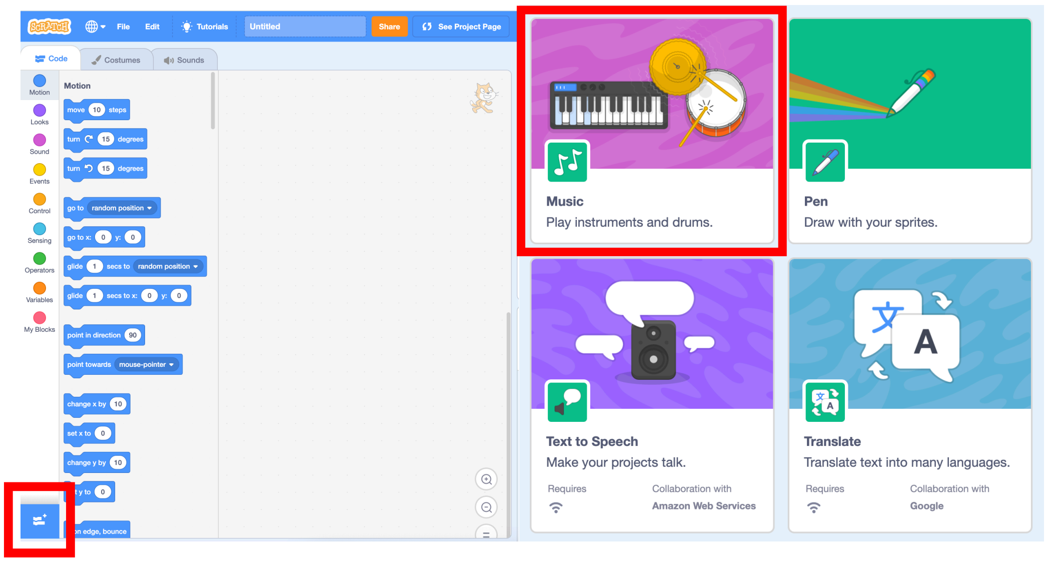
♪ GCC Make Music with Code! ♪

Activity 1: Learn songwriting with code

Open the Scratch project template by typing the following link into the search bar:

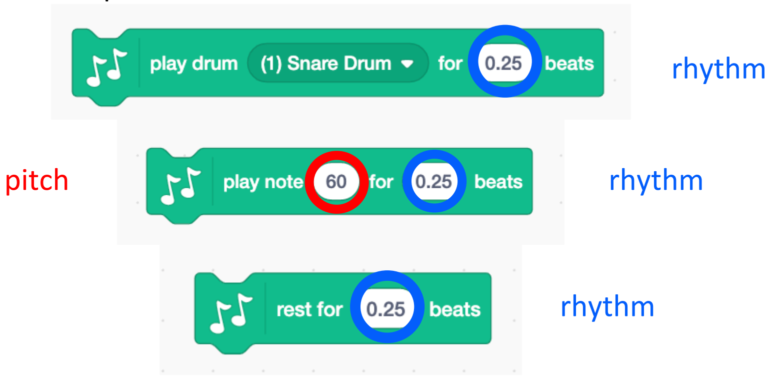
https://scratch.mit.edu/projects/723210418/editor/

Add the Music Extension to your Scratch workspace by clicking the "Add Extensions" button and selecting Music.



Music is made up of **rhythm** and **pitch**. Rhythm refers to the beat in music (*when* in time you make a sound). Pitch refers to how low or how high the sound is. Tubas make low-pitched sounds, while whistles and flutes make high-pitched sounds. Notes on the left side of the piano are lower, and notes on the right side are higher.

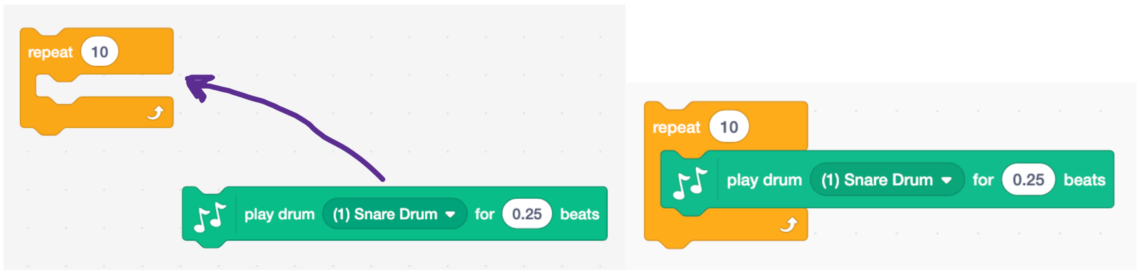
Using the Music Extension in Scratch, we can control both the rhythm and pitch of the sound we want to play using the pieces with special music commands. The most important music pieces are the "play drum" piece, the "play note" piece, and the "rest" piece.


play drum (1) Snare Drum for 0.25 beats
rhythm
pitch
play note 60 for 0.25 beats
rhythm
rest for 0.25 beats
rhythm

Controlling rhythm with code

To control the rhythm, we can set how long we want the sound to last. A higher number of beats will make the sound last longer, while a lower number will make the sound quick. This will make more sense when we string the sounds together in a *loop*. Let's test this for ourselves!

In the "Code" tab on the left, there are many colorful pieces with different commands written on them. We'll add a Control (orange) piece and one of the special Music pieces from above (by finding the pieces on the menu to the left and dragging them into your program workspace). Specifically, we'll want to insert the "play drum" piece into the "repeat 10" loop.

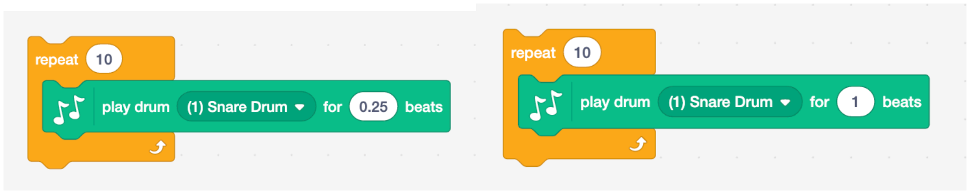


```
repeat 10

play drum (1) Snare Drum for 0.25 beats
```

```
repeat 10
    play drum (1) Snare Drum for 0.25 beats
```

To make the sound play (and run your code), click on your block of code until there is a yellow outline around it. Next, add a second identical block of code below your first block. This time, change the number of beats from 0.25 to 1. When you click on the second block, how does it sound different from the first block?

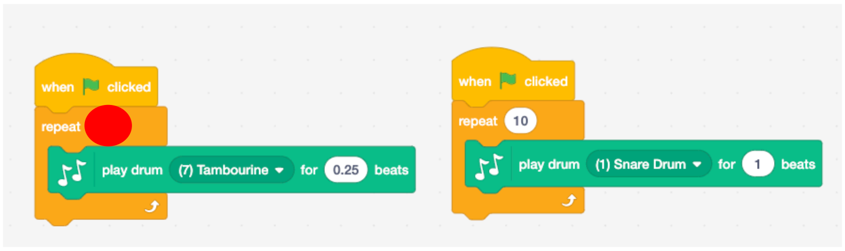


```
repeat 10
    play drum (1) Snare Drum for 0.25 beats
```

```
repeat 10
    play drum (1) Snare Drum for 1 beats
```

You can hear that the drum beats faster for lower beat durations. Something else to consider is how long you want your drum loop to last. If the drum plays for a smaller number of beats, then you need more beats to fill the same amount of time. How many times would you need to repeat the 0.25 beat drum to play for the same amount of time as 10 repeats of the 1 beat drum? (Hint: Four 0.25-beat sounds take the same amount of time as a 1 beat sound.)

To check to see if your calculation is correct, adjust the number of repeats for the 0.25-beat block to the number you calculated. Change the type of drum for the 0.25-beat sound to something different than (1) Snare Drum (e.g. (7) Tambourine). Then add a "when [green flag] clicked" piece to the beginning of both sound blocks. This will make sure that both of your drum sounds start playing at the exact same time. Your code should look something like this:



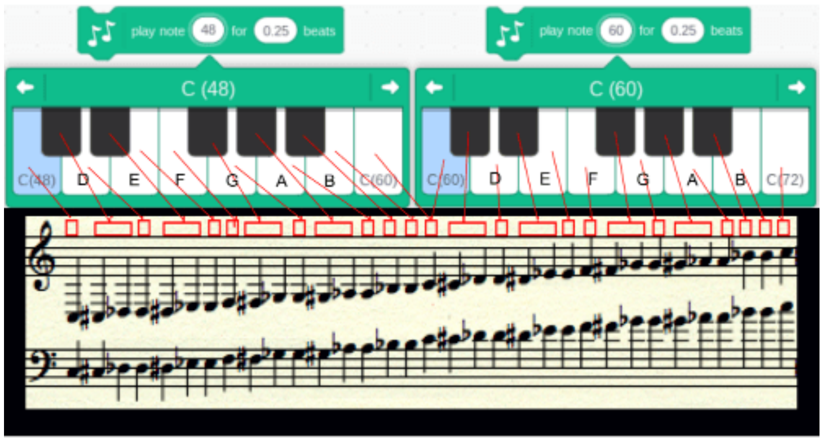
```
when clicked
repeat
    play drum (7) Tambourine for 0.25 beats
```

```
when clicked
repeat 10
    play drum (1) Snare Drum for 1 beats
```

This is an example of running code in *parallel*. This means running two different blocks of code at the same time.

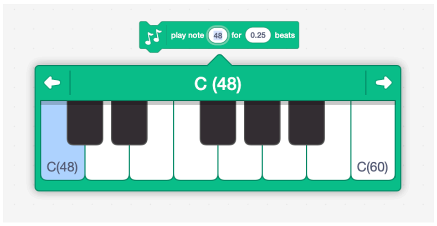
Controlling pitch with code

We control the pitch by choosing the note we want to play. In Scratch, each note is assigned a number that is listed on the piano key. Notes with higher numbers are higher, and those with lower numbers are lower in pitch.

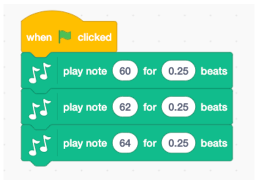


https://en.scratch-wiki.info/wiki/Play_Note_()_for_()_Beats_(block)

Western music is divided into 12 different note names (A-G) that repeat. Notes that have different numbers but the same letter name are related by a special interval called an *octave*. For example C(48) is one octave lower than C(60). Take a few minutes to explore the notes on the piano.

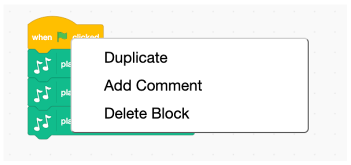


When we string different notes together in a sequence, we create a *melody*. Add three “play note” pieces to your workspace. Set each of the three note pitches to a different number. Add a “when [green flag] clicked” piece to the beginning of your melody.

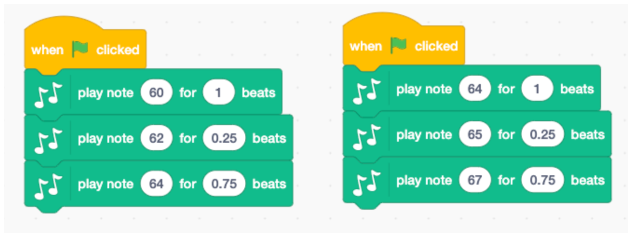

when clicked
play note 60 for 0.25 beats
play note 62 for 0.25 beats
play note 64 for 0.25 beats

By changing the number of beats each note lasts, we add rhythmic variety and structure to our melody. Change the number of beats so that each note plays for a different amount of time.

We can add another melody in parallel with our original melody. The easiest way to do this is to right click on your melody block and click “Duplicate”.



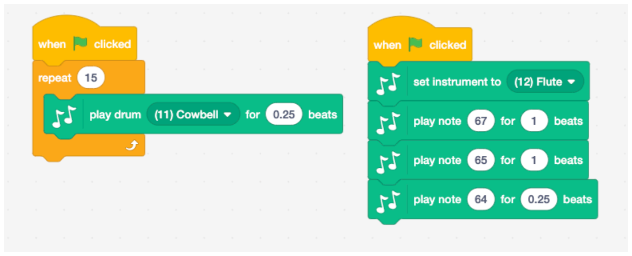
Then drag the new copy next to your original melody. Change the values of the notes in your second melody to be different from your first melody.


when clicked
play note 60 for 1 beats
play note 62 for 0.25 beats
play note 64 for 0.75 beats
when clicked
play note 64 for 1 beats
play note 65 for 0.25 beats
play note 67 for 0.75 beats

When we play two melodies at the same time, we create *harmony*. In this example, the notes change at the same time, since the number of beats for the first, second, and third notes in both melodies are the same. If you'd like, you can also change the number of beats for each note to make the notes change at different times. This will add even more texture to your tune.

Writing your first song

Let's combine everything we've learned about rhythm and pitch to write a song! Let's start with a basic structure upon which you will build. On the left, add a percussion block that repeats a "play drum" sound, and choose your favorite drum from the drop-down menu. On the right, add a melody block where you will string together "play note" pieces. For your melody block, add a "set instrument" piece and choose your favorite instrument sound. Remember to start both percussion and melody blocks with a "when [green flag] clicked" to make sure your sounds play at the same time. Your initial song setup should look something like this.


when clicked
repeat 15
play drum (11) Cowbell for 0.25 beats
when clicked
set instrument to (12) Flute
play note 67 for 1 beats
play note 65 for 1 beats
play note 64 for 0.25 beats

Don't worry about the number of repeats for your percussion block until you're done writing your melody.

Now it's your turn to make the song your own. Adjust the note values and beat numbers. Add "rest" pieces to create pauses in your rhythm or melody. Change the instruments and even add additional percussion and melody blocks (using the "Duplicate" shortcut). Be creative and have fun!

♪ ♪

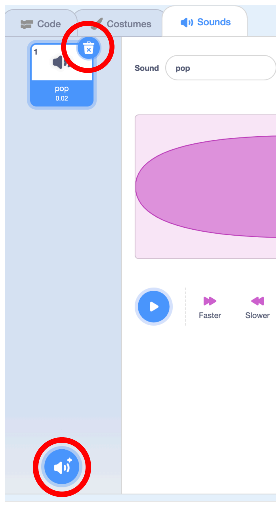
Activity 2: DJ an end-of-summer celebration!

In a second browser window, open another copy of the Scratch project template by typing the following link into the search bar:

https://scratch.mit.edu/projects/723210418/editor/

Creating your playlist

Click the "Sounds" tab on the left. Delete the default sound by clicking on the trash can in the top right corner. Then click, "Choose a sound". In general, the sounds under the "Loops" and "Percussion" tabs are the most song-like, but you're welcome to choose any sound you like! To test out a sound, hover your mouse or click on the play button over each sound. To add each sound to your coding environment, click on the name of the sound. To add each additional sound, return to the "Choose a sound" menu and click on the name of the sound. Write five sounds that you choose in the list below.



1. ______________________________

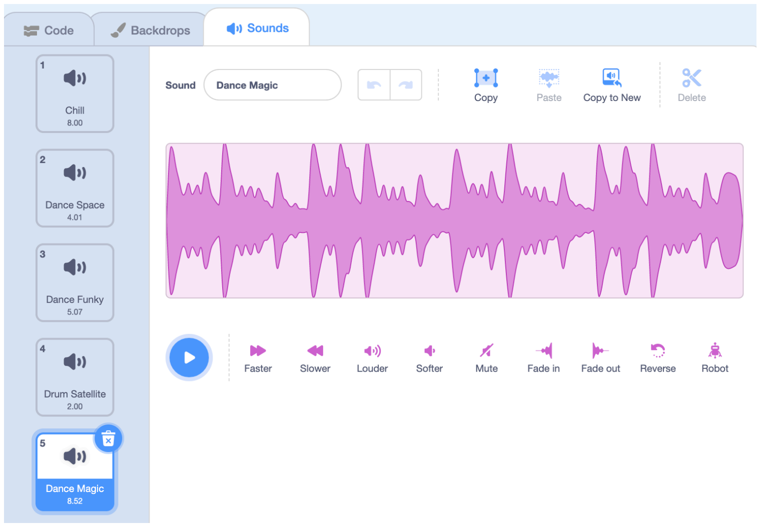
2. ______________________________

3. ______________________________

4. ______________________________

5. ______________________________

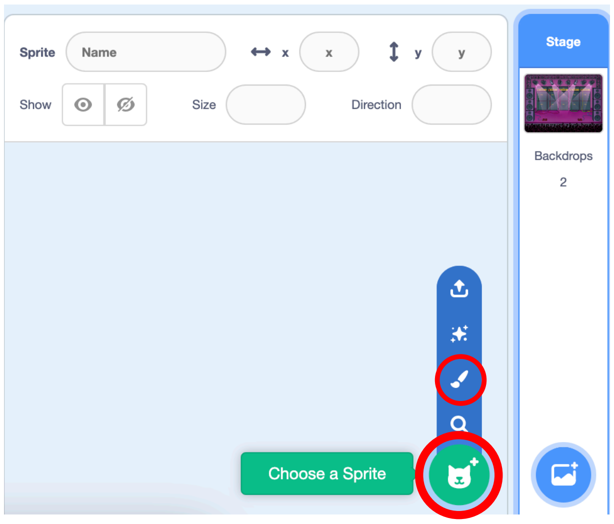
Once you've added the sounds to your playlist, your sounds tab should look something like this:



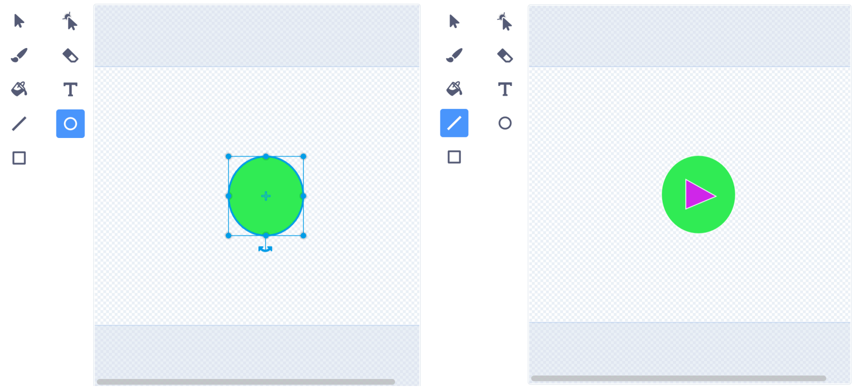
Notice that each sound has a number in the top left corner. This number will allow us to identify each sound in our code.

Creating a GUI for your music player

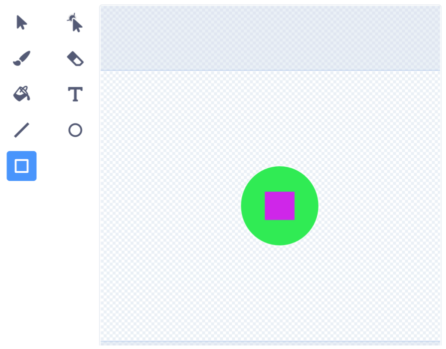
Now, we'll want to set up our *graphical user interface (GUI)* for our music player. In Scratch, we do this by adding Sprites. Click the "Add a Sprite" button. We'll want to create our own "play" button by selecting the Paint option.



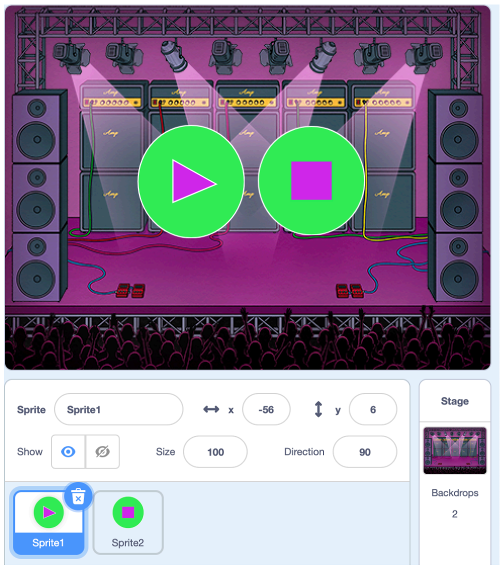
In the paint menu, click the circle icon to draw a circle in the center of the paint area. Then, use the line feature to draw three connected lines forming a triangle. This is the symbol for "play". Feel free to use the paint fill icon to add color to your triangle and/or circle.



Now, we want to create a "stop" button. Again, click the "Add a Sprite" button. Draw a circle in the center of the paint area. Then, draw a square inside the circle. This is the symbol for "stop".

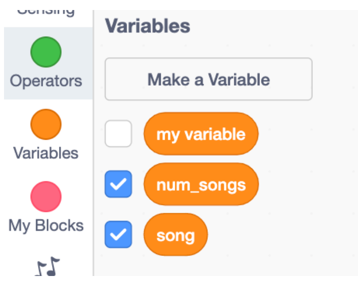


Then, organize the Sprites in your GUI to match the picture below by clicking each Sprite to drag it to the desired location.

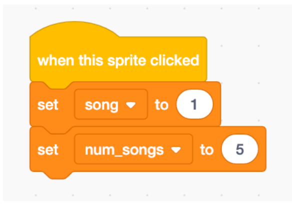


Coding your music player

Click on Sprite1, your "play" button. Go back to the "Code" tab to add some code to start playing music when the "play" button is clicked. For this, we'll need to create two variables **song** and **num_songs**. The variable **song** will keep track of the number of the song we'd like to play. The variable **num_songs** will keep track of the total number of songs in our playlist, in this case, 5. Scroll to the Variables menu and click "Make a variable". Your Variables menu should now show the name of your two new variables.

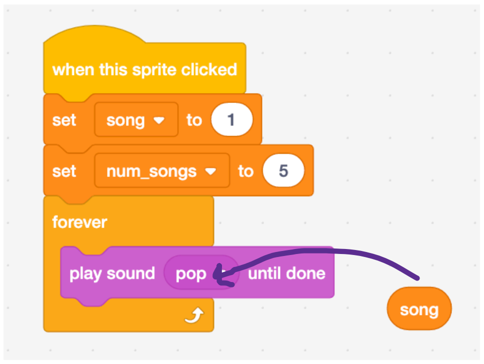


When our "play" button called Sprite1 is clicked, we want to *initialize* our **song** variable by setting it equal to the first track: 1. We also need to set our variable **num_songs** to 5.



```
when this sprite clicked
set song to 1
set num_songs to 5
```

Next, we'll need to create a loop to make sure the music doesn't stop while we're dancing! Add an orange "forever" loop. Inside the loop, first insert our **song** variable into a purple "play sound" piece.



At this point, our code would play the first song (**song**=1) in our playlist forever. How can we make our music player go to the next song when the current one finishes playing? To do this, we'll need to increase the track number (our **song** variable) by 1 after each song plays using a "change **song** by 1" piece.

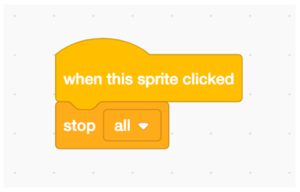
But let's consider what happens when we get to the end of our five-song playlist (song=5). The music will stop playing, and the celebration will be over. That's a bummer.

Can you figure out a way to keep the celebration going? To solve this very important problem, you'll need to use the "change **song** by 1" piece we just discussed in addition to an if/else statement. An if/else statement allows you to do one thing *if* the statement in the hexagon is true and something *else* when it is false. For this problem, you'll need to check whether the track number **song** is less than or equal to the total number of songs (**num_songs**). If the track number **song** is less than the total number of songs, change the track number by one and repeat the commands inside the loop. If the track number is equal to the last song in our playlist, we want to set our **song** variable to 1 to return to the beginning of the playlist and repeat.

Now it's your turn to code the solution to this problem in Scratch and create a music player that keeps playing!

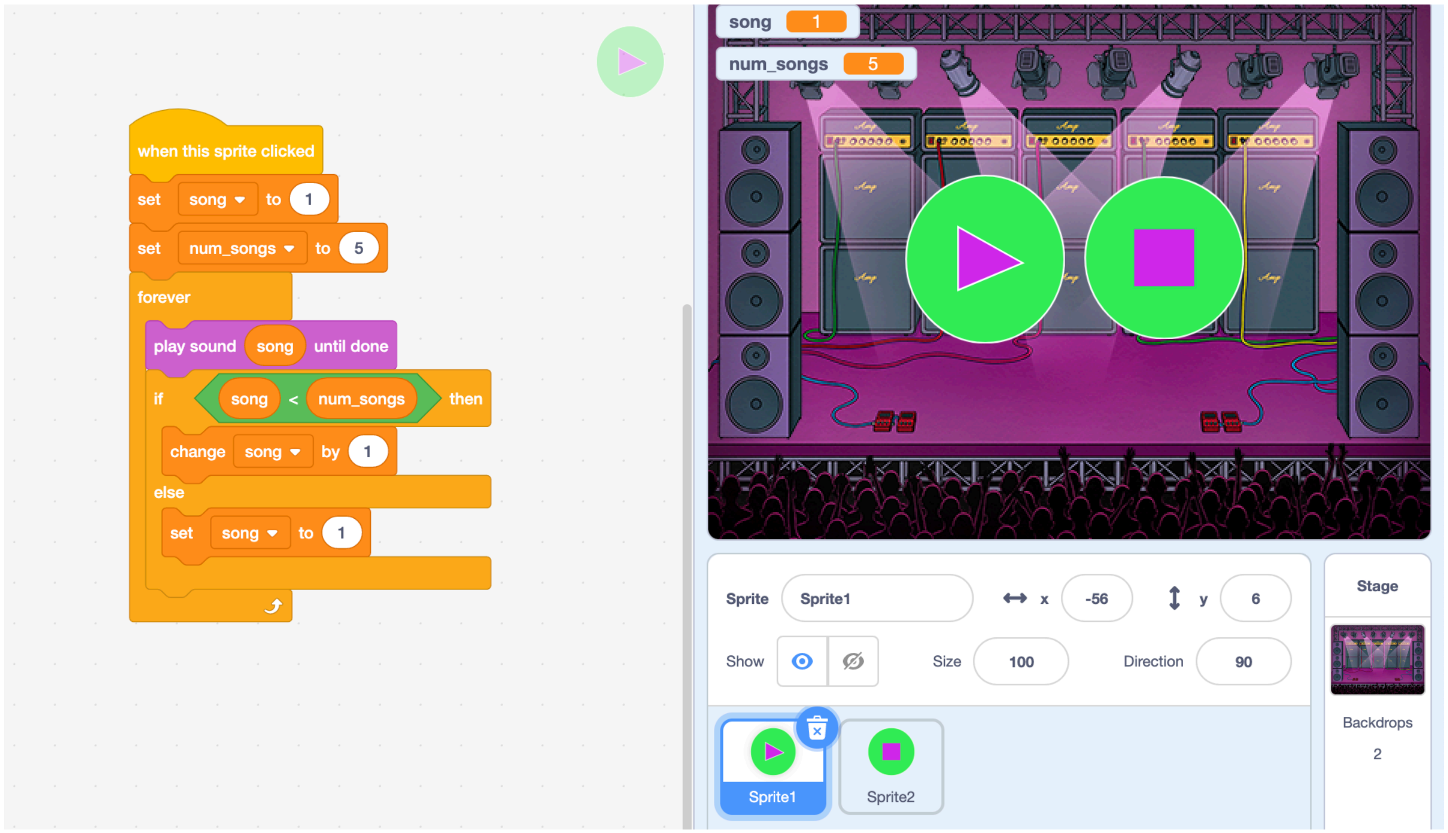
♪ ♪

Lastly, let's add some functionality to our "stop" button. In your Sprite list, click on Sprite2, your "stop" button. Then, in the "Code" tab for Sprite2, add code that stops all sounds when the stop button is clicked to exit the script. Your code should look like this:



Congrats! You wrote your own song and created a music player!

Example solution to Activity 2:



```
when this sprite clicked
set song to 1
set num_songs to 5
forever
    play sound song until done
    if song < num_songs then
        change song by 1
    else
        set song to 1
```

song 1
num_songs 5

Sprite Sprite1 x -56 y 6
Show Size 100 Direction 90