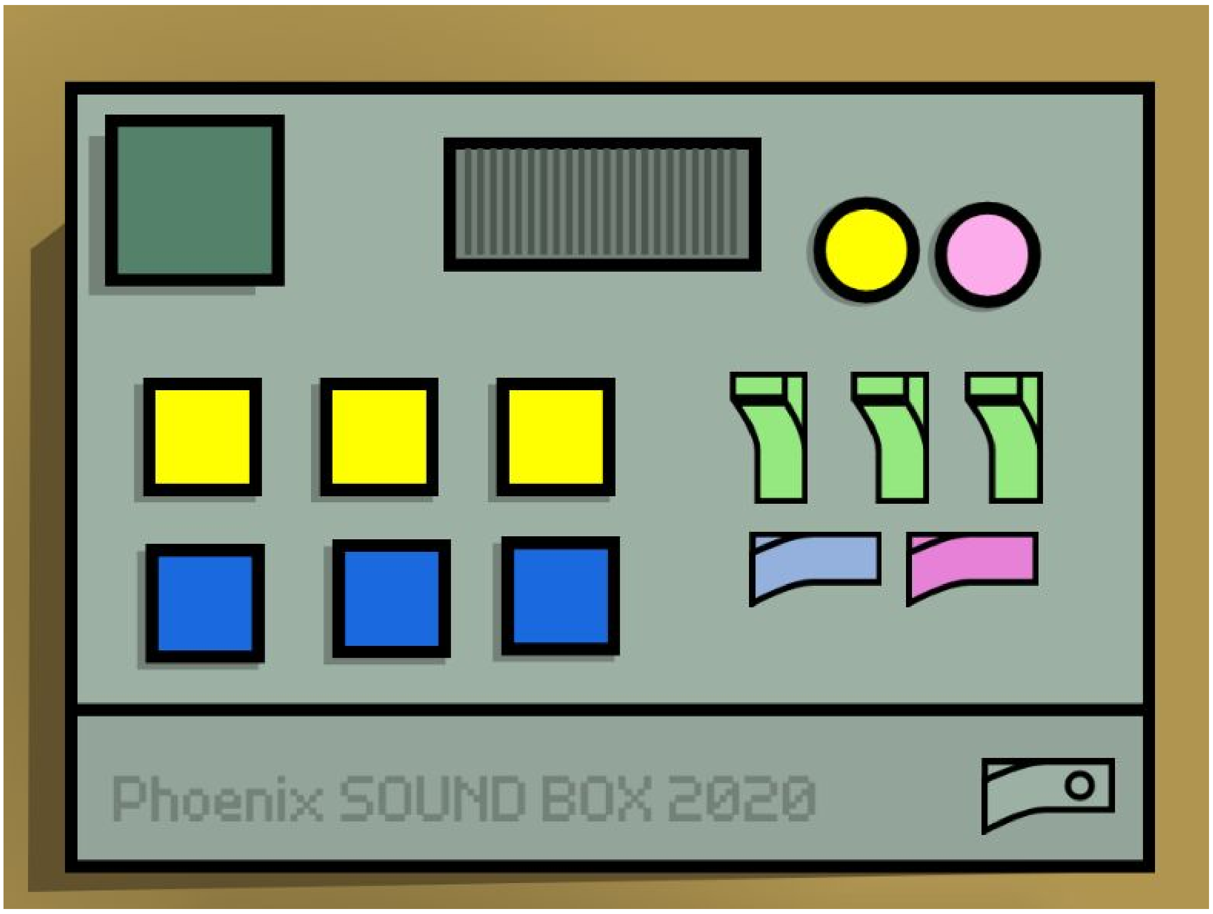
Phoenix



You will need:

Access to the Scratch website, from a computer or laptop.
(It might also work from an ipad or tablet)

This tutorial is best for people who have used Scratch a little bit before.

Setup:

If you haven't used Scratch before, click join scratch at the top and create an account. You might need your parent's email to do this, so ask for help. If you can't make an account, you can follow along, but you won't be able to save your work.

Once you've made an account follow the Phoenix Scratch studio here. This is where you can find the template versions of all the workshops we create. Feel free to remix them and post the new things you create.

Wait, what is Scratch?

Scratch is a free, online, block-based code editor. You might have used it in school. If you've been to one of our family workshops, you've probably seen us using it there.

There are lots of different tutorials available for using Scratch. Here we are going to focus on showing you ways of using Scratch that link to the work we do at Phoenix. This will include things like animation and drawing.

Sound Box

This project shows you more about using variables to create and control different effects. Remix the template and view the tutorial to create your own custom Sound Box.

To get started **remix** this project,

https://scratch.mit.edu/projects/379161910/

Step 1

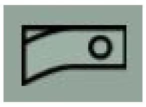
There is quite a lot going on in this project.

Before taking a look at the code, run the project, using the green flag. Click around on the Sound Box and see if you can find some of the different sounds it can make.

If you get stuck there is a hint in the next step.

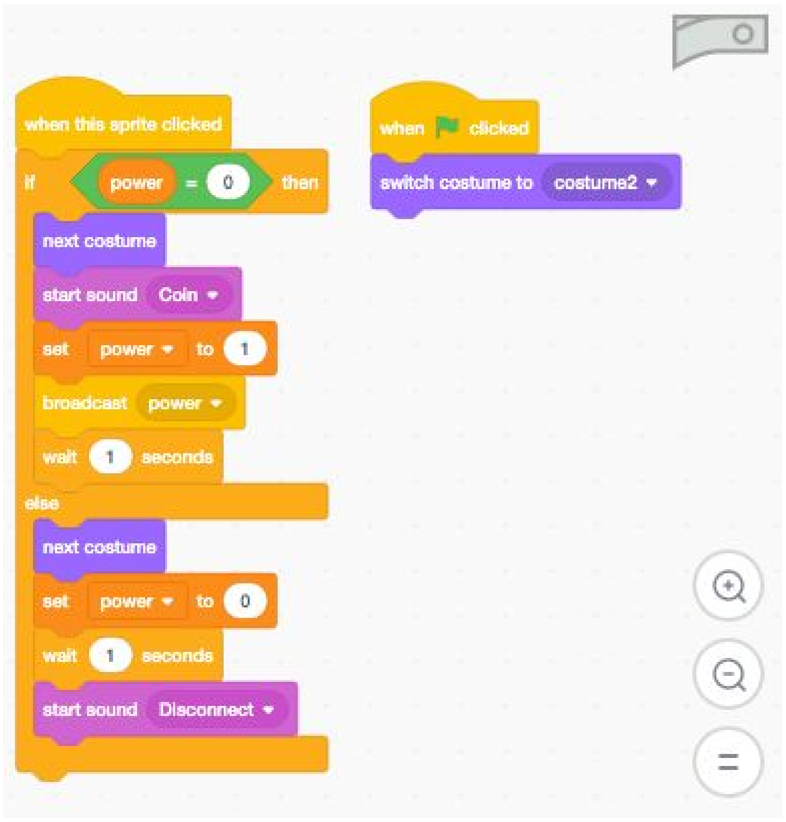
Step 2

The first thing to do is switch the box on. You can do that using this button.



Now you will notice that the other buttons and switches all work.

This is happening because this switch is controlling a variable called power. Open up the code for the power switch sprite and take a look.



```
when this sprite clicked
if <(power) = (0)> then
    next costume
    start sound (Coin ▾)
    set [power ▾] to (1)
    broadcast (power ▾)
    wait (1) seconds
else
    next costume
    set [power ▾] to (0)
    wait (1) seconds
    start sound (Disconnect ▾)
end

when ⚑ clicked
switch costume to (costume2 ▾)
```

So, when we click on the switch, first of all we check what state the power variable is in already. If it is 0 then we know that it is off, and we can switch it on. We do this by set power to 1. If the variable is 1, the else part of the code block will then reset the variable to 0.

The other parts of this block are mostly to make the interaction look and sound right. So when the box switches on we start sound Coin and when it switches off we start sound Disconnect.We also have different costumes set up so that the button appears to move.

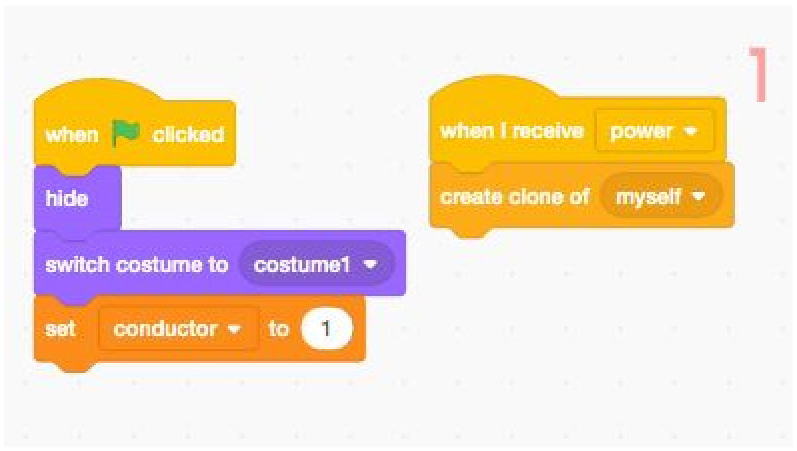
The wait 1 second block in both the if and the else section, means that we can't switch the button on and off too quickly. This is important as we need time for this code to interact with the other parts of our sound box.

Finally, you might have noticed the broadcast power message. See if you can work out what this controls. We'll reveal the answer in the next step.

Step 3

Did you find it?

Open up the Conductor sprite to find the answer. As you might have guessed by its name, this sprite is really important as it keeps the rest of our Sound Box in time.



```
when 🏴 clicked
hide
switch costume to costume1 ▾
set conductor ▾ to 1

when I receive power ▾
create clone of myself ▾
```

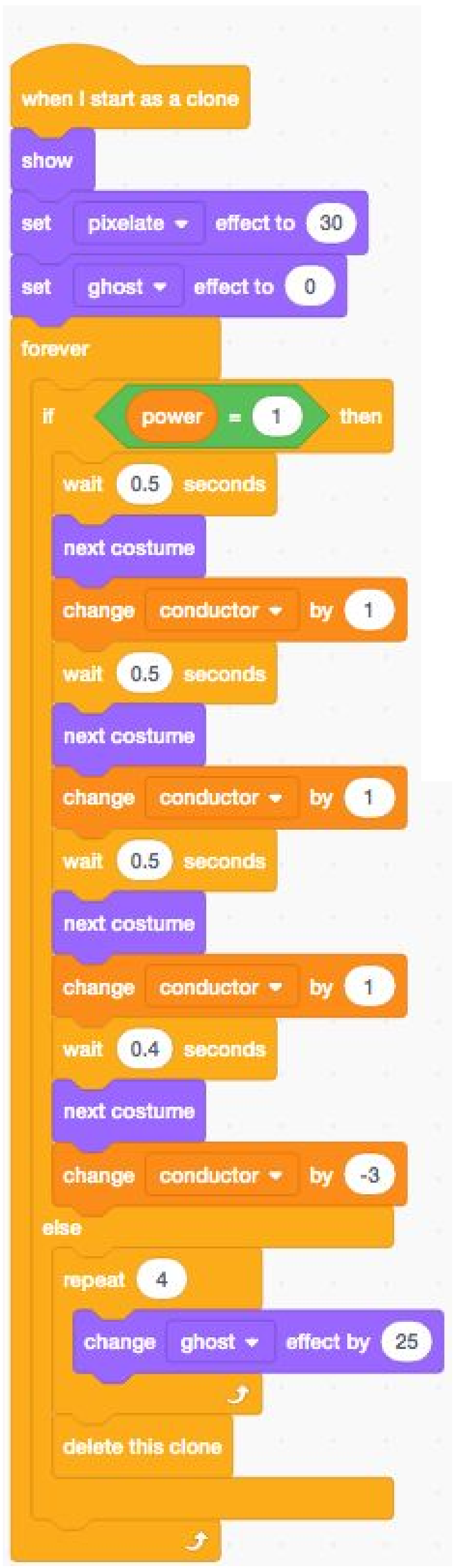
As you can see, when this sprite receives the power message, it creates a clone of itself. Each time the power is switched on we start with a fresh conductor.

The job of the conductor sprite is to count time for the rest of the sounds. Usually when using sound in Scratch, you might not need something like this. However, as we are creating a Sound Box, it is important that we have a way of starting this together.

Our project is set to a tempo of 120 BPM (beats per minute). This means that when we use the music code blocks which have commands like, play for 1 beat, each beat they count will be at 120 BPM.

This means that each second there are two beats as 120 divided by 60 equals 2.

Therefore, our conductor sprite should count a new beat every 0.5 seconds. The conductor variable allows us to make sure this count is communicated to the other sprites.



```
when I start as a clone
show
set pixelate effect to 30
set ghost effect to 0
forever
    if power = 1 then
        wait 0.5 seconds
        next costume
        change conductor by 1
        wait 0.5 seconds
        next costume
        change conductor by 1
        wait 0.5 seconds
        next costume
        change conductor by 1
        wait 0.4 seconds
        next costume
        change conductor by -3
    else
        repeat 4
            change ghost effect by 25
        delete this clone
```

You might have noticed that the last count is slightly shorter. This is because as the code loops around it takes a tiny bit longer. For this step it isn't just changing the costume and variable. Instead it also has to check again that the power is still on.

Experiment with changing this value and see what happens.

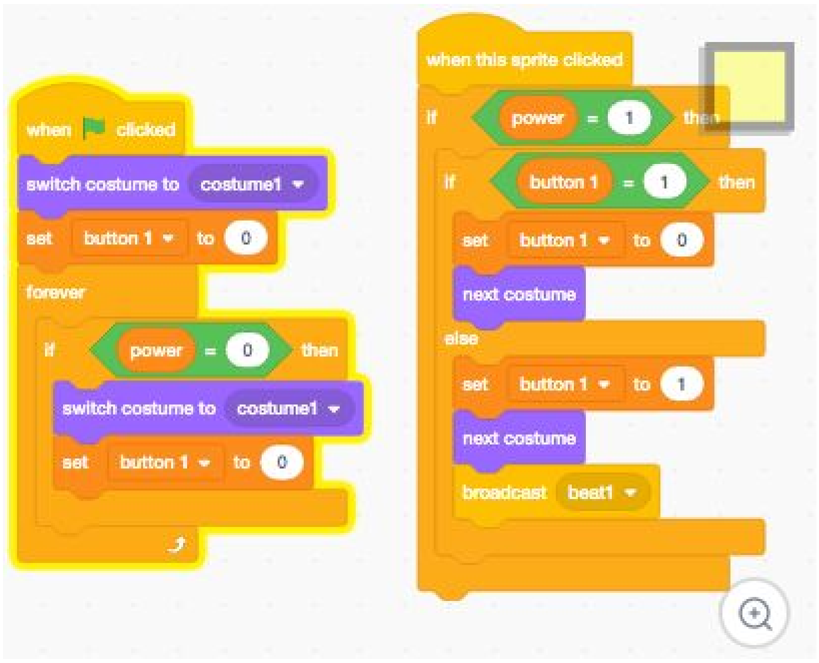
Our sound box is set up to count to 4 before resetting to 1. We are setting it up so that we have four beats in a bar.

Deciding this, means we know how many beats we have to work with, when creating the sound loops later on.

The ghost effect is a nice way of making our conductor sprite fade in and out, as if it is really a display on the sound box.

Step 4

Let's open up a button and see what's going on.



```
when ⚑ clicked
switch costume to costume1
set button 1 to 0
forever
    if power = 0 then
        switch costume to costume1
        set button 1 to 0
```

```
when this sprite clicked
if power = 1 then
    if button 1 = 1 then
        set button 1 to 0
        next costume
    else
        set button 1 to 1
        next costume
        broadcast beat1
```

So, as you can see, it is quite similar to the parts we have already looked at. We are checking whether the power is on or off, and we are also controlling a variable and message related just to this button.

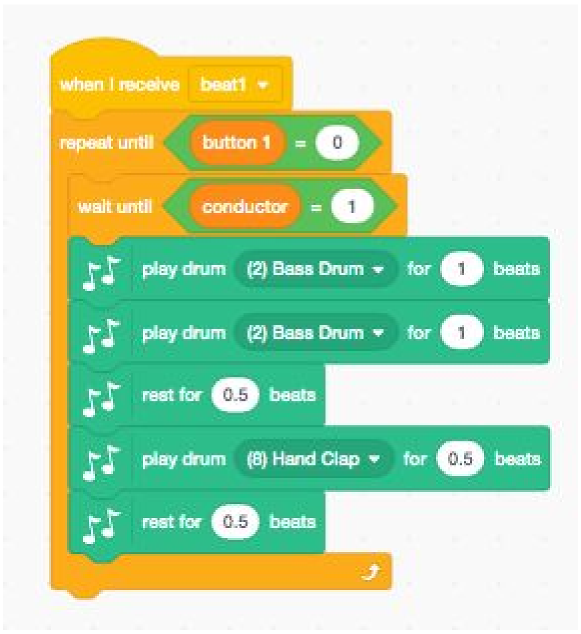
If you switch between the other square button sprites, you will notice that they are all doing this. The only difference is that the variable and message they control is different.

Step 4

Now we are ready to have a look at where most of the sound is created.
Open up the backdrop code.

This might look a bit complicated at first. But don't worry. Most of it is quite simple.

We have a bunch of blocks that look like this.



```
when I receive beat1
repeat until button 1 = 0
    wait until conductor = 1
    play drum (2) Bass Drum for 1 beats
    play drum (2) Bass Drum for 1 beats
    rest for 0.5 beats
    play drum (8) Hand Clap for 0.5 beats
    rest for 0.5 beats
```

They are listening for the button message. In this case when Button 1 sends the message beat1 this block is activated. This is set to loop around until the button 1 variable is set back to 0. Finally, before the sequence of sounds is played, the block will wait until our Conductor sprite counts back to 1 using the conductor variable.

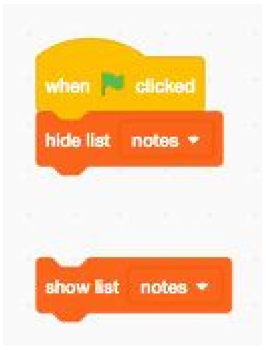
Now the music blocks are set up differently for each of these messages. One thing is similar about them all. This is that if you add up the value of all the beats, it will equal 4. Doing this means that we have created a set of beats that we know will be played and repeated each time the conductor counts to 1.

We have set up these beats and tunes as examples. But feel free to change them up as you like!

Step 5

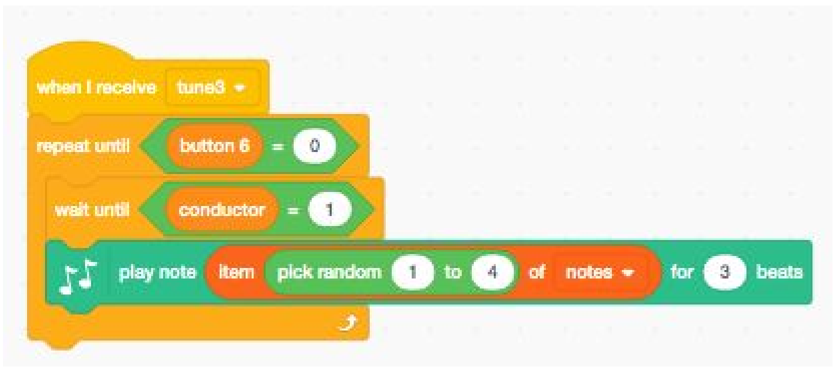
You might have noticed this already but one of our blocks related to Button 6 does something a little bit different. Instead of having the notes specified. We have a list of notes stored.

You can swap these blocks around in the Button 6 sprite to reveal the notes list. There is a table at the end of this document showing all the note values.


```
when [flag] clicked
hide list [notes ▾]

show list [notes ▾]
```

Now we are using the pick random block to select randomly from this list of notes. This allows us to create a bit of variety in the sounds we are creating.

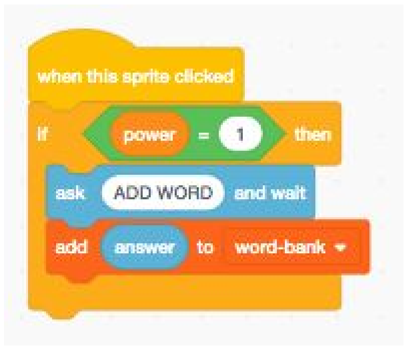

```
when I receive [tune3 ▾]
repeat until <(button 6) = (0)>
    wait until <(conductor) = (1)>
    play note (item (pick random (1) to (4)) of [notes ▾]) for (3) beats
```

Step 6

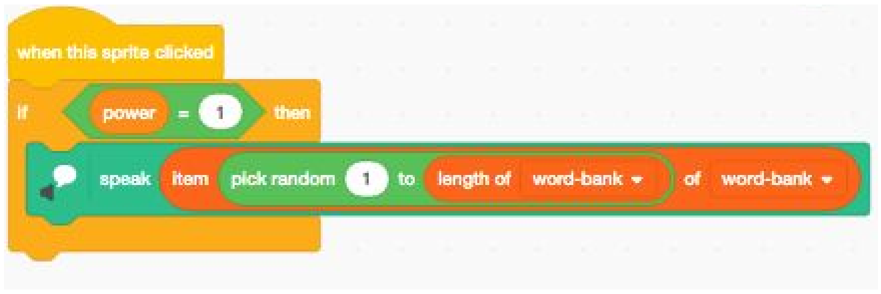
Let’s talk about the circular buttons at the top.

If you click between the code for these, you can probably work out what they are doing.

They are using the text-to-speech extension.



```
when this sprite clicked
if <(power) = (1)> then
    ask [ADD WORD] and wait
    add (answer) to [word-bank v]
```



```
when this sprite clicked
if <(power) = (1)> then
    speak (item (pick random (1) to (length of [word-bank v])) of [word-bank v])
```

With one button we can add words to a list.
With the other button we randomly select a word to be spoken.

Extra steps

Wow, you've just learned the basics of how this sound box works. There are still a few switches we didn't mention though.

- Can you work out what the other switches do, and how to control them?
- What about changing the look of the sound box by drawing a new backdrop costume?
- Can you add in some extra buttons?

Remember to share your project to the Phoenix Scratch studio!

Phoenix.org.uk
@PhoenixLeic

Note	MIDI number
C_3	48
$C\sharp_3/D\flat_3$	49
D_3	50

$D\sharp_3/E\flat_3$	51
E_3	52
F_3	53
$F\sharp_3/G\flat_3$	54
G_3	55
$G\sharp_3/A\flat_3$	56
A_3	57
$A\sharp_3/B\flat_3$	58
B_3	59
C_4 (middle C)	60
$C\sharp_4/D\flat_4$	61
D_4	62
$D\sharp_4/E\flat_4$	63
E_4	64
F_4	65
$F\sharp_4/G\flat_4$	66
G_4	67
$G\sharp_4/A\flat_4$	68
A_4	69
$A\sharp_4/B\flat_4$	70
B_4	71
C_5	72

*This table adapted from the Scratch wiki,
https://en.scratch-wiki.info/wiki/Play_Note_()_for_()_Beats_(block)#Notes